UNIVERSITÄT ZU LÜBECK
INSTITUTE FOR SOFTWARE ENGINEERING
AND PROGRAMMING LANGUAGES

isp

Runtime Verification of AUTOSAR Timing Extensions

Max Friese* Hannes Kallwies Martin Leucker Martin Sachenbacher
Hendrik Streichhahn **Daniel Thoma**

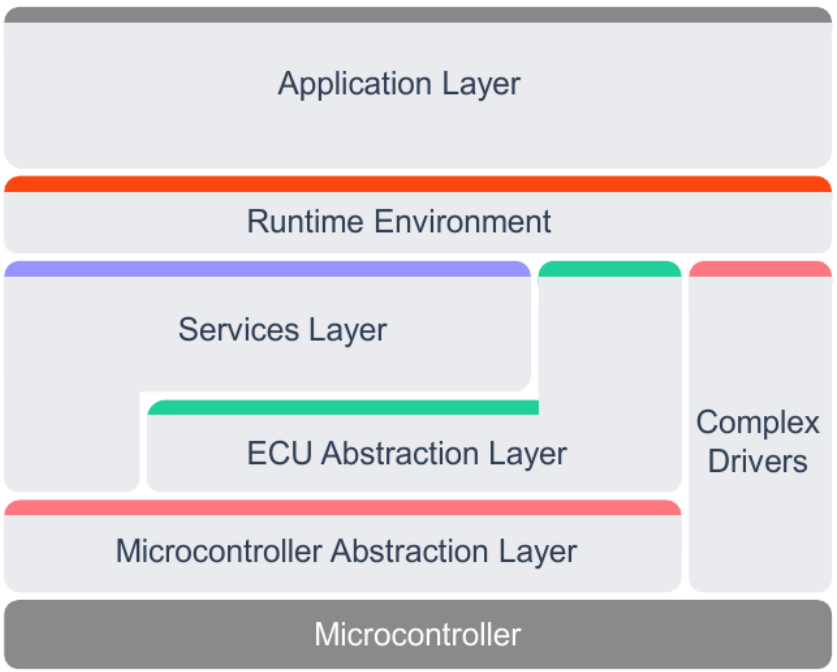
Institute for Software Engineering and Programming Languages, University of Lübeck

*Mercedes-Benz AG

RTNS, June 2022

AUTOSAR

- Cars are complex, distributed, real-time systems
- AUTomotive Open System ARchitecture
- *Runnables* defined by programmer
- Deployed to multiple *Electronic Control Units*
- Covers Requirements, Software/Hardwarearchitecture, Correctness Specification, . . .



Timing Constraints

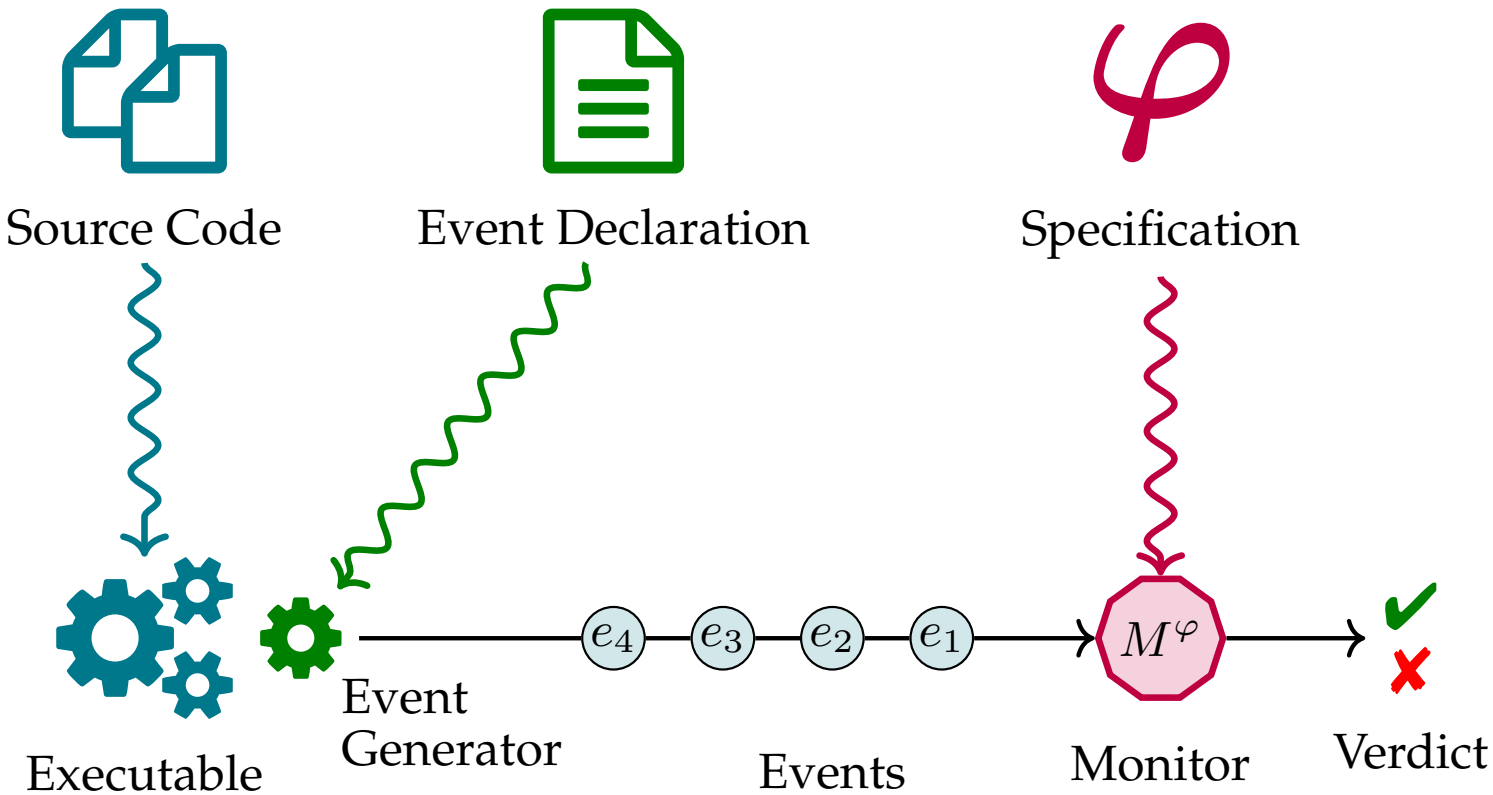
- Timing particularly hard to check
- Multiple Runnables on one ECU/Core (schedulability)

