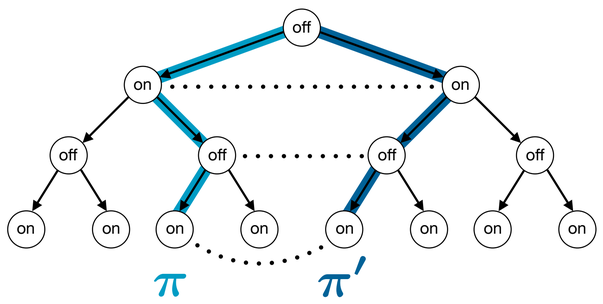
Thu, 17:30
Sun I

Monitoring Hyperproperties

Bernd Finkbeiner, Christopher Hahn, Marvin Stenger, Leander Tentrup

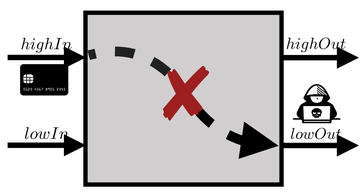
HyperLTL: A Temporal Logic for Hyperproperties

$$\forall\pi.\forall\pi'.\ \square(on_\pi \leftrightarrow on_{\pi'})$$



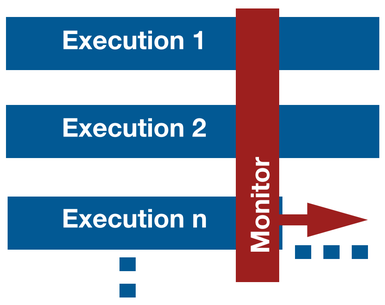
$$\varphi := \ \forall\pi.\ \varphi \mid \exists\pi.\ \varphi \mid \psi$$
$$\psi := \ a_\pi \mid \psi \wedge \psi \mid \neg\psi \mid \bigcirc\psi \mid \psi\ \mathcal{U}\ \psi$$

- HyperLTL handles relations between traces (hyperproperties) by explicit quantification
- Can express information-flow policies, symmetry, Hamming-distance and alike.
- For example, noninterference:



$$\forall\pi.\forall\pi'.(o_\pi = o_{\pi'})\,\mathcal{W}(i_\pi \neq i_{\pi'})$$

Challenges in Monitoring Hyperproperties



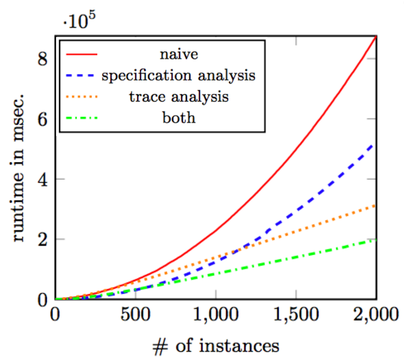
- Previously seen executions have to be stored to determine the satisfaction of a hyperproperty
- Algorithmic workload increases with the number of incoming traces
- Our model: monitor observes incoming traces sequentially

Optimizations in Monitoring Hyperproperties

Trace Analysis:

- Prunes redundant execution traces that pose strictly less requirements on future traces than others
- Small overhead, because an on the fly implementation

→ **Keeps the minimal set of traces needed to provide a counter example at the earliest possible point in time**



