# Symbolic Runtime Verification for Monitoring under Uncertainties and Assumptions

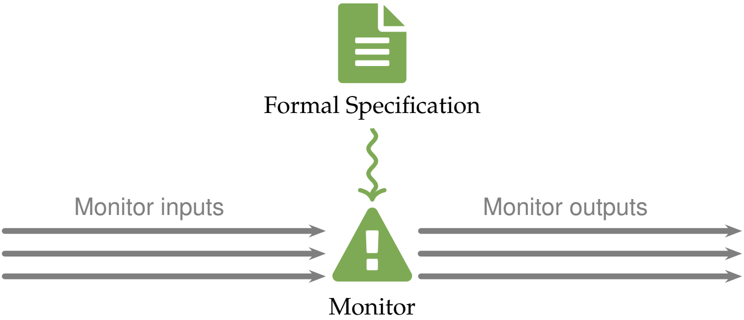
**Hannes Kallwies**[1] Martin Leucker[1] César Sánchez[2]

[1] Institute for Software Engineering and Programming Languages,
University of Lübeck, Lübeck, Germany

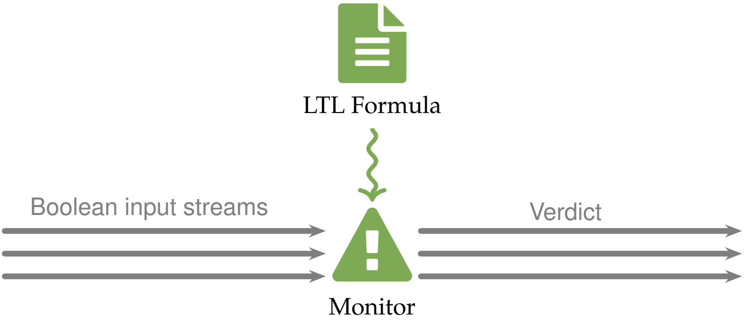
[2] IMDEA Software Institute, Madrid, Spain

International Symposium on Automated Technology for Verification and Analysis (ATVA), Oct. ’23
Adjusted Presentation from ATVA 2022

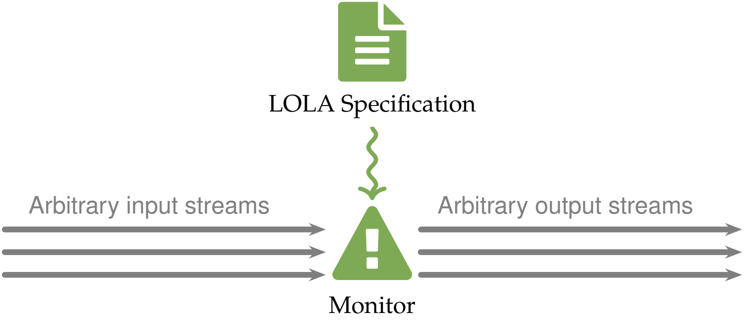
# General Setting: Runtime Verification



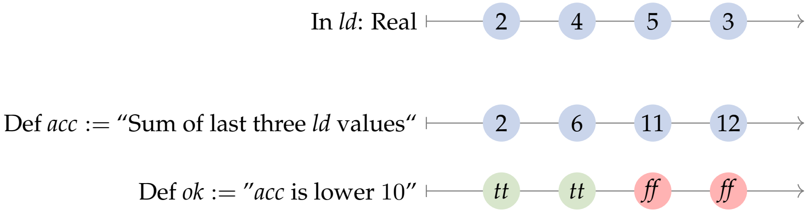
# General Setting: Runtime Verification



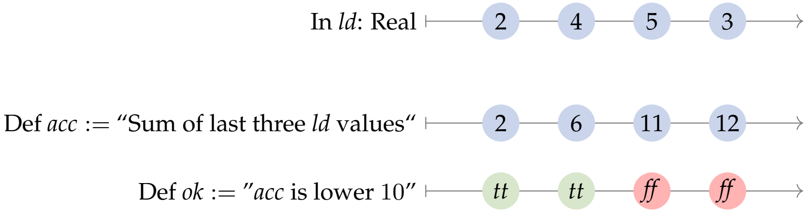
# General Setting: Stream Runtime Verification



# Stream Runtime Verification: Example



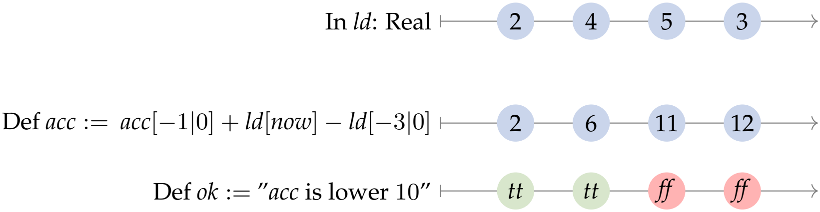
# Stream Runtime Verification: Example




Three basic LOLA stream expressions:

- Constant streams e.g. $10$
- Offset operators $s[o|c]$
  $\Rightarrow$: We restrict our self to the past fragment here (i.e. $o \leq 0$)
- Function applications e.g. $a[now] + b[now]$