- Processing chains, e.g. break pedal to breaks
- AUTOSAR Timex constraints: informal specification
- TADL: formalization motivated by AUTOSAR

- Missing tool support to check/monitor timing constraints
- Current approach: handwritten monitors
- Our appoach: DSL based on Runtime Verification Language TeSSla

- **Declarative** style: Specification rather than implementation
- Abstractions for both **events** and **signals**
- Useful for description of **Cyber Physical Systems**
- **Modularity**: Allowing abstractions based on **few primitives**
- **Time** as first-class citizen
- **Recursion** to reason about past
- Implementable with **limited memory**

Runtime Verification with TeSSLa

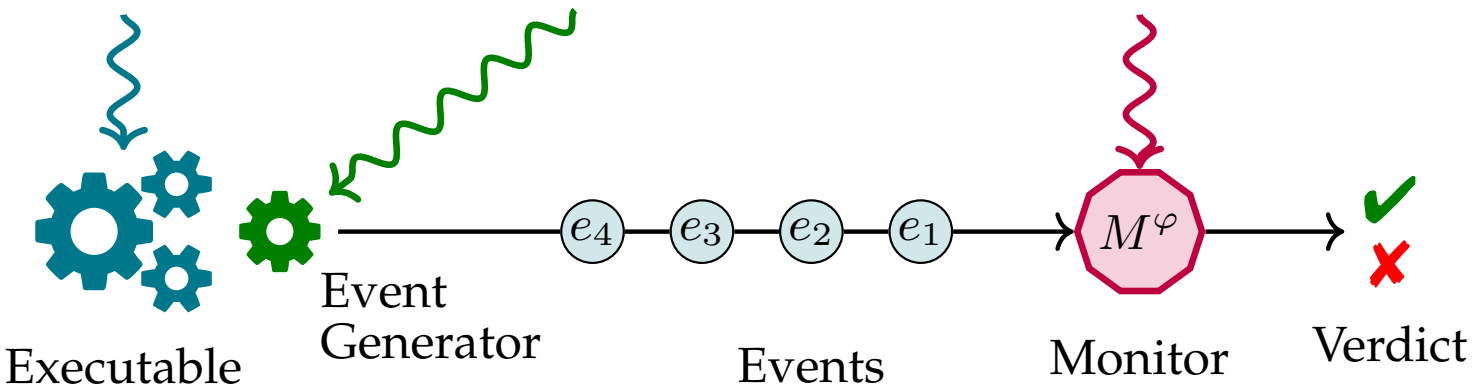


Runtime Verification with TeSSLa

```
int main() {
  while (1) {
    lock();
    critical();
    unlock();
  }
}
```

