# (Asynchronous) Temporal Logics for Hyperproperties on Finite Traces


**Alberto Bombardelli** [1] Laura Bozzelli [2] César Sánchez [3] Stefano Tonetta [1]

[1]FBK, Italy

[2]University of Napoli Federico II, Italy

[3]IMDEA Software Institute, Spain



SPIN'25, 7-May-2025, Hamilton, Canada

# Outline

- Hyperproperties
- Temporal Logics for Hyperproperties
- Temporal Logics for Asynchronous Hyperproperties
- Temporal Logics for Asynchronous Hyperproperties over finite traces (SC-HyperLTL)
- Model Checking of SC-HyperLTL
- Conclusions

# Hyperproperties

[Clarkson and Schneider, "Hyperproperties" J. of Computer Security, 2010]

- Trace properties $P$ *set of trace*:

**Safety:** My program never reach a bad state
**Eventuality:** My process will eventually enter in the critical section

$$K \text{ satisfies } P \text{ if } \mathit{Traces}(K) \subseteq P$$

# Hyperproperties

[Clarkson and Schneider, "Hyperproperties" J. of Computer Security, 2010]

- Trace properties $P$ *set of trace*:

> **Safety:** My program never reach a bad state
> **Eventuality:** My process will eventually enter in the critical section

$$K \text{ satisfies } P \text{ if } Traces(K) \subseteq P$$

- Hyperproperties $H$ of systems *set of set of traces*:

> "For any execution with a secret input, there is a distinct execution Observationally equivalent without the secret input. "

$$K \text{ satisfies } H \text{ if } Traces(K) \in H$$

# Information Flow Examples

**Goguen and Meseguer non interference:** For any execution with a secret input, there is a distinct execution Observationally equivalent without the secret input.

**Observational determinism** traces with the same initial low input are indistinguishable for low users

# HyperLTL

[Clarkson, Finkbeiner, Koleini, Micinski, Rabe and Sánchez, "Temporal Logics for Hyperproperties. POST'14]

Quantifiers with trace variables: $\forall x.\, \varphi \qquad \exists x.\, \varphi$

**Syntax:**

$$\varphi ::= \forall x. \varphi \mid \exists x. \varphi \mid \psi$$

$$\psi ::= p[x] \mid X\psi \mid G\psi \mid F\psi \mid \psi U \psi \mid \psi R \psi$$

**Notes:**

- HyperLTL starts with quantifiers, then temporal formula (prenex form).
- Model Checking decidable.

# HyperLTL

[Clarkson, Finkbeiner, Koleini, Micinski, Rabe and Sánchez,“Temporal Logics for Hyperproperties. POST’14]

Quantifiers with trace variables: $\forall x.\,\varphi \qquad \exists x.\,\varphi$

**Syntax:**

$$\varphi ::= \forall x.\varphi \mid \exists x.\varphi \mid \psi$$

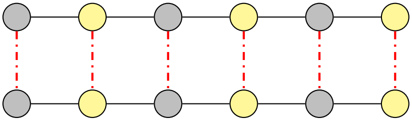
$$\psi ::= p[x] \mid X\psi \mid G\psi \mid F\psi \mid \psi U \psi \mid \psi R \psi$$

**Notes:**

- HyperLTL starts with quantifiers, then temporal formula (prenex form).
- Model Checking decidable.

All executions have the light on at the same time:

$$\forall x \forall y.\, G(\mathrm{on}[x] \leftrightarrow \mathrm{on}[y])$$



# Information Flow Examples in HyperLTL

**Goguen and Meseguer non interference:** For any execution with a secret input, there is a distinct execution Observationally equivalent without the secret input.

$$\forall x \exists y. G\lambda[y] \wedge \bigwedge_{lo \in LO} G(lo[x] = lo[y])$$

**Observational determinism** traces with the same initial low input are indistinguishable for low users

$$\forall x \forall y. \bigwedge_{v \in LI} (v[x] = v[y]) \rightarrow G \bigwedge_{v \in LO} (v[x] = v[y])$$

# Problem: Synchronous Semantics of HyperLTL

**Reactive monotonicity**: For any couple of executions of a program, the trace with smaller input has always the smaller sequence of output(s).