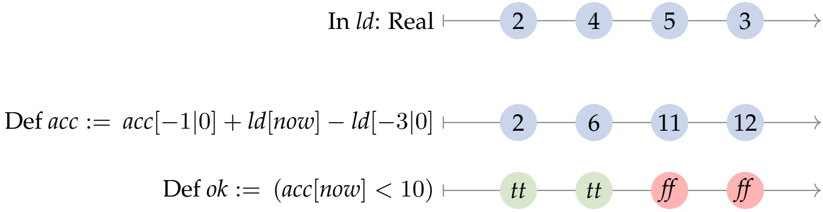
# Stream Runtime Verification: Example




Three basic LOLA stream expressions:

- Constant streams e.g. 10
- Offset operators $s[o|c]$
  $\Rightarrow$: We restrict our self to the past fragment here (i.e. $o \leq 0$)
- Function applications e.g. $a[now] + b[now]$

# Stream Runtime Verification: Example




Three basic LOLA stream expressions:

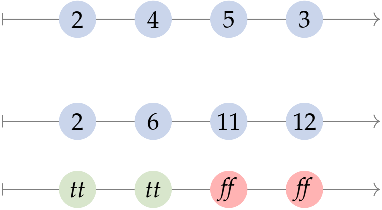
- Constant streams e.g. $10$
- Offset operators $s[o|c]$
  $\Rightarrow$: We restrict our self to the past fragment here (i.e. $o \leq 0$)
- Function applications e.g. $a[now] + b[now]$

# Research Question: Uncertainties & Assumptions

In *ld*: Real

Def $acc := acc[-1|0] + ld[now] - ld[-3|0]$

Def $ok := (acc[now] < 10)$

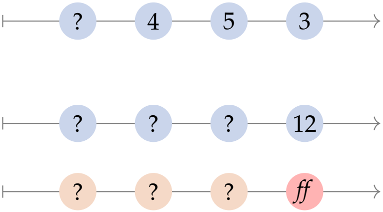


# Research Question: Uncertainties & Assumptions

In *ld*: Real

Def $acc := acc[-1|0] + ld[now] - ld[-3|0]$

Def $ok := (acc[now] < 10)$



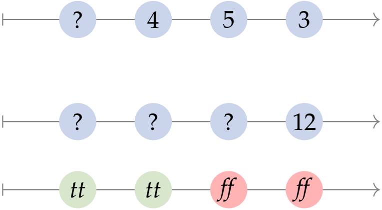

**Uncertainty:** What to do when some events are (partially) unknown?

# Research Question: Uncertainties & Assumptions

In *ld*: Real

Def $acc := acc[-1|0] + ld[now] - ld[-3|0]$

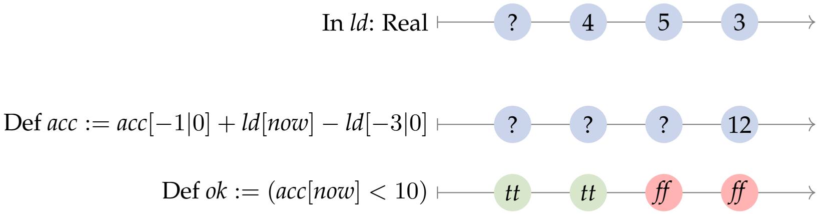
Def $ok := (acc[now] < 10)$




**Assumptions:** How to use additional information about the system?

E.g. The value of every input is between 1 and 5

# Research Question: Uncertainties & Assumptions




**Assumptions:** How to use additional information about the system?

E.g. The value of every input is between 1 and 5

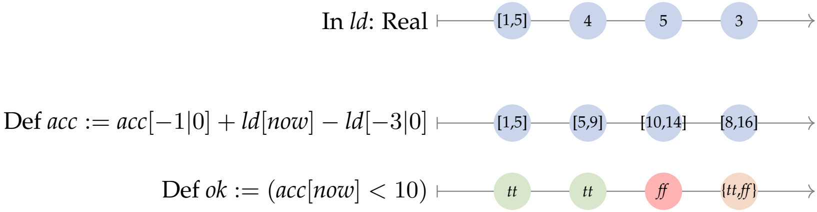
$\Rightarrow$ In general very powerful!

# Previous approach: Interval arithmetic

2019 Leucker et al. presented approach with intervals as abstract domain:

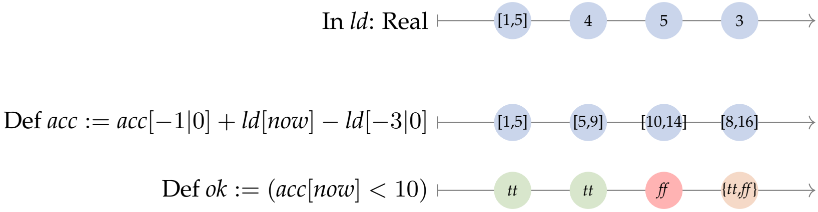
# Previous approach: Interval arithmetic

2019 Leucker et al. presented approach with intervals as abstract domain:



# Previous approach: Interval arithmetic

2019 Leucker et al. presented approach with intervals as abstract domain:




- Approach is sound, but not perfect.

# Previous approach: Interval arithmetic

2019 Leucker et al. presented approach with intervals as abstract domain:

| | | | | |
|---|---|---|---|---|
| In *ld*: Real | [1,5] | 4 | 5 | 3 |
| Def $acc := acc[-1\|0] + ld[now] - ld[-3\|0]$ | [1,5] | [5,9] | [10,14] | [8,16] |
| Def $ok := (acc[now] < 10)$ | *tt* | *tt* | *ff* | {*tt*,*ff*} |

- Approach is sound, but not perfect.
- Handling of complex assumptions not clear.