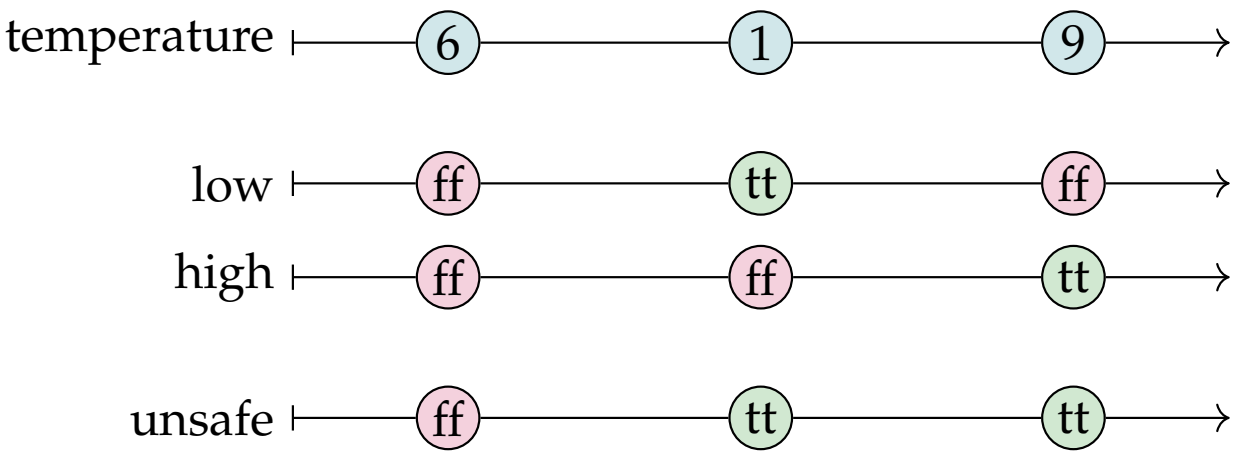
```
@InstFunctionCall("lock")
    in lock: Events[Unit]
@InstFunctionCall("unlock")
    in unlock: Events[Unit]
@InstFunctionCall("critical")
    in crit: Events[Unit]

    out on(crit, count(lock) −
                 count(unlock) == 1)
    as verdict
```



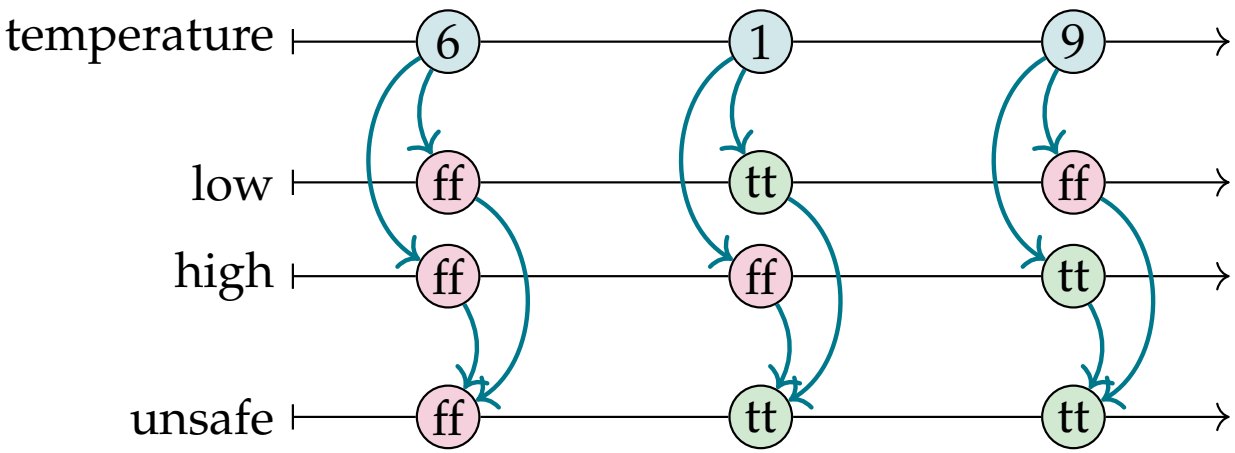
SRV: Combining streams

Correctness property: Temperature is between 2 and 8.