# Problem: Synchronous Semantics of HyperLTL

**Reactive monotonicity**: For any couple of executions of a program, the trace with smaller input has always the smaller sequence of output(s).
$\phi_{mon} ::= \forall x \forall y.(in[x] < in[y] \rightarrow G(out[x] < out[y]))$

# Problem: Synchronous Semantics of HyperLTL

**Reactive monotonicity**: For any couple of executions of a program, the trace with smaller input has always the smaller sequence of output(s).

$\phi_{mon} ::= \forall x \forall y.(in[x] < in[y] \rightarrow G(out[x] < out[y]))$

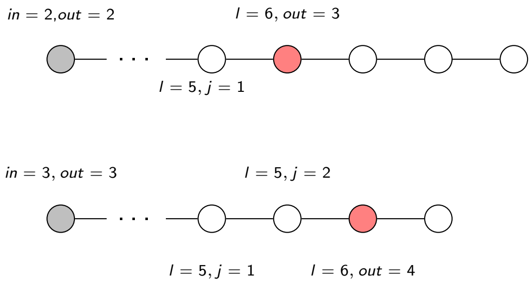
```
int x = 1, v = in;
out = v;
for (int i=0;i<10;i++){
  int j = 0;
  while (j++ < v) x++;
  out = x;
}
```

# Problem: Synchronous Semantics of HyperLTL

**Reactive monotonicity**: For any couple of executions of a program, the trace with smaller input has always the smaller sequence of output(s).

$\phi_{mon} ::= \forall x \forall y.(in[x] < in[y] \rightarrow G(out[x] < out[y]))$

```
int x = 1, v = in;
out = v;
for (int i=0;i<10;i++){
  int j = 0;
  while (j++ < v) x++;
  out = x;
}
```

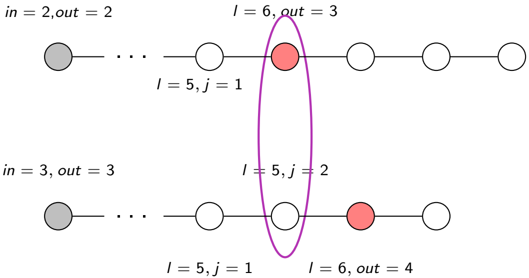


# Problem: Synchronous Semantics of HyperLTL

**Reactive monotonicity**: For any couple of executions of a program, the trace with smaller input has always the smaller sequence of output(s).

$$\phi_{mon} ::= \forall x \forall y.(in[x] < in[y] \rightarrow G(out[x] < out[y]))$$

```c
int x = 1, v = in;
out = v;
for (int i=0;i<10;i++){
  int j = 0;
  while (j++ < v) x++;
  out = x;
}
```



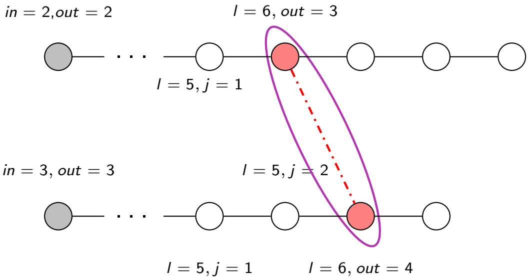

Property falsified because traces are evaluated synchronously. We want to consider them at observation points

# Problem: Synchronous Semantics of HyperLTL

**Reactive monotonicity**: For any couple of executions of a program, the trace with smaller input has always the smaller sequence of output(s).

$\phi_{mon} ::= \forall x \forall y.(in[x] < in[y] \rightarrow G(out[x] < out[y]))$

```
int x = 1, v = in;
out = v;
for (int i=0;i<10;i++){
  int j = 0;
  while (j++ < v) x++;
  out = x;
}
```




Property falsified because traces are evaluated synchronously. We want to consider them at observation points

# Asynchronous Hyperlogics (Related Works)

[Bozzelli, Peron, Sánchez, "Asynchronous Extensions of HyperLTL" LICS'21]

Stuttering HyperLTL

Context HyperLTL

# Asynchronous Hyperlogics (Related Works)

[Bozzelli, Peron, Sánchez, "Asynchronous Extensions of HyperLTL" LICS'21]

Stuttering HyperLTL Traces evaluated "synchronously" over observation points

Context HyperLTL Moves synchronously a subset of traces in a formula with $\langle C \rangle$