# New approach: Symbolic evaluation

# New approach: Symbolic evaluation

**Idea:** Use symbolic formulas for representation of unknown values and additional logical constraints (e.g. assumptions).

# New approach: Symbolic evaluation

**Idea:** Use symbolic formulas for representation of unknown values and additional logical constraints (e.g. assumptions).

$\Rightarrow$ Use SMT solver for queries on possible values.

# New approach: Symbolic evaluation

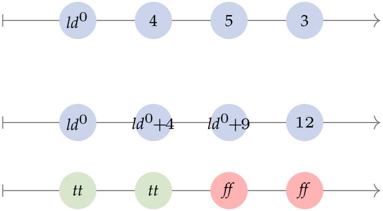
**Idea:** Use symbolic formulas for representation of unknown values and additional logical constraints (e.g. assumptions).

$\Rightarrow$ Use SMT solver for queries on possible values.

In $ld$: Real

Def $acc := acc[-1|0] + ld[now] - ld[-3|0]$

Def $ok := (acc[now] < 10)$




Additional constraints: $\{1 \leq ld^0 \leq 5\}$

# New approach: Symbolic evaluation

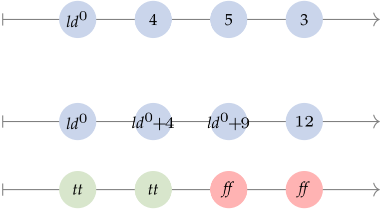
**Idea:** Use symbolic formulas for representation of unknown values and additional logical constraints (e.g. assumptions).

$\Rightarrow$ Use SMT solver for queries on possible values.

In $ld$: Real

Def $acc := acc[-1|0] + ld[now] - ld[-3|0]$

Def $ok := (acc[now] < 10)$




Additional constraints: $\{1 \leq ld^0 \leq 5\}$

- Approach in principle perfect.

# New approach: Symbolic evaluation

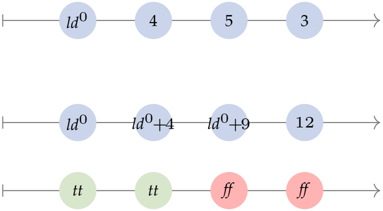
**Idea:** Use symbolic formulas for representation of unknown values and additional logical constraints (e.g. assumptions).

$\Rightarrow$ Use SMT solver for queries on possible values.

In *ld*: Real

Def $acc := acc[-1|0] + ld[now] - ld[-3|0]$

Def $ok := (acc[now] < 10)$




Additional constraints: $\{1 \leq ld^0 \leq 5\}$

- Approach in principle perfect.
- Assumptions up to $t$ can be added as propositions to constraint set.

# Symbolic SRV: Monitoring Algorithm

**Algorithm:** Symbolic Monitoring Algorithm for LOLA specification $\varphi$

$t \leftarrow 0$ and $E \leftarrow \emptyset$;
**while** $t \in \mathbb{T}$ **do**
    Read Input$^t$;
    $E \leftarrow E \cup [\![\varphi]\!]^t_{sym}$ ;
    $E \leftarrow E \cup \text{Input}^t$;
    Evaluate and Simplify;
    Output;
    $t \leftarrow t + 1$ ;

# Symbolic SRV: Monitoring Algorithm

**Algorithm:** Symbolic Monitoring Algorithm for LOLA specification $\varphi$

$t \leftarrow 0$ and $E \leftarrow \emptyset$;
**while** $t \in \mathbb{T}$ **do**
  Read Input$^t$;
  $E \leftarrow E \cup [\![\varphi]\!]^t_{sym}$ ;
  $E \leftarrow E \cup$ Input$^t$;
  Evaluate and Simplify;
  Output;
  $t \leftarrow t + 1$ ;

# Symbolic SRV: Monitoring Algorithm

**Algorithm:** Symbolic Monitoring Algorithm for LOLA specification $\varphi$

$t \leftarrow 0$ and $E \leftarrow \emptyset$;
**while** $t \in \mathbb{T}$ **do**
  Read Input$^t$;
  $E \leftarrow E \cup [\![\varphi]\!]^t_{sym} \cup [\![A^t]\!]_\varphi$;
  $E \leftarrow E \cup$ Input$^t$;
  Evaluate and Simplify;
  Output;
  $t \leftarrow t + 1$ ;

Assumption knowledge up to timestamp $t$ can also be included.

# Symbolic SRV: Monitoring Algorithm

**Algorithm:** Symbolic Monitoring Algorithm for LOLA specification $\varphi$

$t \leftarrow 0$ and $E \leftarrow \emptyset$;
**while** $t \in \mathbb{T}$ **do**
    Read Input$^t$;
    $E \leftarrow E \cup [\![\varphi]\!]^t_{sym} \cup [\![A^t]\!]_\varphi$;
    $E \leftarrow E \cup$ Input$^t$;
    Evaluate and Simplify;
    Output;
    $t \leftarrow t + 1$ ;

**Major problem: Symbolic formulas may grow unboundedly.**
$\Rightarrow$ **No monitoring with trace-length-independent resources!**

# Symbolic SRV: Monitoring Algorithm

**Algorithm:** Symbolic Monitoring Algorithm for LOLA specification $\varphi$

$t \leftarrow 0$ and $E \leftarrow \emptyset$;
**while** $t \in \mathbb{T}$ **do**
  Read $\text{Input}^t$;
  $E \leftarrow E \cup [\![\varphi]\!]^t_{sym} \cup [\![A^t]\!]_\varphi$;
  $E \leftarrow E \cup \text{Input}^t$;
  Evaluate and Simplify;
  Output;
  $t \leftarrow t + 1$ ;

**Major problem: Symbolic formulas may grow unboundedly.**
**⇒ No monitoring with trace-length-independent resources!**

⇒ Pruning of formulas necessary.