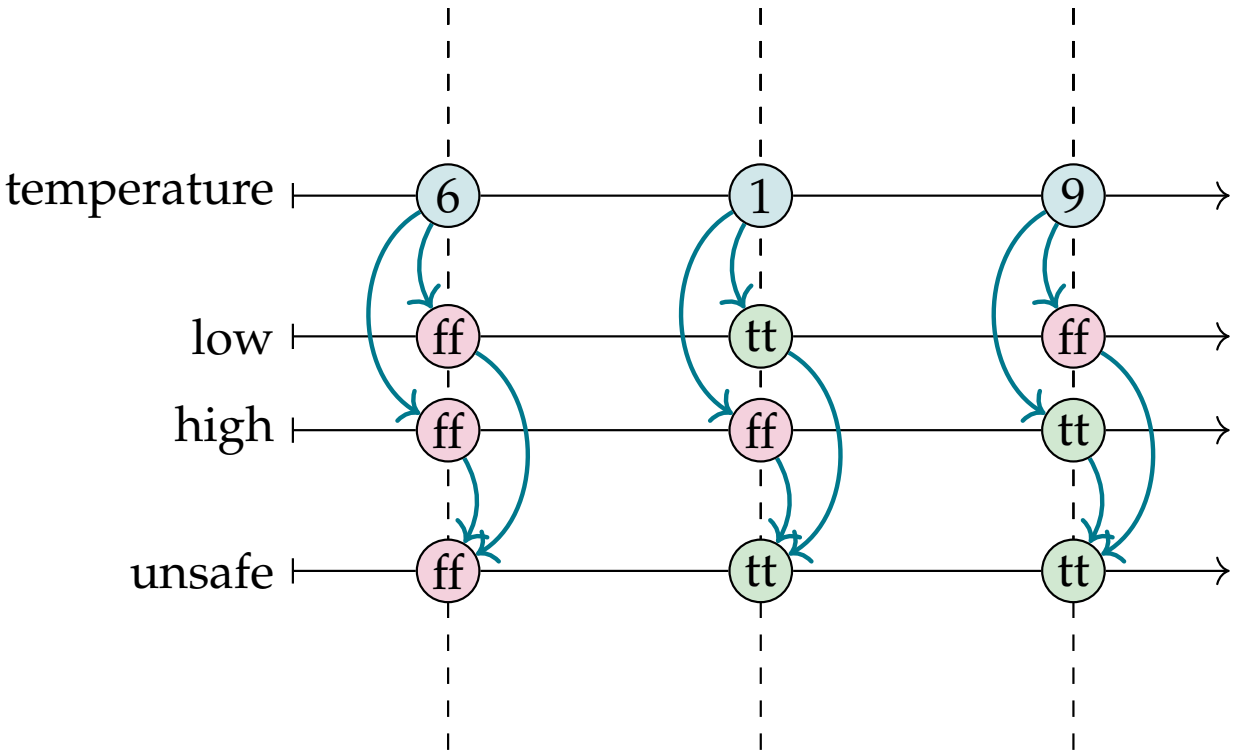
SRV: Combining streams

Correctness property: Temperature is between 2 and 8.



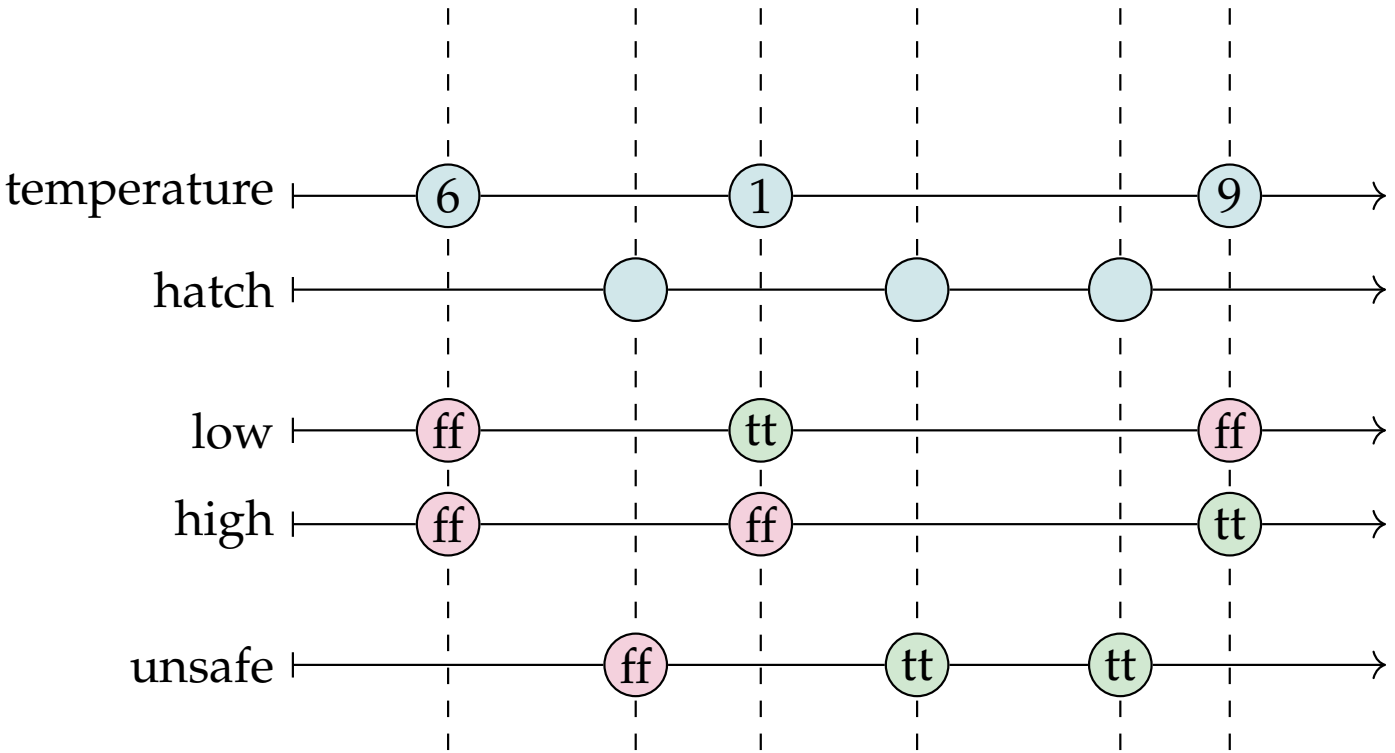
SRV: Synchronous streams

Correctness property: Temperature is between 2 and 8.