# Asynchronous Hyperlogics (Related Works)

[Bozzelli, Peron, Sánchez, "Asynchronous Extensions of HyperLTL" LICS'21]

Stuttering HyperLTL Traces evaluated "synchronously" over observation points

Context HyperLTL Moves synchronously a subset of traces in a formula with $\langle C \rangle$

[Bombardelli, Bozzelli, Sánchez, Tonetta, "Unifying Asynchronous Logics for Hyperproperties" FSTTCS'24]

Generalized HyperLTL$_{S+C}$

# Asynchronous Hyperlogics (Related Works)

[Bozzelli, Peron, Sánchez, "Asynchronous Extensions of HyperLTL" LICS'21]

Stuttering HyperLTL Traces evaluated "synchronously" over observation points

Context HyperLTL Moves synchronously a subset of traces in a formula with $\langle C \rangle$

[Bombardelli, Bozzelli, Sánchez, Tonetta, "Unifying Asynchronous Logics for Hyperproperties" FSTTCS'24]

Generalized HyperLTL$_{S+C}$ Combines Stuttering+Context + inner quantification

# Asynchronous Hyperlogics (Related Works)

[Bozzelli, Peron, Sánchez, "Asynchronous Extensions of HyperLTL" LICS'21]
Stuttering HyperLTL Traces evaluated "synchronously" over observation points
Context HyperLTL Moves synchronously a subset of traces in a formula with $\langle C \rangle$

[Bombardelli, Bozzelli, Sánchez, Tonetta, "Unifying Asynchronous Logics for Hyperproperties" FSTTCS'24]
Generalized HyperLTL$_{S+C}$ Combines Stuttering+Context + inner quantification

[Gutsfeld, Müller-Olm, Ohrem, "Automata and Fixpoints for Asynchronous Hyperproperties" POPL'21]
$H_\mu$: Fix-point calculus for hyperproperties
nAAWA: Asynchronous alternating automata

# Asynchronous Hyperlogics (Related Works)

[Bozzelli, Peron, Sánchez, "Asynchronous Extensions of HyperLTL" LICS'21]
Stuttering HyperLTL Traces evaluated "synchronously" over observation points
Context HyperLTL Moves synchronously a subset of traces in a formula with $\langle C \rangle$

[Bombardelli, Bozzelli, Sánchez, Tonetta, "Unifying Asynchronous Logics for Hyperproperties" FSTTCS'24]
Generalized HyperLTL$_{S+C}$ Combines Stuttering+Context + inner quantification

[Gutsfeld, Müller-Olm, Ohrem, "Automata and Fixpoints for Asynchronous Hyperproperties" POPL'21]
$H_\mu$: Fix-point calculus for hyperproperties
nAAWA: Asynchronous alternating automata

[Baumeister, Coenen, Bonakdarpour, Finkbeiner and Sánchez. "A Temporal Logic for Asynchronous Hyperproperties." CAV'21.] A-HyperLTL:

# Asynchronous Hyperlogics (Related Works)

[Bozzelli, Peron, Sánchez, "Asynchronous Extensions of HyperLTL" LICS'21]
Stuttering HyperLTL Traces evaluated "synchronously" over observation points
Context HyperLTL Moves synchronously a subset of traces in a formula with $\langle C \rangle$

[Bombardelli, Bozzelli, Sánchez, Tonetta, "Unifying Asynchronous Logics for Hyperproperties" FSTTCS'24]
Generalized HyperLTL$_{S+C}$ Combines Stuttering+Context + inner quantification

[Gutsfeld, Müller-Olm, Ohrem, "Automata and Fixpoints for Asynchronous Hyperproperties" POPL'21]
$H_\mu$: Fix-point calculus for hyperproperties
nAAWA: Asynchronous alternating automata

[Baumeister, Coenen, Bonakdarpour, Finkbeiner and Sánchez. "A Temporal Logic for Asynchronous Hyperproperties." CAV'21.] A-HyperLTL:Trace Trajectory/scheduling quantification