# Symbolic SRV: Monitoring Algorithm

**Algorithm:** Symbolic Monitoring Algorithm for LOLA specification $\varphi$

$t \leftarrow 0$ and $E \leftarrow \emptyset$;
**while** $t \in \mathbb{T}$ **do**
    Read Input$^t$;
    $E \leftarrow E \cup [\![\varphi]\!]^t_{sym} \cup [\![A^t]\!]_\varphi$;
    $E \leftarrow E \cup$ Input$^t$;
    Evaluate and Simplify;
    Output;
    Prune;
    $t \leftarrow t + 1$ ;

**Major problem: Symbolic formulas may grow unboundedly.**
$\Rightarrow$ **No monitoring with trace-length-independent resources!**

$\Rightarrow$ Pruning of formulas necessary.

# Pruning of symbolic formulas

**First case: Boolean LOLA Fragment**
(Only Boolean streams, operators and assumptions)

# Pruning of symbolic formulas

**First case: Boolean LOLA Fragment**
(Only Boolean streams, operators and assumptions)

$$\begin{array}{rcl} a & := & a[-1|\mathit{ff}] \oplus x[\mathit{now}] \\ b & := & b[-1|\mathit{tt}] \oplus x[\mathit{now}] \\ \mathit{ok} & := & a[\mathit{now}] \oplus b[\mathit{now}] \quad // = \mathit{true} \end{array}$$

| $t$ | 0 | 1 | 2 | 3 | … |
|---|---|---|---|---|---|
| $a^t$ | $x^0$ | $x^0 \oplus x^1$ | $x^0 \oplus x^1 \oplus x^2$ | $x^0 \oplus x^1 \oplus x^2 \oplus x^3$ | … |
| $b^t$ | $\neg x^0$ | $\neg x^0 \oplus x^1$ | $\neg x^0 \oplus x^1 \oplus x^2$ | $\neg x^0 \oplus x^1 \oplus x^2 \oplus x^3$ | … |
| $\mathit{ok}^t$ | $\mathit{tt}$ | $\mathit{tt}$ | $\mathit{tt}$ | $\mathit{tt}$ | … |

# Pruning of symbolic formulas

**First case: Boolean LOLA Fragment**
(Only Boolean streams, operators and assumptions)

$$\begin{aligned} a &:= a[-1|ff] \oplus x[now] \\ b &:= b[-1|tt] \oplus x[now] \\ ok &:= a[now] \oplus b[now] \quad // = true \end{aligned}$$

| $t$ | 0 | 1 | 2 | 3 | ... |
|---|---|---|---|---|---|
| $a^t$ | $x^0$ | $x^0 \oplus x^1$ | $x^0 \oplus x^1 \oplus x^2$ | $x^0 \oplus x^1 \oplus x^2 \oplus x^3$ | ... |
| $b^t$ | $\neg x^0$ | $\neg x^0 \oplus x^1$ | $\neg x^0 \oplus x^1 \oplus x^2$ | $\neg x^0 \oplus x^1 \oplus x^2 \oplus x^3$ | ... |
| $ok^t$ | $tt$ | $tt$ | $tt$ | $tt$ | ... |

**Observation:** Growing formulas in steps 1, 2, 3 describe only two possible vectors: $(a^3, b^3, ok^3) \in \{(ff, tt, tt), (tt, ff, tt)\}$.

# Pruning of symbolic formulas

**First case: Boolean LOLA Fragment**
(Only Boolean streams, operators and assumptions)

$$\begin{aligned} a &:= a[-1|\mathit{ff}] \oplus x[now] \\ b &:= b[-1|tt] \oplus x[now] \\ ok &:= a[now] \oplus b[now] \quad // = true \end{aligned}$$

| $t$ | 0 | 1 | 2 | 3 | … |
|---|---|---|---|---|---|
| $a^t$ | $x^0$ | $v^1$ | $v^1 \oplus x^2$ | $v^1 \oplus x^2 \oplus x^3$ | … |
| $b^t$ | $\neg x^0$ | $\neg v^1$ | $\neg v^1 \oplus x^2$ | $\neg v^1 \oplus x^2 \oplus x^3$ | … |
| $ok^t$ | $tt$ | $tt$ | $tt$ | $tt$ | … |

**Observation:** Growing formulas in steps 1, 2, 3 describe only two possible vectors: $(a^3, b^3, ok^3) \in \{(\mathit{ff}, tt, tt), (tt, \mathit{ff}, tt)\}$.