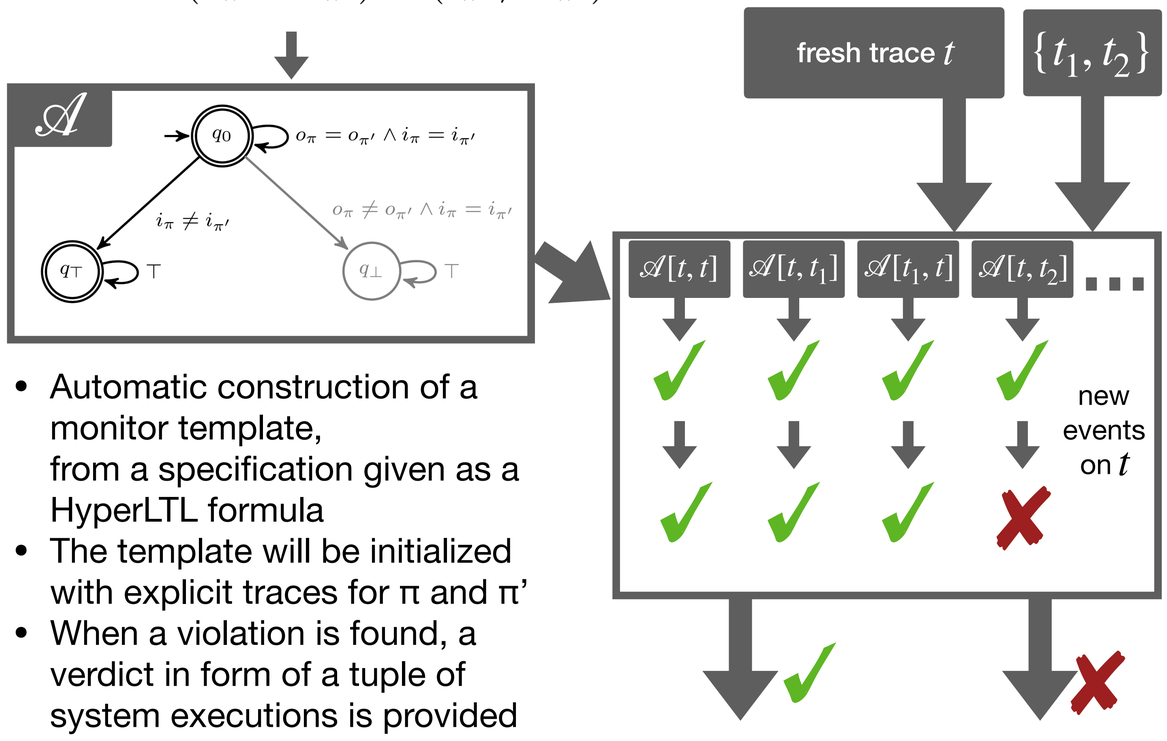
Specification Analysis:

- Reduces the number of comparisons between incoming traces by analyzing the HyperLTL formula
- The HyperLTL SAT solver EAHyper is used to detect whether a formula is reflexive, symmetric or transitive

→ **Reduces the algorithmic workload needed to detect a violation**

Automaton-based Monitoring Approach

$$\forall\pi.\forall\pi'.(o_\pi = o_{\pi'})\,\mathcal{W}(i_\pi \neq i_{\pi'})$$



- Automatic construction of a monitor template, from a specification given as a HyperLTL formula
- The template will be initialized with explicit traces for π and π'
- When a violation is found, a verdict in form of a tuple of system executions is provided

→ **Despite the optimization efforts, at least quadratic in the number of incoming traces**

Constraint-based Monitoring Approach

$$\forall\pi,\pi'.\,(out_\pi \leftrightarrow out_{\pi'})\,\mathcal{W}(in_\pi \nleftrightarrow in_{\pi'})$$

$$(in_\pi \nleftrightarrow in_{\pi'}) \vee (out_\pi \leftrightarrow out_{\pi'}) \wedge \bigcirc((out_\pi \leftrightarrow out_{\pi'})\,\mathcal{W}(in_\pi \nleftrightarrow in_{\pi'}))$$

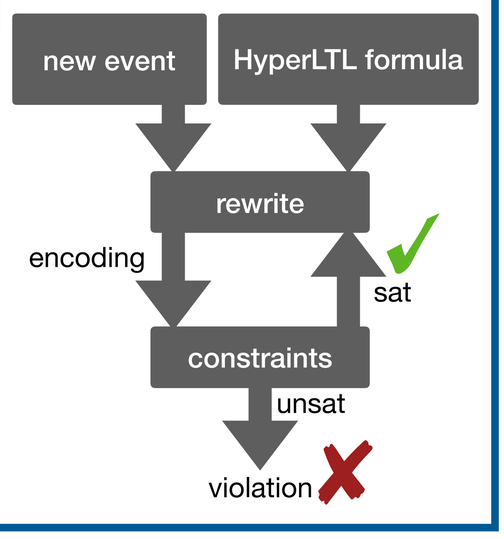
event {in, out}

$$(\top \nleftrightarrow in) \vee (\top \leftrightarrow out) \wedge \bigcirc((out_\pi \leftrightarrow out_{\pi'})\,\mathcal{W}(in_\pi \nleftrightarrow in_{\pi'}))$$