# Finite semantics

# Finite semantics

- Many systems/programs are finite *(not bounded)*

# Finite semantics

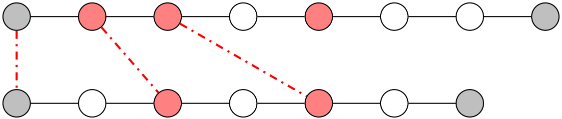
- Many systems/programs are finite *(not bounded)*
- Logics to reason over finite traces e.g. $\mathrm{LTL}_f$ [De Giacomo, Vardi "Linear temporal logic and linear dynamic logic on finite traces" IJCAI13]

# Finite semantics

- Many systems/programs are finite *(not bounded)*
- Logics to reason over finite traces e.g. $\mathrm{LTL}_f$ [De Giacomo, Vardi "Linear temporal logic and linear dynamic logic on finite traces" IJCAI13]
- Not straightforward for hyperproperties properties, in particular when Asynchronous:
  - Different lengths in traces
  - Different amount of observation
  - Assertion on local *finite* traces (singleton context)

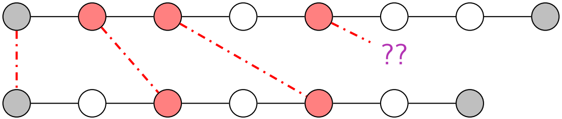
# Finite semantics

- Many systems/programs are finite *(not bounded)*
- Logics to reason over finite traces e.g. $\mathrm{LTL}_f$ [De Giacomo, Vardi "Linear temporal logic and linear dynamic logic on finite traces" IJCAI13]
- Not straightforward for hyperproperties properties, in particular when Asynchronous:
  - Different lengths in traces
  - Different amount of observation
  - Assertion on local *finite* traces (singleton context)



# Finite semantics

- Many systems/programs are finite *(not bounded)*
- Logics to reason over finite traces e.g. $\mathrm{LTL}_f$ [De Giacomo, Vardi "Linear temporal logic and linear dynamic logic on finite traces" IJCAI13]
- Not straightforward for hyperproperties properties, in particular when Asynchronous:
  - Different lengths in traces
  - Different amount of observation
  - Assertion on local *finite* traces (singleton context)




What is the correct semantics?

# SC-HyperLTL

$$\varphi ::= \overbrace{\forall x.\varphi \mid \exists x.\varphi}^{\textit{Quantifiers}} \mid \overbrace{\Gamma.\psi}^{\text{Points of interest}}$$

$$\psi ::= \overbrace{p[x] \mid \neg\psi \mid \psi \vee \psi \mid \psi \wedge \psi}^{\textit{Propositional}} \mid \overbrace{X\psi \mid N\psi \mid G\psi \mid F\psi \mid \psi U \psi \mid \psi R \psi \mid}^{\text{Future fragment}}$$

$$\underbrace{\langle x \rangle \beta[x]}_{\text{Local trace execution of } \beta} \mid \underbrace{Y\psi \mid Z\psi \mid H\psi \mid O\psi \mid \psi S \psi \mid \psi T \psi}_{\text{Past fragment}}$$

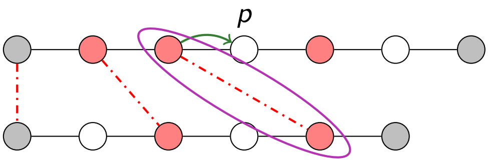
# SC-HyperLTL

$$\varphi ::= \overbrace{\forall x.\varphi \mid \exists x.\varphi}^{\textit{Quantifiers}} \mid \overbrace{\Gamma.\psi}^{\text{Points of interest}}$$

$$\psi ::= \overbrace{p[x] \mid \neg\psi \mid \psi \vee \psi \mid \psi \wedge \psi}^{\textit{Propositional}} \mid \overbrace{X\psi \mid N\psi \mid G\psi \mid F\psi \mid \psi U \psi \mid \psi R \psi}^{\text{Future fragment}} \mid$$

$$\underbrace{\langle x\rangle\beta[x]}_{\text{Local trace execution of } \beta} \mid \underbrace{Y\psi \mid Z\psi \mid H\psi \mid O\psi \mid \psi S \psi \mid \psi T \psi}_{\text{Past fragment}}$$

$\langle x\rangle X p[x]$ eval. $\sigma, i+1 \models p[x]$



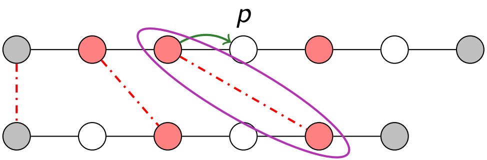
# SC-HyperLTL

$$\varphi ::= \overbrace{\forall x.\varphi \mid \exists x.\varphi}^{\textit{Quantifiers}} \mid \overbrace{\Gamma.\psi}^{\text{Points of interest}}$$