# Pruning of symbolic formulas

**First case: Boolean LOLA Fragment**
(Only Boolean streams, operators and assumptions)

$$\begin{array}{rcl} a & := & a[-1|\mathit{ff}] \oplus x[\mathit{now}] \\ b & := & b[-1|\mathit{tt}] \oplus x[\mathit{now}] \\ \mathit{ok} & := & a[\mathit{now}] \oplus b[\mathit{now}] \quad // = \mathit{true} \end{array}$$

| $t$ | 0 | 1 | 2 | 3 | ... |
|---|---|---|---|---|---|
| $a^t$ | $x^0$ | $v^1$ | $v^2$ | $v^3$ | ... |
| $b^t$ | $\neg x^0$ | $\neg v^1$ | $\neg v^2$ | $\neg v^3$ | ... |
| $ok^t$ | $tt$ | $tt$ | $tt$ | $tt$ | ... |

**Observation:** Growing formulas in steps 1, 2, 3 describe only two possible vectors: $(a^3, b^3, ok^3) \in \{(\mathit{ff}, tt, tt), (tt, \mathit{ff}, tt)\}$.

For the **Boolean fragment** trace-length independent symbolic monitoring is always possible!

# Pruning of symbolic formulas

**Second case: Linear Algebra LOLA Fragment**
(Only Real streams/assumptions of form
$s = c_1 \cdot s_1[o_1, d_1] + \cdots + c_n \cdot s_n[o_n, d_n]$)

# Pruning of symbolic formulas

**Second case: Linear Algebra LOLA Fragment**
(Only Real streams/assumptions of form
$s = c_1 \cdot s_1[o_1, d_1] + \cdots + c_n \cdot s_n[o_n, d_n]$)

$$\begin{aligned} acc_a &:= acc_a[-1|0] + ld_a[now] \\ acc_b &:= acc_b[-1|0] + ld_b[now] \\ total &:= total[-1|0] + \tfrac{1}{2}(ld_a[now] + ld_b[now]) \end{aligned}$$

| $t$ | 0 | 1 | 2 |
|---|---|---|---|
| $acc_a^t$ | $ld_a^0$ | $ld_a^0 + ld_a^1$ | $ld_a^0 + ld_a^1 + ld_a^2$ |
| $acc_b^t$ | $ld_b^0$ | $ld_b^0 + ld_b^1$ | $ld_b^0 + ld_b^1 + ld_b^2$ |
| $total^t$ | $\frac{1}{2}(ld_a^0 + ld_b^0)$ | $\frac{1}{2}(ld_a^0 + ld_b^0 + ld_a^1 + ld_b^1)$ | $\frac{1}{2}(ld_a^0 + ld_b^0 + ld_a^1 + ld_b^1 + ld_a^2 + ld_b^2)$ |

# Pruning of symbolic formulas

**Second case: Linear Algebra LOLA Fragment**
(Only Real streams/assumptions of form
$s = c_1 \cdot s_1[o_1, d_1] + \cdots + c_n \cdot s_n[o_n, d_n]$)

$$
\begin{array}{lcl}
acc_a & := & acc_a[-1|0] + ld_a[now] \\
acc_b & := & acc_b[-1|0] + ld_b[now] \\
total & := & total[-1|0] + \frac{1}{2}(ld_a[now] + ld_b[now])
\end{array}
$$

| $t$ | 0 | 1 | 2 |
|---|---|---|---|
| $acc_a^t$ | $ld_a^0$ | $ld_a^0 + ld_a^1$ | $ld_a^0 + ld_a^1 + ld_a^2$ |
| $acc_b^t$ | $ld_b^0$ | $ld_b^0 + ld_b^1$ | $ld_b^0 + ld_b^1 + ld_b^2$ |
| $total^t$ | $\frac{1}{2}(ld_a^0 + ld_b^0)$ | $\frac{1}{2}(ld_a^0 + ld_b^0 + ld_a^1 + ld_b^1)$ | $\frac{1}{2}(ld_a^0 + ld_b^0 + ld_a^1 + ld_b^1 + ld_a^2 + ld_b^2)$ |

**Similar Observation:** Growing formulas in steps 1, 2, 3 all describe
$span\{(1, 0, \frac{1}{2}), (0, 1, \frac{1}{2})\}$.

# Pruning of symbolic formulas

**Second case: Linear Algebra LOLA Fragment**
(Only Real streams/assumptions of form
$s = c_1 \cdot s_1[o_1, d_1] + \cdots + c_n \cdot s_n[o_n, d_n]$)

$$\begin{array}{rcl} acc_a & := & acc_a[-1|0] + ld_a[now] \\ acc_b & := & acc_b[-1|0] + ld_b[now] \\ total & := & total[-1|0] + \frac{1}{2}(ld_a[now] + ld_b[now]) \end{array}$$

$$\begin{pmatrix} acc_a^1 \\ acc_b^1 \\ total^1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{pmatrix} * \begin{pmatrix} ld_a^0 \\ ld_b^0 \\ ld_a^1 \\ ld_b^1 \end{pmatrix}$$

**Similar Observation:** Growing formulas in steps 1, 2, 3 all describe $span\{(1, 0, \frac{1}{2}), (0, 1, \frac{1}{2})\}$.

# Pruning of symbolic formulas

**Second case: Linear Algebra LOLA Fragment**
(Only Real streams/assumptions of form
$s = c_1 \cdot s_1[o_1, d_1] + \cdots + c_n \cdot s_n[o_n, d_n]$)

$$\begin{array}{rcl} \mathit{acc}_a & := & \mathit{acc}_a[-1|0] + \mathit{ld}_a[\mathit{now}] \\ \mathit{acc}_b & := & \mathit{acc}_b[-1|0] + \mathit{ld}_b[\mathit{now}] \\ \mathit{total} & := & \mathit{total}[-1|0] + \frac{1}{2}(\mathit{ld}_a[\mathit{now}] + \mathit{ld}_b[\mathit{now}]) \end{array}$$

$$\begin{pmatrix} \mathit{acc}_a^1 \\ \mathit{acc}_b^1 \\ \mathit{total}^1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix} * \begin{pmatrix} u^1 \\ v^1 \end{pmatrix}$$

**Similar Observation:** Growing formulas in steps 1, 2, 3 all describe
$\mathit{span}\{(1, 0, \frac{1}{2}), (0, 1, \frac{1}{2})\}$.

