ARV Overview

• **New approach** to runtime verification in which **overhead control**, **state estimation**, and **predictive analysis** are synergistically combined;

• Every monitor instance has an associated **criticality level**, which is a measure of how ``close'' the instance is to violating property underinvestigation;

• As **criticality levels** of monitor instances **rise**, so will fraction of monitoring resources allocated to these instances, **thereby increasing probability of violation detection**;

• There is a **delicate interplay** between (software) **state estimation** and (monitoring) **overhead control**.
Outline

- Adaptive Runtime Verification Framework
- Runtime Verification with State Estimation
- Precomputing RVSE
- Predictive Analysis of Criticality Level
- Case Study
- Results
- Conclusions
Adaptive Runtime Verification Framework

Monitored System

InterAspect

Instrumentation


InterAspect: Aspect-Oriented Instrumentation with GCC. Formal Methods in System Design 2012
Adaptive Runtime Verification Framework

Monitored System

InterAspect

Software monitoring with controllable overhead

Monitoring Framework

Primary Controller

Secondary Controller | Overhead: Low
Monitor: On

Secondary Controller | Overhead: Low
Monitor: On

Secondary Controller | Overhead: High
Monitor: Off

Adaptive Runtime Verification Framework

Monitored System

InterAspect

Software monitoring with controllable overhead

Monitoring Framework

Primary Controller

Secondary Controller
Overhead: Low
Monitor: On

Secondary Controller
Overhead: Low
Monitor: On

Secondary Controller
Overhead: High
Monitor: Off

Real Trace
DISP CMD SUCC CMD FAIL

Monitored Trace
DISP CMD SUCC CMD

DFSM
Adaptive Runtime Verification Framework

Monitored System

InterAspect

Software monitoring with controllable overhead

Monitoring Framework

Primary Controller

Secondary Controller

Overhead: Low

Monitor: On

Secondary Controller

Overhead: Low

Monitor: On

Secondary Controller

Overhead: Low

Monitor: On

Real Trace

DISP CMD SUCC CMD

CMD FAIL SUCC

Monitored Trace

DISP CMD SUCC CMD

CMD FAIL SUCC

Monitor: On

Off

On

m1

CMD

m2

DISP

m3

FAIL

SUCC

DFSM

?
Adaptive Runtime Verification Framework

InterAspect

Monitored System

Runtime Verification with State Estimation (RVSE)

Monitoring Framework

Primary Controller

Secondary Controller
Overhead: Low
Monitor: On

Secondary Controller
Overhead: Low
Monitor: On

Secondary Controller
Overhead: Low
Monitor: On

Learn an HMM

Compose with the monitor

Estimate the Error Probability (EP)

Real Trace

DISP CMD SUCC CMD CMD SUCC CMD DISP DISP ........

Real Trace

DISP CMD SUCC DISP CMD FAIL SUCC

Monitored Trace

DISP CMD SUCC CMD CMD SUCC CMD DISP DISP ........

Scott D. Stoller, Ezio Bartocci, Justin Seyster, Radu Grosu, Klaus Havelund, Scott A. Smolka, Erez Zadok.

Adaptive Runtime Verification Framework

Monitored System

InterAspect

Primary Controller Monitoring Framework

Monitoring Framework

Secondary Controller

Overhead: Low

Monitor: On
EP: Med Crit: Safe

Secondary Controller

Overhead: High

Monitor: Off
EP: Low Crit: Safe

Secondary Controller

Overhead: High

Monitor: On
EP: Low Crit: Critical

EP \rightarrow \text{Error Probability}
Crit. \rightarrow \text{Criticality}

To give an intuition

Low Critical

High Critical

m_1 \rightarrow \text{DISP FAIL SUCC}
m_2 \rightarrow \text{CMD DISP FAIL SUCC}
m_3 \rightarrow \text{CMD DISP FAIL SUCC}
Runtime Verification with State Estimation

1) Learning (offline) an HMM from a set of traces (Baum and Welch)

\[ \pi = \begin{bmatrix} 0.9999578 & 3.9036e-12 & 4.2129e-05 \end{bmatrix} \]

\[ A = \begin{bmatrix} 5.7498e-07 & 0.999994 & 1.0838e-11 \\ 7.9406e-12 & 0.7999385 & 0.2000614 \\ 0.9999990 & 3.5693e-11 & 9.4621e-07 \end{bmatrix} \]

\[ B = \begin{bmatrix} 0.9987651 & 2.1899e-41 & 5.7327e-07 & 0.0012343 \\ 3.8173e-21 & 2.4628e-18 & 0.9996049 & 3.9504e-04 \\ 1.3882e-40 & 0.9990775 & 9.4410e-07 & 9.2147e-04 \end{bmatrix} \]