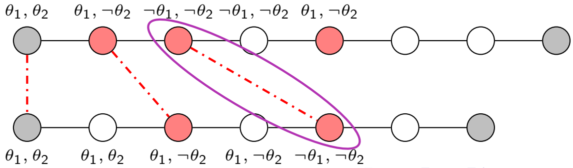
$$\psi ::= \overbrace{p[x] \mid \neg\psi \mid \psi \vee \psi \mid \psi \wedge \psi}^{\textit{Propositional}} \mid \overbrace{X\psi \mid N\psi \mid G\psi \mid F\psi \mid \psi U \psi \mid \psi R \psi}^{\text{Future fragment}} \mid$$

$$\underbrace{\langle x\rangle\beta[x]}_{\text{Local trace execution of } \beta} \mid \underbrace{Y\psi \mid Z\psi \mid H\psi \mid O\psi \mid \psi S \psi \mid \psi T \psi}_{\text{Past fragment}}$$

$\langle x\rangle Xp[x]$ eval. $\sigma, i+1 \models p[x]$




$\Gamma = \{\theta_1, \ldots, \theta_n\}$: ● iff init or $\exists\theta_j : chd(\theta_j)$



# Example properties

**Async GMNI:**

$$\forall x \exists y. LO. \langle y \rangle G \lambda[y] \wedge G \bigwedge_{lo \in LO} (lo[x] = lo[y])$$

**Async observational Determinism:**

$$\forall x \forall y. LO. \bigwedge_{v \in LI} (v[x] = v[y]) \rightarrow G \bigwedge_{v \in LO} (v[x] = v[y])$$

**Reactive Monotonicity:** $\forall x \forall y. \{out\}. G(out[x] = out[y])$

# Finite semantics of SC-HyperLTL

Singleton operator treated like trace properties

# Finite semantics of SC-HyperLTL

Singleton operator treated like trace properties

Evaluation up to the “shortest” trace

# Finite semantics of SC-HyperLTL

Singleton operator treated like trace properties

Evaluation up to the “shortest” trace

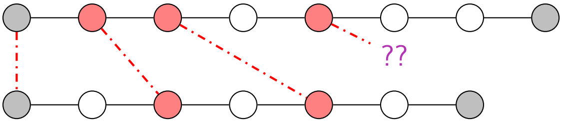
Initial points and final points are considered observations

# Finite semantics of SC-HyperLTL

Singleton operator treated like trace properties

Evaluation up to the "shortest" trace

Initial points and final points are considered observations

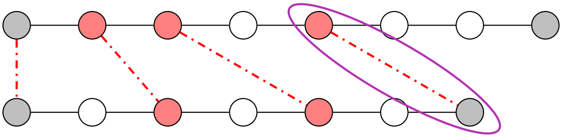


# Finite semantics of SC-HyperLTL

Singleton operator treated like trace properties

Evaluation up to the “shortest” trace

Initial points and final points are considered observations



**Local reasoning:** Don’t care if another trace is finished

**Flexibility:** Permit to compare traces with different amount of observations

**Symmetry:** Past and future are treated symmetrically

# Reversability and Past

**Reversability:** For every formula $\varphi$, $rev(rev(\varphi)) \equiv \varphi$.
Given trace assignment $\Pi$, $(\Pi, \Gamma) \models \varphi$ if and only if $(\Pi^{-1}, \Gamma_R) \models rev(\varphi)$.