# Pruning of symbolic formulas

**Second case: Linear Algebra LOLA Fragment**
(Only Real streams/assumptions of form
$s = c_1 \cdot s_1[o_1, d_1] + \cdots + c_n \cdot s_n[o_n, d_n]$)

$$\begin{aligned} acc_a &:= acc_a[-1|0] + ld_a[now] \\ acc_b &:= acc_b[-1|0] + ld_b[now] \\ total &:= total[-1|0] + \tfrac{1}{2}(ld_a[now] + ld_b[now]) \end{aligned}$$

| $t$ | 0 | 1 | 2 |
|---|---|---|---|
| $acc_a^t$ | $ld_a^0$ | $u^1$ | $u^2$ |
| $acc_b^t$ | $ld_b^0$ | $v^1$ | $v^2$ |
| $total^t$ | $\frac{1}{2}(ld_a^0 + ld_b^0)$ | $\frac{1}{2}(u^1 + v^1)$ | $\frac{1}{2}(u^2 + v^2)$ |

**Similar Observation:** Growing formulas in steps 1, 2, 3 all describe
$span\{(1, 0, \frac{1}{2}), (0, 1, \frac{1}{2})\}$.

For the **Linear Algebra fragment** trace-length independent symbolic monitoring is always possible!

# Pruning of symbolic formulas

**Third case: Linear Arithmetic LOLA Fragment**
(Combination of previous fragments with additional operators $==$ and $<$)

# Pruning of symbolic formulas

**Third case: Linear Arithmetic LOLA Fragment**
(Combination of previous fragments with additional operators $==$ and $<$)

**A perfect pruning is not possible here!**

$$\begin{array}{lcl} x & := & x[-1|0] + i[now] \\ y & := & 2 * y[-1|0] + i[now] \end{array}$$

$$\begin{pmatrix} x^2 \\ y^2 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 \\ 4 & 2 & 1 \end{pmatrix} * \begin{pmatrix} i^0 \\ i^1 \\ i^2 \end{pmatrix}$$

Assumption: $0 \leq i[now] \leq 1$

# Pruning of symbolic formulas

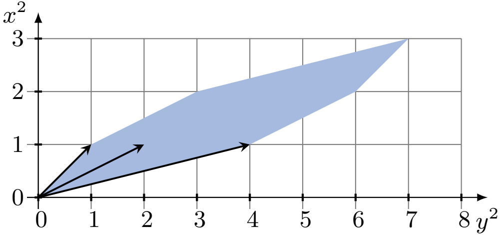
**Third case: Linear Arithmetic LOLA Fragment**
(Combination of previous fragments with additional operators == and <)

**A perfect pruning is not possible here!**

$$\begin{array}{lll} x & := & x[-1|0] + i[now] \\ y & := & 2 * y[-1|0] + i[now] \end{array}$$

$$\begin{pmatrix} x^2 \\ y^2 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 \\ 4 & 2 & 1 \end{pmatrix} * \begin{pmatrix} i^0 \\ i^1 \\ i^2 \end{pmatrix}$$

Assumption: $0 \leq i[now] \leq 1$



# Pruning of symbolic formulas

**Third case: Linear Arithmetic LOLA Fragment**
(Combination of previous fragments with additional operators $==$ and $<$)

**Sound, but imperfect (over-approximating) pruning strategy:**

- Perform pruning strategies for Boolean and Real streams separately.
- Calculate (over approximating) bounds for new Real variables (e.g. with Linear Arithmetic Optimizer).
- Add calculated bounds as constraints.

# Pruning of symbolic formulas

**Third case: Linear Arithmetic LOLA Fragment**
(Combination of previous fragments with additional operators $==$ and $<$)

**Sound, but imperfect (over-approximating) pruning strategy:**

- Perform pruning strategies for Boolean and Real streams separately.
- Calculate (over approximating) bounds for new Real variables (e.g. with Linear Arithmetic Optimizer).
- Add calculated bounds as constraints.

$\Rightarrow$ Leads to imperfect over-approximation of exact polyhedron.

For the **Linear Arithmetic fragment** trace-length independent symbolic monitoring is not possible!

For the **Linear Arithmetic fragment** trace-length independent symbolic monitoring is not possible!

But: Over-approximation possible

# Addition: Quantifier-Elimination

Quantifier elimination can be used to get a perfect pruning strategy.

# Addition: Quantifier-Elimination

Quantifier elimination can be used to get a perfect pruning strategy.

Given: Set of equations

$$\{s_1 = \varphi_1[i_1, \ldots, i_m], \ldots, s_n = \varphi_n[i_1, \ldots, i_m]\}$$

# Addition: Quantifier-Elimination

Quantifier elimination can be used to get a perfect pruning strategy.