\( \pi \) is the initial state distribution

\( A \) is the transition probability distribution

\( B \) is the observation probability distribution
1) Learning (offline) an HMM from a set of traces (Baum and Welch)

2) Computing (offline) the gap distribution for a particular overhead

\[ L(\ell) = \]

![Graph showing gap distribution](image)
Runtime Verification with State Estimation

1) Learning (offline) an HMM from a set of traces (Baum and Welch)

2) Computing (offline) the gap distribution for a particular overhead

3) Computing (online) the forward algorithm for RVSE

\[ p_i(m,n) = \sum_{v \in V, s. l. \delta(m,n) = v} b_i(v) \quad 1 \leq j \leq N_s \text{ and } n \in S_M \]

\[ g_0(i,m,j,n) = (i = j \land m = n) ? 1 : 0 \]

\[ g_{t+1}(i,m,j,n) = \sum_{i' \in [1...N_s], m' \in S_M} g_t(i,m,i',m') A_{i',j} p_j(m',n) \]

\[ \alpha_{t+1}(j,n) = \begin{cases} 
L(0)(n = m_{init} ? \pi_j : 0) + \sum_{t \geq 0, i \in [1...N_s]} L(\ell) \pi_j g_t(i,m_{init},j,n) & \text{if } O_t = \text{gap}(L) \\
(n = \delta(m_{init}, O_t)) ? \pi_j b_j(O_t) : 0 & \text{if } O_t \neq \text{gap}(L) 
\end{cases} \text{ for } 1 \leq j \leq N_s \text{ and } n \in S_M \]

\[ \alpha_t(j,n) = \begin{cases} 
\sum_{i \in [1...N_s], m \in S_M} \alpha_t(i,m) A_{i,j} b_j(O_{t+1}) & \text{if } O_{t+1} \neq \text{gap}(L) \\
L(0) \alpha_t(j,n) + \sum_{t>0} L(\ell) \sum_{i \in [1...N_s], m \in S_M} \alpha_t(i,m) g_t(i,m,j,n) & \text{if } O_{t+1} = \text{gap}(L) 
\end{cases} \text{ for } 1 \leq t \leq T - 1 \text{ and } 1 \leq j \leq N_s \text{ and } n \in S_M \]
Runtime Verification with State Estimation

1) Learning (offline) an HMM from a set of traces (Baum and Welch)

2) Computing (offline) the gap distribution for a particular overhead

3) Computing (online) the forward algorithm for RVSE

Very expensive to compute !!!

\[
p_i(m,n) = \sum_{v \in V \text{ s.t. } \delta(m,n)=v} b_j(v) 1 \leq j \leq N_s \text{ and } n \in S_M
\]

\[
g_0(i, m, j, n) = (i = j \land m = n) ? 1 : 0
\]

\[
g_{t+1}(i, m, j, n) = \sum_{i' \in \{1 \ldots N_s\}, m' \in S_M} g_t(i, m, i', m') A_{i', i} p_j(m', n)
\]

\[
\alpha_i(j, n) = \begin{cases} L(0)(n = m_{\text{init}} \land \pi_j : 0) + \sum_{\ell > 0, \ell \leq N_s} L(\ell) \pi_j g_t(i, m_{\text{init}}, j, n) & \text{if } O_t = \text{gap}(L) \\ (n = \delta(m_{\text{init}}, O_i)) \pi_j b_j(O_t) : 0 & \text{if } O_t \neq \text{gap}(L) \end{cases} \quad \text{for } 1 \leq j \leq N_s \text{ and } n \in S_M
\]

\[
\alpha_{t+1}(j, n) = \begin{cases} \left( \sum_{i \in \{1 \ldots N_s\}} \alpha_t(i, m) A_{i, j} \right) b_j(O_{t+1}) & \text{if } O_{t+1} \neq \text{gap}(L) \\ L(0) \alpha_t(j, n) + \sum_{\ell > 0} L(\ell) \sum_{i \in \{1 \ldots N_s\}} \alpha_t(i, m) g_t(i, m, j, n) & \text{if } O_{t+1} = \text{gap}(L) \end{cases} \quad \text{for } 1 \leq t \leq T - 1 \text{ and } 1 \leq j \leq N_s \text{ and } n \in S_M
\]
Solution: Precomputing RVSE

\[ V = \{O_1, O_2, O_3\} \]

workset = \{\alpha_0\}

\[ \alpha_0 = \text{the probability distribution with } \alpha_0(j, m_{\text{init}}) = \pi_0(j), \text{ and } \alpha_0(j, n) = 0 \text{ for } n \neq m_{\text{init}} \]

workset = \{\alpha_0\}

nodes = \{\alpha_0\}

while workset \neq \emptyset

\[ \alpha = \text{workset.removeOne();} \]

for each observation symbol \( O \) in \( V \)

\[ \alpha' = \text{normalize-successor}(\alpha, O); \]