Validate past and future are treated symmetrically

# Examples with finite semantics

**Async Observational Determinism:** Shortest trace

$$\forall x \forall y . LO. \bigwedge_{v \in LI} (v[x] = v[y]) \rightarrow G \left( \bigwedge_{v \in LO} (v[x] = v[y]) \right)$$

# Examples with finite semantics

**Async Observational Determinism:** Enforce #obs

$$\forall x \forall y. LO. \bigwedge_{v \in LI} (v[x] = v[y]) \rightarrow G \left( \bigwedge_{v \in LO} (v[x] = v[y]) \wedge ((\langle x \rangle N \bot) \leftrightarrow (\langle y \rangle N \bot) \right)$$

**Pre-post conditions**

$$\forall x \exists y. \emptyset. Pre(x, y) \rightarrow XPost(x, y)$$

# Model Checking

**Stuttering extension:**

- **Idea:** Reduce $\forall^*/\exists^*$ async to $\forall^*/\exists^*$ sync
- Use rewriting technique similar to [Bombardelli, Tonetta "Asynchronous Composition of Local Interface LTL Properties" NFM22]
- In principle similar to the approach used for A-HLTL [Baumeister, Coenen, Bonakdarpour, Finkbeiner and Sánchez. "A Temporal Logic for Asynchronous Hyperproperties." CAV'21.]

**Bounded observations:**

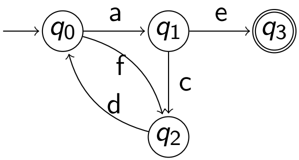
- Support $\forall\exists$ fixing at most $k$ observation points
- Traces can be unbounded
- Reduces to HyperLTL

# Model Checking via stuttering encoding: Overall idea

Extend $K$ with stuttering and halting state

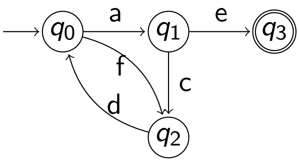
# Model Checking via stuttering encoding: Overall idea

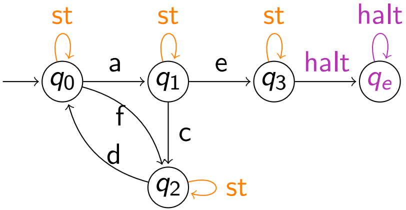
Extend $K$ with stuttering and halting state



# Model Checking via stuttering encoding: Overall idea

Extend $K$ with stuttering and halting state

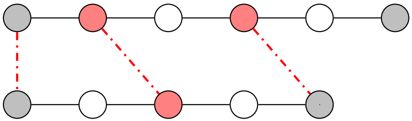


# Model Checking via stuttering encoding: Overall idea

Extend $K$ with stuttering and halting state

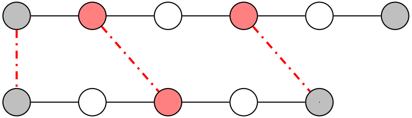
Align traces over $\Gamma$-points



# Model Checking via stuttering encoding: Overall idea

Extend $K$ with stuttering and halting state

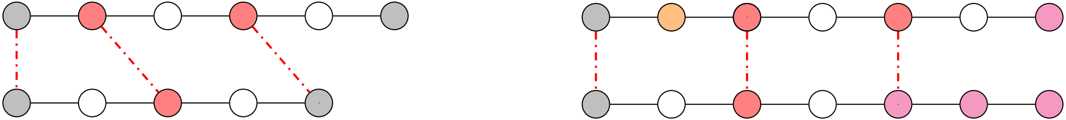
Align traces over $\Gamma$-points



# Model Checking via stuttering encoding: Overall idea

Extend $K$ with stuttering and halting state

Align traces over $\Gamma$-points



$$G(\neg end[x] \wedge \neg end[y] \rightarrow (\Phi_\Gamma[x] \leftrightarrow \Phi_\Gamma[y]))$$

# Model Checking via stuttering encoding: Overall idea

Extend $K$ with stuttering and halting state

Align traces over $\Gamma$-points

Rewrite temporal operators to evaluate temporal operators at observation points synchronously

$$\langle x \rangle \beta[x] \Rightarrow \mathcal{R}_{st[x] \vee halt[x]}(\beta[x])$$
$$\varphi_1 U \varphi_2 \Rightarrow \mathcal{R}_{\Phi_\Gamma}(\varphi_1 U \varphi_2)$$

$$\Phi_\Gamma := \bigvee_{\theta \in \Gamma} (Y\theta \leftrightarrow \neg\theta) \vee init \vee last$$