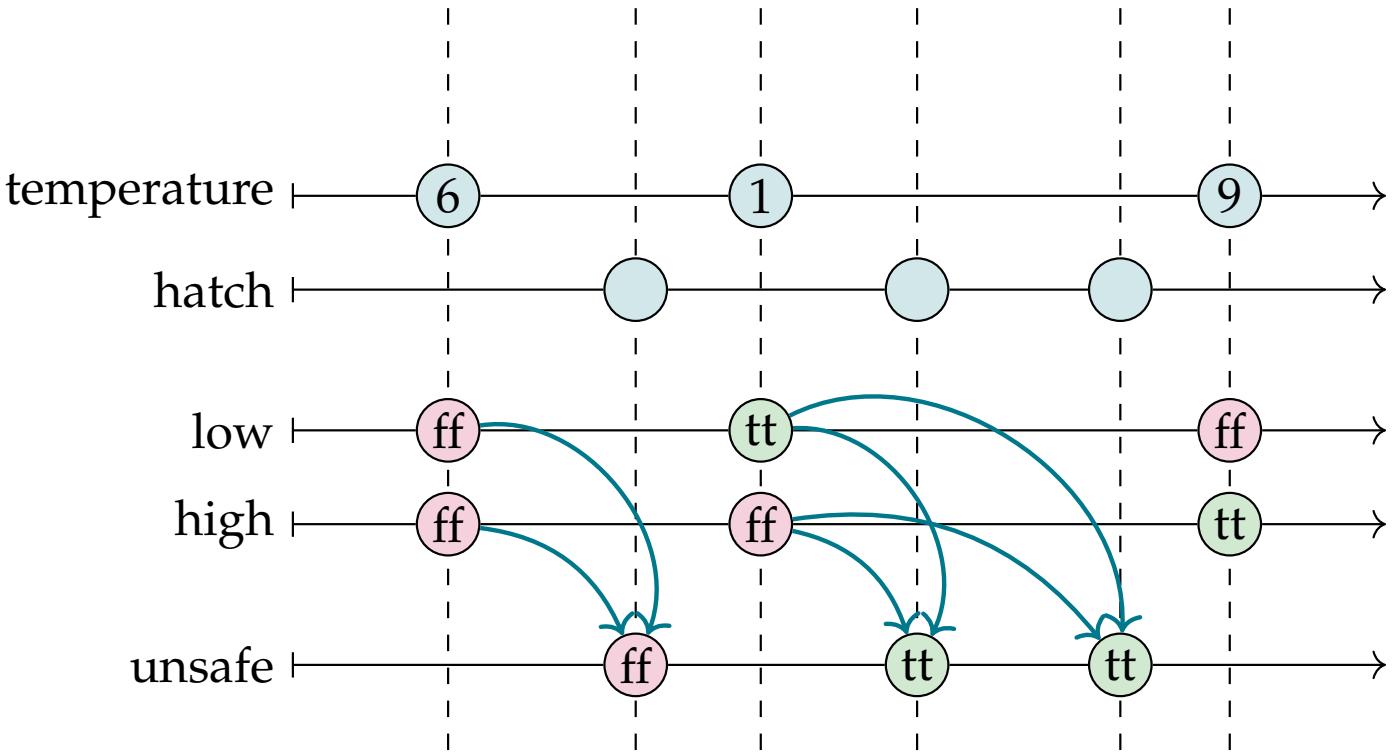
SRV: Synchronous streams

Correctness property: Temperature is between 2 and 8, when hatch is opened.