if dead(\( \alpha' \))

\[ \text{continue;} \]

endif

if \( \alpha' \in \text{nodes} \)

\[ \text{add an exact edge labeled with } O \text{ from } \alpha \text{ to } \alpha'; \]

elseif there exists \( \alpha'' \) in nodes such that closeEnough(\( \alpha', \alpha'' \))

\[ \text{add an approximate edge labeled with } O \text{ from } \alpha \text{ to } \alpha'; \]

else

\[ \text{add } \alpha' \text{ to } \text{nodes and workset; } \]

\[ \text{add an exact edge labeled with } O \text{ from } \alpha \text{ to } \alpha'; \]

endif

endfor

 endwhile
Solution: Precomputing RVSE

\[ V = \{ O_1, O_2, O_3 \} \]

\[ \text{workset} = \{ \alpha_0 \} \]

\[ \alpha_0 = \text{the probability distribution with } \alpha_0(j,m_{\text{init}}) = \pi_0(j), \text{ and } \alpha_0(j,n) = 0 \text{ for } n \neq m_{\text{init}} \]

\[ \text{workset} = \{ \alpha_0 \} \]

\[ \text{nodes} = \{ \alpha_0 \} \]

\[ \text{while workset} \neq \emptyset \]

\[ \alpha = \text{workset}.\text{removeOne}(); \]

\[ \text{for each observation symbol } O \text{ in } V \]

\[ \alpha' = \text{normalize}(\text{successor}(\alpha, O)); \]

\[ \text{if } \text{dead}(\alpha') \]

\[ \text{continue}; \]

\[ \text{endif} \]

\[ \text{if } \alpha' \in \text{nodes} \]

\[ \text{add an exact edge labeled with } O \text{ from } \alpha \text{ to } \alpha'; \]

\[ \text{elseif there exists } \alpha'' \text{ in nodes such that closeEnough}(\alpha', \alpha''); \]

\[ \text{add an approximate edge labeled with } O \text{ from } \alpha \text{ to } \alpha'; \]

\[ \text{else} \]

\[ \text{add } \alpha' \text{ to } \text{nodes} \text{ and workset}; \]

\[ \text{add an exact edge labeled with } O \text{ from } \alpha \text{ to } \alpha'; \]

\[ \text{endif} \]

\[ \text{endfor} \]

\[ \text{ endwhile} \]
Solution: Precomputing RVSE

\[ V = \{O_1, O_2, O_3\} \]

**workset** = \{\(\alpha_0, \alpha_{02}\}\}

\(\alpha_0\) = the probability distribution with \(\alpha_0(j, m_{\text{init}}) = \pi_0(j)\), and \(\alpha_0(j, n) = 0\) for \(n \neq m_{\text{init}}\)

**workset** = \{\(\alpha_0\)\}

**nodes** = \{\(\alpha_0\)\}

while **workset** ≠ \(\emptyset\)

\(\alpha = \text{workset.removeOne}()\);

for each observation symbol \(O\) in \(V\)

\(\alpha' = \text{normalize}(\text{successor}(\alpha, O))\);

if dead(\(\alpha'\))

continue;

endif

if \(\alpha' \in \text{nodes}\)

add an exact edge labeled with \(O\) from \(\alpha\) to \(\alpha'\);

elseif there exists \(\alpha''\) in nodes such that closeEnough(\(\alpha', \alpha''\))

add an approximate edge labeled with \(O\) from \(\alpha\) to \(\alpha'\);

else

add \(\alpha'\) to **nodes** and **workset**;

add an exact edge labeled with \(O\) from \(\alpha\) to \(\alpha'\);

endif

endfor

endwhile
Solution: Precomputing RVSE

\[ V = \{ O_1, O_2, O_3 \} \]

**workset** = \{ \alpha_0, \alpha_{02} \}

**nodes** = \{ \alpha_0, \alpha_{02}, \alpha_03 \} \quad \alpha_{03} = \alpha_{02} \rightarrow \alpha_{03} \in \text{nodes}

\[
\alpha_0 = \text{the probability distribution with } \alpha_0(j, m_{\text{init}}) = \pi_0(j), \text{ and } \alpha_0(j, n) = 0 \text{ for } n \neq m_{\text{init}}
\]

\[ \text{workset = \{ } \alpha_0 \} \]

\[ \text{nodes = \{ } \alpha_0 \} \]

**while** workset \neq \emptyset

\[
\alpha = \text{workset.removeOne();}
\quad \alpha = \text{normalize(successor(\alpha, O));}
\]

**if** dead(\alpha')

\quad continue;

**endif**

**if** \alpha' \in \text{nodes}

\quad add an exact edge labeled with O from \alpha to \alpha';

**elseif** there exists \alpha'' in nodes such that closeEnough(\alpha', \alpha'');

\quad add an approximate edge labeled with O from \alpha to \alpha';