Given: Set of equations

$$\{s_1 = \varphi_1[i_1, \ldots, i_m], \ldots, s_n = \varphi_n[i_1, \ldots, i_m]\}$$

Find quantifier-free formula

$$\psi \equiv \exists i_1, \ldots, i_m. \left( (s_1 = \varphi_1[i_1, \ldots, i_m]) \wedge \cdots \wedge (s_n = \varphi_n[i_1, \ldots, i_m]) \right)$$

# Addition: Quantifier-Elimination

Quantifier elimination can be used to get a perfect pruning strategy.

Given: Set of equations

$$\{s_1 = \varphi_1[i_1, \ldots, i_m], \ldots, s_n = \varphi_n[i_1, \ldots, i_m]\}$$

Find quantifier-free formula

$$\psi \equiv \exists i_1, \ldots, i_m. ((s_1 = \varphi_1[i_1, \ldots, i_m]) \wedge \cdots \wedge (s_n = \varphi_n[i_1, \ldots, i_m]))$$

**But:** QE is usually perfect, but not constant, and does not exist for every logic.

# Evaluation

Developed proof-of-concept implementation in Scala with z3 as SMT solver and optimizer.

# Evaluation

- Developed proof-of-concept implementation in Scala with z3 as SMT solver and optimizer.
- Evaluated concept on two case-studies from previous publications:

# Evaluation

Developed proof-of-concept implementation in Scala with z3 as SMT solver and optimizer.

Evaluated concept on two case-studies from previous publications:

- **Emission Example:** LOLA specification monitoring a car test drive and checking for NOx emission and valid test ride. *($\sim x$ offset/function applications)*

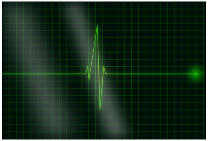
# Evaluation

Developed proof-of-concept implementation in Scala with z3 as SMT solver and optimizer.

Evaluated concept on two case-studies from previous publications:

- **Emission Example:** LOLA specification monitoring a car test drive and checking for NOx emission and valid test ride. *($\sim x$ offset/function applications)*

- **Heart Rate Example:** LOLA specification which detects peaks in an ECG signal *($\sim x$ offset/function applications)*.



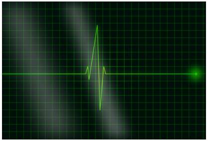
# Evaluation

Developed proof-of-concept implementation in Scala with z3 as SMT solver and optimizer.

Evaluated concept on two case-studies from previous publications:

- **Emission Example:** LOLA specification monitoring a car test drive and checking for NOx emission and valid test ride. *($\sim x$ offset/function applications)*

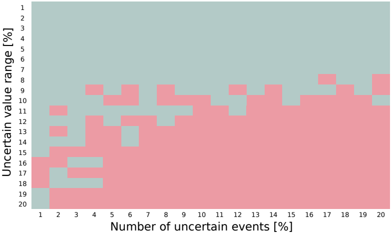
- **Heart Rate Example:** LOLA specification which detects peaks in an ECG signal *($\sim x$ offset/function applications)*.



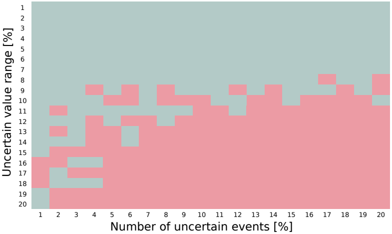
⇒ Introduced different kinds of uncertainty into the traces and added assumptions.

Compared to interval approach from previous publication.

# Evaluation: Emission Example



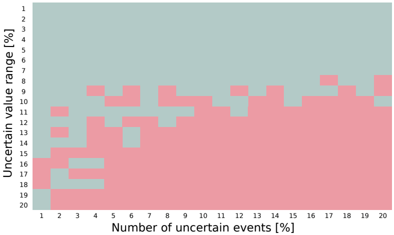
# Evaluation: Emission Example




**Observations:**

- If interval around correct value is known symbolic approach is still able to give certain results (figure; green part).

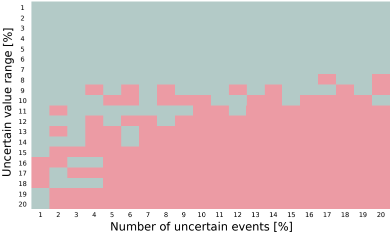
# Evaluation: Emission Example




**Observations:**

- If interval around correct value is known symbolic approach is still able to give certain results (figure; green part).
- More precise intermediate results than interval approach, but no difference in final results.

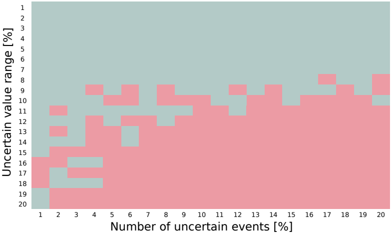
# Evaluation: Emission Example




**Observations:**

- If interval around correct value is known symbolic approach is still able to give certain results (figure; green part).
- More precise intermediate results than interval approach, but no difference in final results.
- If inputs are fully unknown no certain results.

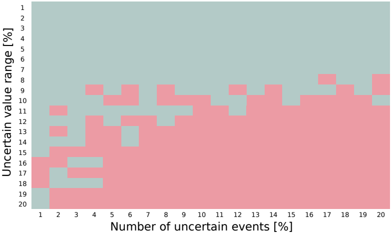
# Evaluation: Emission Example




**Observations:**

- If interval around correct value is known symbolic approach is still able to give certain results (figure; green part).
- More precise intermediate results than interval approach, but no difference in final results.
- If inputs are fully unknown no certain results.
- With additional assumptions (e.g. limited acceleration): Certain results up to 4% of fully uncertain inputs.