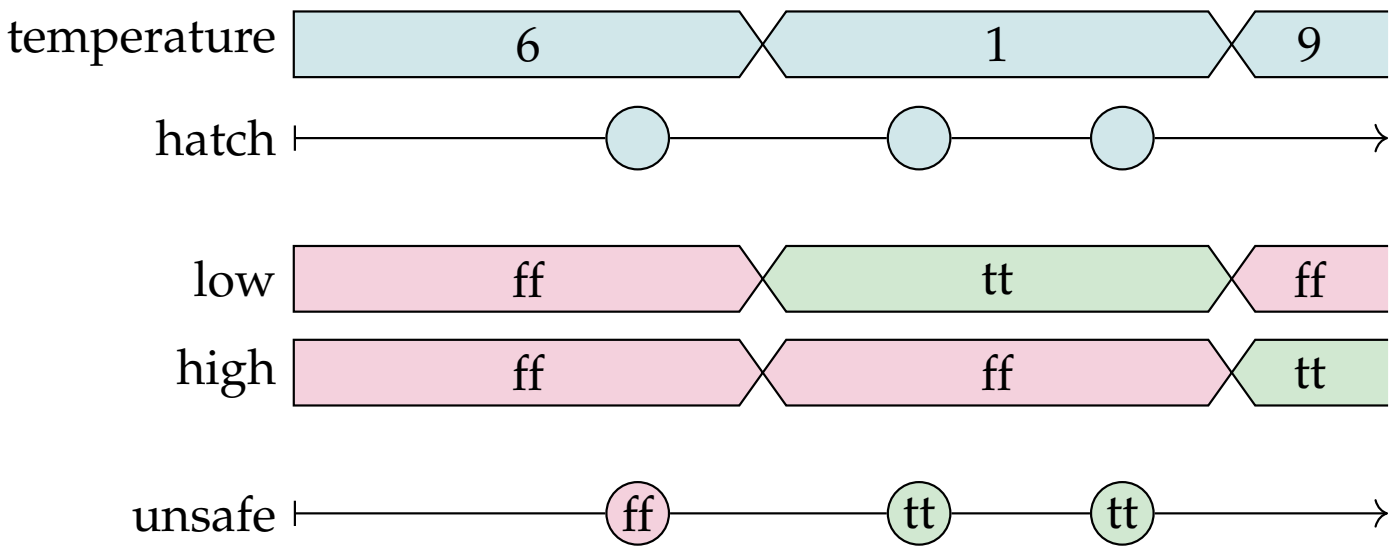
SRV: Synchronous streams

Correctness property: Temperature is between 2 and 8, when hatch is opened.



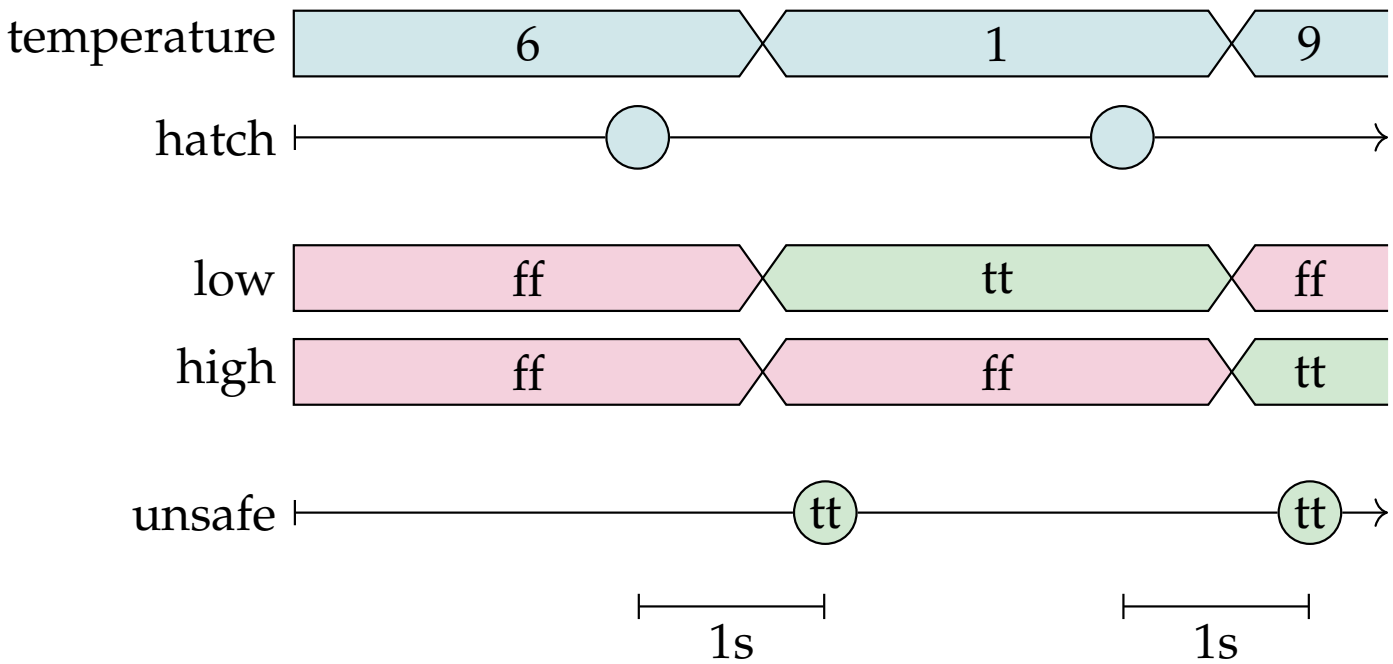
SRV: Signal semantics

Correctness property: Temperature is between 2 and 8, when hatch is opened.