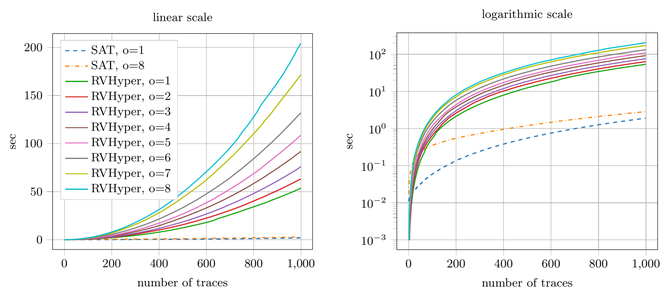
$$\neg in \vee out \wedge \bigcirc((out_\pi \leftrightarrow out_{\pi'})\,\mathcal{W}(in_\pi \nleftrightarrow in_{\pi'}))$$

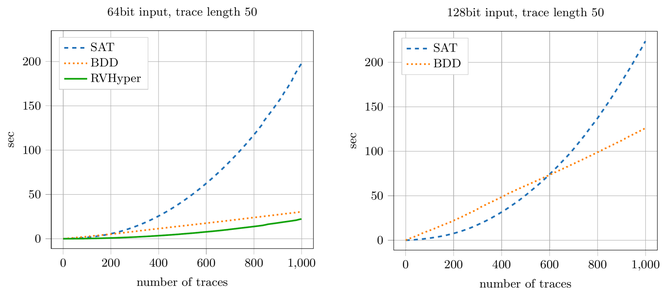
- Algorithm rewrites a HyperLTL formula and single event into a constraint composed of an LTL part and a HyperLTL part
- Incrementally build a constraint system by encoding the HyperLTL part as a variable

→ **Solely store necessary information (instead of entire traces) to detect violations of a hyperproperty**



Experimental Results: Comparing the two Approaches





- Guarded invariants (top)

$$\forall\pi,\pi'.\Diamond(\vee_{i\in I} i_\pi \nleftrightarrow i_{\pi'}) \rightarrow \square(P(\pi) \leftrightarrow P(\pi'))$$

- Noninterference (bottom)

$$\forall\pi.\forall\pi'.(o_\pi = o_{\pi'})\,\mathcal{W}(i_\pi \neq i_{\pi'})$$

- Dependencies between input and output signals in hardware designs (table)

$$\forall\pi_1\forall\pi_2.\,(\boldsymbol{o}_{\pi_1} \leftrightarrow \boldsymbol{o}_{\pi_2})\,\mathcal{W}(\bar{\boldsymbol{i}}_{\pi_1} \nleftrightarrow \bar{\boldsymbol{i}}_{\pi_2})$$

instance	# traces	length	time RVHyper	time SAT	time BDD
XOR1	19	5	12ms	47ms	49ms
XOR2	1000	5	16913ms	996ms	1666ms
counter1	961	20	9610ms	8274ms	303ms
counter2	1353	20	19041ms	13772ms	437ms
MUX1	1000	5	14924ms	693ms	647ms
MUX2	80	5	121ms	79ms	81ms

Implementation

- Two approaches for monitoring hyperproperties: automaton-based and constraint-based
- Constraint-based approach ensures more fine grained storage handling
- SAT implementation especially suited for guarded invariants
- Source code + Benchmarks are available: https://www.react.uni-saarland.de/tools/

→ **A complete tool box for the efficient runtime verification of hyperproperties in reactive systems**

Reactive Systems Group
Computer Science | Saarland University

European Research Council
Established by the European Commission
OSARES

Bernd Finkbeiner, Christopher Hahn, Marvin Stenger, and Leander Tentrup: **Monitoring Hyperproperties** (RV 2017)
Bernd Finkbeiner, Christopher Hahn, Marvin Stenger, and Leander Tentrup: **RVHyper: A Runtime Verification Tool for Temporal Hyperproperties** (TACAS 2018)
Christopher Hahn, Marvin Stenger, and Leander Tentrup: **Constraint-based Monitoring of Hyperproperties** (TACAS 2019)