# Evaluation: Emission Example




**Observations:**

- If interval around correct value is known symbolic approach is still able to give certain results (figure; green part).
- More precise intermediate results than interval approach, but no difference in final results.
- If inputs are fully unknown no certain results.
- With additional assumptions (e.g. limited acceleration): Certain results up to 4% of fully uncertain inputs.
- Interval approach does not support assumptions.

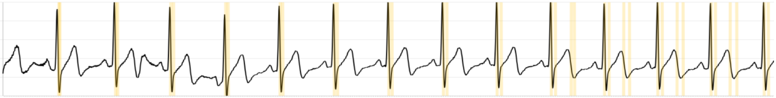
# Evaluation: Heart Rate Example

**Certain percentage of values uncertain (interval of 20% around real value)**
**[ peak detected; uncertain detection]**

# Evaluation: Heart Rate Example

**Certain percentage of values uncertain (interval of 20% around real value)**
**[ peak detected; uncertain detection]**

Interval approach [5% uncertain values]:

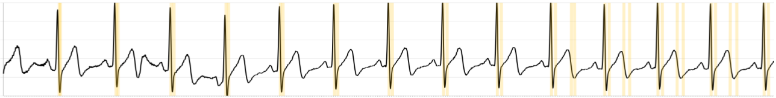


⇒ Due to interval arithmetic uncertainty gets accumulated forever.

# Evaluation: Heart Rate Example

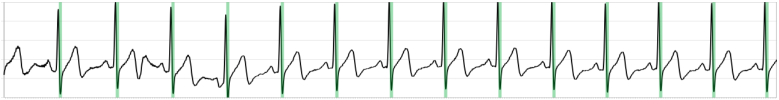
**Certain percentage of values uncertain (interval of 20% around real value)**
**[■ peak detected; ■ uncertain detection]**

Interval approach [5% uncertain values]:



⇒ Due to interval arithmetic uncertainty gets accumulated forever.

Symbolic approach with assumption (two heart peaks cannot be too close) [20% uncertain values]:



# Evaluation: Heart Rate Example

**Bursts: 5 to 20 fully uncertain events in a row**
**[ burst; peak detected; uncertain detection]**

# Evaluation: Heart Rate Example

**Bursts: 5 to 20 fully uncertain events in a row**
**[■ burst; ■ peak detected; ■ uncertain detection]**

Interval approach:

⇒ Again due to interval arithmetic total uncertainty is accumulated forever.

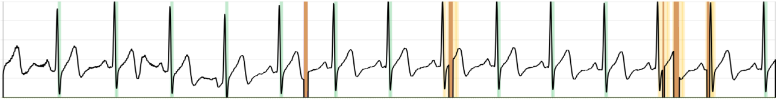
# Evaluation: Heart Rate Example

**Bursts: 5 to 20 fully uncertain events in a row**
**[ burst; peak detected; uncertain detection]**

Interval approach:

⇒ Again due to interval arithmetic total uncertainty is accumulated forever.

Symbolic approach with assumption (two heart peaks cannot be too close):



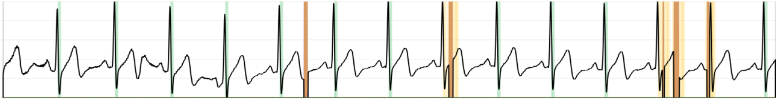
# Evaluation: Heart Rate Example

**Bursts: 5 to 20 fully uncertain events in a row**
**[ burst; peak detected; uncertain detection]**

Interval approach:

⇒ Again due to interval arithmetic total uncertainty is accumulated forever.

Symbolic approach with assumption (two heart peaks cannot be too close):



**Drawback: Symbolic approach (with assumptions) has sig. increased runtime: 25-100 ms/event**

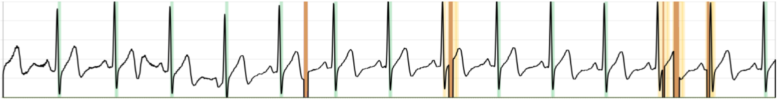
# Evaluation: Heart Rate Example

**Bursts: 5 to 20 fully uncertain events in a row**
**[ burst; peak detected; uncertain detection]**

Interval approach:

⇒ Again due to interval arithmetic total uncertainty is accumulated forever.

Symbolic approach with assumption (two heart peaks cannot be too close):



**Research question:** How to use solver more efficiently for regular problem

# Conclusion & Future Work

**Conclusion**

- **Symbolic evaluation** is a powerful, generic approach for Runtime Verification under **Uncertainty** and **Assumptions**
- **Pruning strategies** allow trace-length independent monitoring
- For some fragments **perfect pruning strategies** exist, for others not
- Approach useful in practice, but significantly slower than interval-based approaches

# Conclusion & Future Work

**Conclusion**

- **Symbolic evaluation** is a powerful, generic approach for Runtime Verification under **Uncertainty** and **Assumptions**
- **Pruning strategies** allow trace-length independent monitoring
- For some fragments **perfect pruning strategies** exist, for others not
- Approach useful in practice, but significantly slower than interval-based approaches

**Future Work**

- Support future offsets (and future assumptions)
- Investigate further LOLA fragments
- Improve implementation