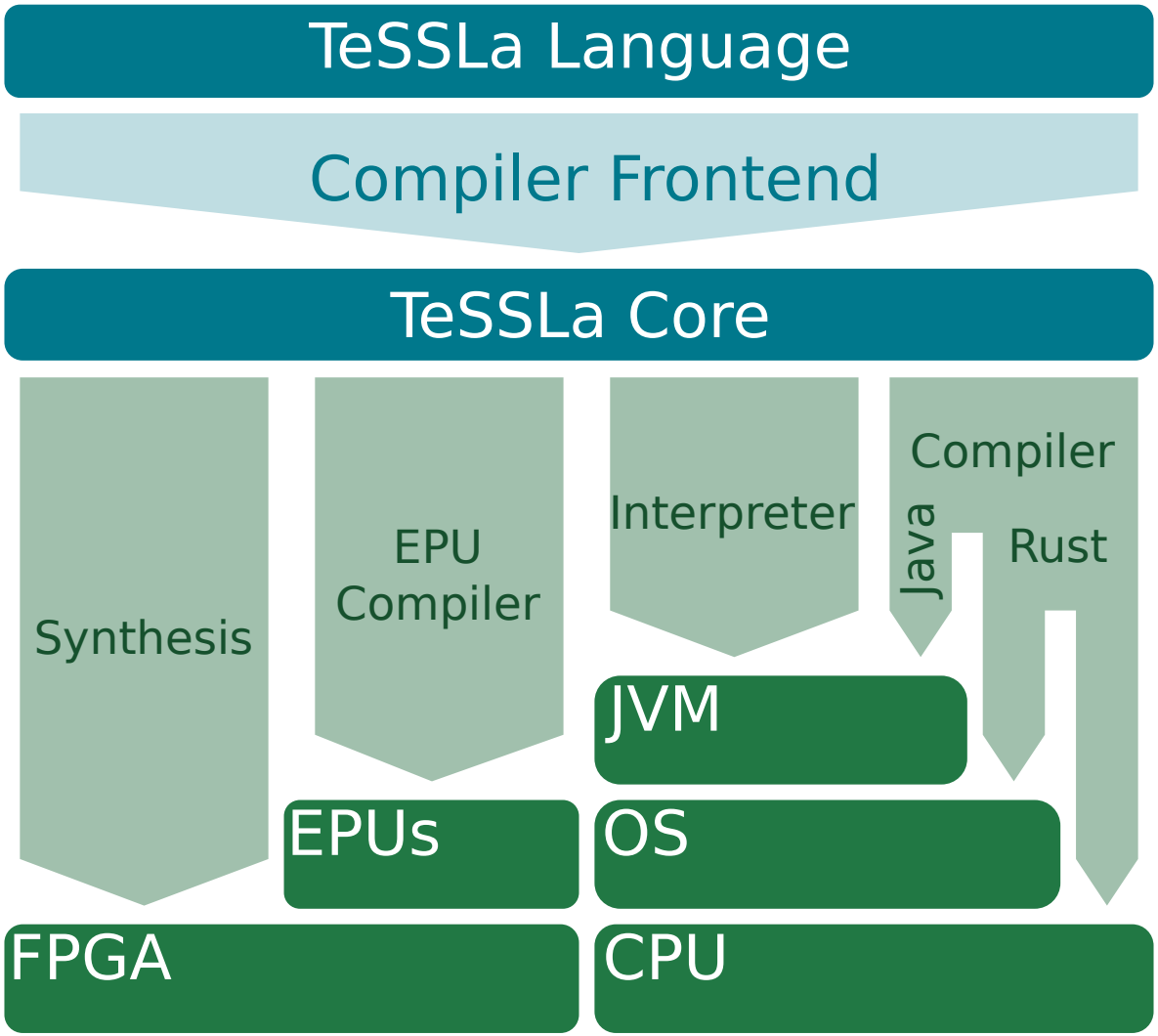


SRV: Asynchronous streams

Correctness property: Temperature is between 2 and 8, one second after hatch is opened.

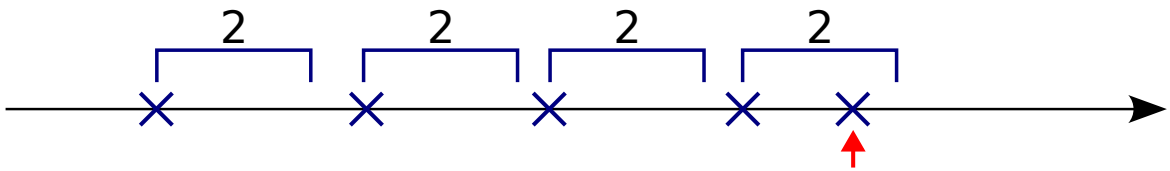


Evaluation Approaches for TeSSLa

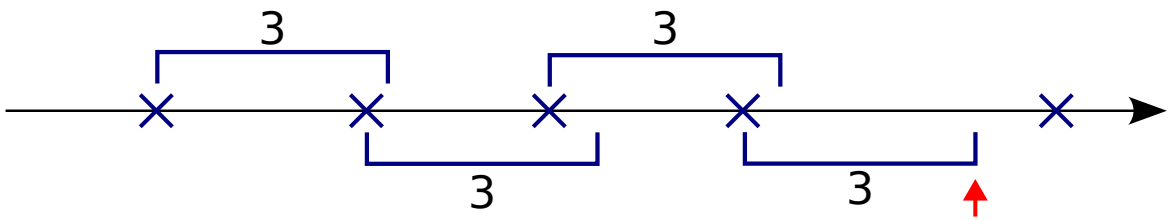


Classification of AUTOSAR/TADL Constraints

Minimal Distance (simple)