# Model Checking via stuttering encoding: Overall idea

Extend $K$ with stuttering and halting state

Align traces over $\Gamma$-points

Rewrite temporal operators to evaluate temporal operators at observation points synchronously

$$\langle x \rangle \beta[x] \Rightarrow \mathcal{R}_{st[x] \vee halt[x]}(\beta[x])$$
$$\varphi_1 U \varphi_2 \Rightarrow \mathcal{R}_{\Phi_\Gamma}(\varphi_1 U \varphi_2)$$

$$\Phi_\Gamma := \bigvee_{\theta \in \Gamma} (Y\theta \leftrightarrow \neg\theta) \vee init \vee last$$

# Model Checking via stuttering encoding: Overall idea

Extend $K$ with stuttering and halting state

Align traces over $\Gamma$-points

Rewrite temporal operators to evaluate temporal operators at observation points synchronously

# Model checking with Bounded observation

Assume at most K observations but unbounded length of trace

# Model checking with Bounded observation

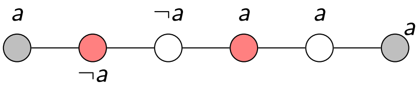
Assume at most K observations but unbounded length of trace

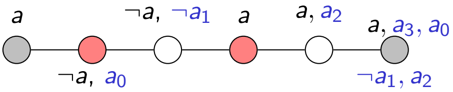
Add past history variables to mimic $i$-th observation of a variable $v$

# Model checking with Bounded observation

Assume at most K observations but unbounded length of trace

Add past history variables to mimic $i$-th observation of a variable $v$



# Model checking with Bounded observation

Assume at most K observations but unbounded length of trace

Add past history variables to mimic $i$-th observation of a variable $v$

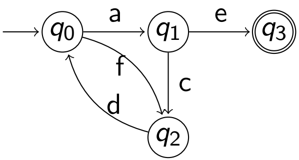
Padding of K with end state to ensure same length

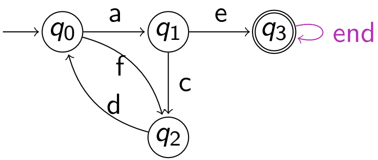
# Model checking with Bounded observation

Assume at most K observations but unbounded length of trace

Add past history variables to mimic $i$-th observation of a variable $v$

Padding of K with end state to ensure same length



# Model checking with Bounded observation

Assume at most K observations but unbounded length of trace

Add past history variables to mimic *i*-th observation of a variable *v*

Padding of K with end state to ensure same length

Evaluate in a BMC-like manner at the end of the traces (sync because of padding)

# Model checking with Bounded observation

Assume at most K observations but unbounded length of trace

Add past history variables to mimic $i$-th observation of a variable $v$

Padding of K with end state to ensure same length

Evaluate in a BMC-like manner at the end of the traces (sync because of padding)

$$
\begin{aligned}
{}^{\rho}[\![v[x]]\!]_i^k &:= v_i[x] & {}^{\rho}[\![\neg v[x]]\!]_i^k &:= \neg v_i[x]\\
{}^{\rho}[\![\psi_1 \vee \psi_2]\!]_i^k &:= {}^{\rho}[\![\psi_1]\!]_i^k \vee {}^{\rho}[\![\psi_2]\!]_i^k & {}^{\rho}[\![\psi_1 \wedge \psi_2]\!]_i^k &:= {}^{\rho}[\![\psi_1]\!]_i^k \wedge {}^{\rho}[\![\psi_2]\!]_i^k\\
{}^{\rho}[\![\mathbf{X}_\Gamma \psi]\!]_i^k &:= \neg end_\Gamma^{i+1} \wedge {}^{\rho}[\![\psi]\!]_{i+1}^k & {}^{\rho}[\![\mathbf{N}_\Gamma \psi]\!]_i^k &:= end_\Gamma^{i+1} \vee {}^{\rho}[\![\psi]\!]_{i+1}^k\\
{}^{\rho}[\![\psi_1 \,\mathbf{U}_\Gamma\, \psi_2]\!]_i^k &:= {}^{\rho}[\![\psi_2]\!]_i^k \vee ({}^{\rho}[\![\psi_1]\!]_i^k \wedge {}^{\rho}[\![\mathbf{X}\top]\!]_{i+1}^k \wedge {}^{\rho}[\![\psi_1 \,\mathbf{U}_\Gamma\, \psi_2]\!]_{i+1}^k)\\
{}^{\rho}[\![\psi_1 \,\mathbf{R}_\Gamma\, \psi_2]\!]_i^k &:= {}^{\rho}[\![\psi_2]\!]_i^k \wedge ({}^{\rho}[\![\psi_1]\!]_i^k \vee {}^{\rho}[\![\mathbf{N}\bot]\!]_{i+1}^k \vee {}^{\rho}[\![\psi_1 \,\mathbf{R}_\Gamma\, \psi_2]\!]_{i+1}^k)
\end{aligned}
$$