Maximal Distance (with delay)



Delay (unbounded memory)

Classification of AUTOSAR/TADL Constraints

Definition (Simple Monitor)

A *simple monitor* is given by a tuple $(Q, q_0 \in Q, n \in \mathbb{N}, \delta, m, o)$ where

- Q is a finite set of states,
- q_0 is the initial state,
- n is the number of timestamps storable in memory,
- $\delta : Q \times \mathbb{R}_0^+ \times D \times (\mathbb{R}_0^+)^n \to Q$ is the transition function,
- $o : Q \times \mathbb{R}_0^+ \times D \times (\mathbb{R}_0^+)^n \to D$ is the output function, and
- $m : Q \times \mathbb{R}_0^+ \times D \times (\mathbb{R}_0^+)^n \to (\mathbb{R}_0^+)^n$ is the memory function.

- Simple monitors use *practically* finite memory.
- Can be extended by delay-mechanism.

Classification of AUTOSAR/TADL Constraints

Simply Monitorable	Simply Monitorable with Delay	Not Simply Monitorable
TADL2 BurstConstraint	TADL2 RepeatConstraint	TADL2 DelayConstraint
AUTOSAR AgeConstraint	TADL2 ExecutionTimeConstraint	TADL2 StrongDelayConstraint
AUTOSAR SynchronizationPointConstraint	TADL2 SporadicConstraint	TADL2 SynchronizationConstraint
AUTOSAR ExecutionOrderConstraint	TADL2 PeriodicConstraint	TADL2 StrongSynchronizationConstraint
	TADL2 PatternConstraint	TADL2 OrderConstraint
	TADL2 ArbitraryConstraint	TADL2 ReactionConstraint
	AUTOSAR PeriodicEventTriggering	TADL2 AgeConstraint
	AUTOSAR SporadicEventTriggering	TADL2 OutputSynchronizationConstraint
	AUTOSAR ConcretePatternEventTriggering	TADL2 InputSynchronizationConstraint
	AUTOSAR ExecutionTimeConstraint	AUTOSAR BurstPatternEventTriggering
	AUTOSAR ConcretePatternEventTriggering	AUTOSAR ArbitraryEventTriggering
		AUTOSAR LatencyConstraint
		AUTOSAR OffsetTimingConstraint
		AUTOSAR SynchronizationConstraint

Implementation in TeSSLa

periodicConstraint

`periodicConstraint[A](events: Events[A], period: Int, jitter: Int, minDist: Int)`

Checks the PeriodicConstraint defined in TADL2.

The events occur in a periodic pattern with an allowed deviation of upper and a minimal distance of minDist.

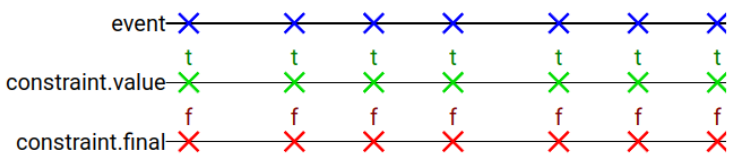
Usage Example

```
in event: Events[Unit]

def period = 10
def jitter = 2
def minDist = 9

def constraint= TADL2.periodicConstraint(event, period, jitter, minDist)
out constraint.value
out constraint.final
```

Trace Example



Source

```
def periodicConstraint[A](events: Events[A], period: Int, jitter: Int, minDist: Int) :=
    sporadicConstraint(events, period, period, jitter, minDist)
```

Experimental Results

- Synthetic traces if 10,000 events
- Constraints with varied parameters
- TeSSLa Interpreter: 0.1-1ms/Event (on standard PC)

- Usecase from OEM using LatencyTimingConstraints
- Latency in powertrain in various driving conditions
- Pre-recorded streams of 10-50 million Events
- 1-2μs/Event
- Recording time larger than proccessing time

Conclusion

- AUTOSAR: established architecture and modelling standard
- Timex/TADL2: set of practically relevant timing constraints
- TeSSLa: stream-based monitoring language with rich tooling
- Theoretical classification of constraints
- TeSSLA DSL for AUTOSAR Timex/TADL2
- Performance evaluation on pratical use case
- Future work: online monitoring on OEM data
- Open Source Implementation
- TeSSLa Website: `www.tessla.io`