# Model checking with Bounded observation

Assume at most K observations but unbounded length of trace

Add past history variables to mimic *i*-th observation of a variable *v*

Padding of K with end state to ensure same length

Evaluate in a BMC-like manner at the end of the traces (sync because of padding)

# Proof of Concept Implementation (reducing to nuXmv ∀∀ and AutoHyper ∀∀-∀∃)

| Name | Method | Bound | Tool | Time (sec) | Res |
|---|---|---|---|---|---|
| Process $n = 6$ | $k$-bound (Alg.2) | 7 | nuXmv | 4.17 | True |
| Process $n = 6$ | $k$-bound (Alg.2) | 7 | AutoHyper | MO | - |
| Process $n = 2$ | $k$-bound (Alg.2) | 3 | nuXmv | 2.82 | True |
| Process $n = 2$ | $k$-bound (Alg.2) | 3 | AutoHyper | 5.54 | True |
| Process $n = 6$ | Stuttering (Alg.1) | - | nuXmv | 2.65 | True |
| Process $n = 6$ (bug) | $k$-bound (Alg.2) | 7 | nuXmv | 3.33 | False |
| Process $n = 6$ (bug) | Stuttering (Alg.1) | - | nuXmv | 3.02 | False |
| Process q. alt. $n = 2$ | $k$-bound (Alg.2) | 3 | AutoHyper | 34.93 | True |
| Process q. alt. $n = 3$ | $k$-bound (Alg.2) | 4 | AutoHyper | TO | - |
| Motivating example | $k$-bound (Alg.2) | 3 | nuXmv | 34.64 | False* |
| Motivating example | Stuttering (Alg.1) | - | AutoHyper | 6.19 | True |
| Battery Sensor | Stuttering (Alg.1) | - | nuXmv | 6.03 | True |
| Battery Sensor | $k$-bound (Alg.2) | 4 | AutoHyper | TO | - |
| Battery Sensor | $k$-bound (Alg.2) | 7 | nuXmv | 11.86 | False* |
| Battery Sensor (bugged) | Stuttering (Alg.1) | - | nuXmv | 3.04 | True |

# Conclusion and Future Work

**Conclusion:**

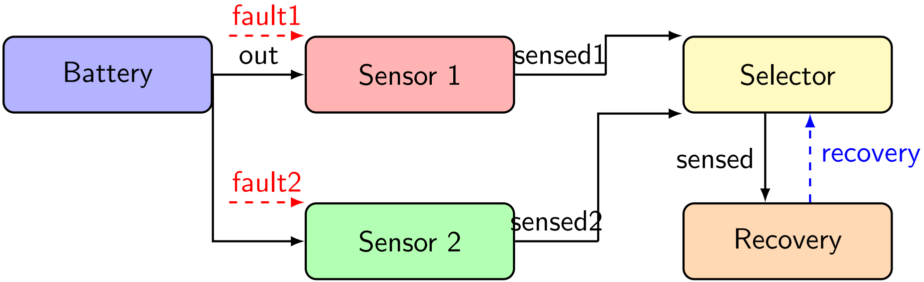
- Presented decidable asynchronous hyperlogics over finite traces
- Validated semantics and showed expressibility over interesting examples
- Provided a proof of concept tool for the verification of the logics.

**Future works:**

- Simplify Algorithm 2 and investigate more efficient techniques with bounded observations
- Relax bounded observation requirement
- Direct reduction of Alg. 1 to $\text{LTL}_f$
- ...

# Appendix

# FDIR of Battery Sensor




$$\phi_{rec} := \forall x.\forall y.\{sensed\}.G(\phi_{obseq} \wedge \langle x\rangle(faulted[x]) \rightarrow \langle y\rangle F^{\leq d} rcv[y])$$

$$\phi_{obseq} := H(sensed[x] \leftrightarrow sensed[y])$$
$$faulted := (\neg chd(sensed)\ \mathbf{S}\ fault)$$