**else**

\quad add \alpha' to nodes and workset;

\quad add an exact edge labeled with O from \alpha to \alpha';

**endif**

**endfor**

**endwhile**
Solution: Precomputing RVSE

\[ V = \{O_1, O_2, O_3\} \]

**workset** = \{\(\alpha_0, \alpha_{02}\}\)

\[ \alpha_0 \]

\[ \text{nodes} = \{\alpha_0, \alpha_{02}\} \quad \alpha_{03} = \alpha_{02} \rightarrow \alpha_{03} \in \text{nodes} \]

\[ \alpha_0 = \text{the probability distribution with } \alpha_0(j, m_{init}) = \pi_0(j), \text{ and } \alpha_0(j, n) = 0 \text{ for } n \neq m_{init} \]

**workset** = \{\(\alpha_0\}\)

**nodes** = \{\(\alpha_0\}\)

while \(\text{workset} \neq \emptyset\)

\[ \alpha = \text{workset.removeOne();} \]

for each observation symbol \(O\) in \(V\)

\[ \alpha' = \text{normalize(successor(\(\alpha, O\)))}; \]

if dead(\(\alpha'\))

continue;
endif

if \(\alpha' \in \text{nodes}\)

add an exact edge labeled with \(O\) from \(\alpha\) to \(\alpha'\);
elseif there exists \(\alpha''\) in nodes such that closeEnough(\(\alpha', \alpha''\))

add an approximate edge labeled with \(O\) from \(\alpha\) to \(\alpha'\);
else

add \(\alpha'\) to \text{nodes} and \text{workset};

add an exact edge labeled with \(O\) from \(\alpha\) to \(\alpha'\);
endif
endfor
endwhile
Solution: Precomputing RVSE

- $V = \{O_1, O_2, O_3\}$

- $\alpha_0 = \text{the probability distribution with } \alpha_0(j, m_{\text{init}}) = \pi_0(j), \text{ and } \alpha_0(j, n) = 0 \text{ for } n \neq m_{\text{init}}$

- $\text{workset} = \{\alpha_0\}$

- $\text{nodes} = \{\alpha_0\}$

- While $\text{workset} \neq \emptyset$

  - $\alpha = \text{workset}.\text{removeOne}();$

  - For each observation symbol $O$ in $V$

    - $\alpha' = \text{normalize}(\text{successor}(\alpha, O));$

    - If $\text{dead}(\alpha')$

      - Continue;

    - Endif

    - If $\alpha' \in \text{nodes}$

      - Add an exact edge labeled with $O$ from $\alpha$ to $\alpha'$;

    - Elseif there exists $\alpha''$ in nodes such that $\text{closeEnough}(\alpha', \alpha'')$

      - Add an approximate edge labeled with $O$ from $\alpha$ to $\alpha'$;

    - Else

      - Add $\alpha'$ to $\text{nodes}$ and $\text{workset}$;

      - Add an exact edge labeled with $O$ from $\alpha$ to $\alpha'$;

    - Endif

- Endwhile
Solution: Precomputing RVSE

\[ V = \{ O_1, O_2, O_3 \} \]

\[
\text{workset} = \{ \alpha_0, \alpha_{02} \}
\]

\[
\alpha_0 = \text{the probability distribution with } \alpha_0(j, m_{\text{init}}) = \pi_0(j), \text{ and } \alpha_0(j, n) = 0 \text{ for } n \neq m_{\text{init}}
\]

\[
\text{workset} = \{ \alpha_0 \}
\]

\[
\text{nodes} = \{ \alpha_0 \}
\]

while \( \text{workset} \neq \emptyset \)

\[
\alpha = \text{workset}.\text{removeOne}();
\]

for each observation symbol \( O \) in \( V \)

\[
\alpha' = \text{normalize} (\text{successor} (\alpha, O));
\]

if \( \text{dead}(\alpha') \)

\[
\text{continue};
\]

endif

if \( \alpha' \in \text{nodes} \)

add an exact edge labeled with \( O \) from \( \alpha \) to \( \alpha' \);

elseif there exists \( \alpha'' \) in nodes such that \( \text{closeEnough}(\alpha', \alpha'') \)

add an approximate edge labeled with \( O \) from \( \alpha \) to \( \alpha' \);

else

add \( \alpha' \) to \( \text{nodes} \) and \( \text{workset} \);

add an exact edge labeled with \( O \) from \( \alpha \) to \( \alpha' \);

endif

endfor

endwhile

Proof of Termination in the paper !!!
1) Compose (offline) an HMM with a DFSM to get a DTMC

Given an HMM $H = \langle S, A, V, B, \pi \rangle$ and a DFSM $M = \langle S_M, m_{init}, V, \delta, F \rangle$

their composition is a DTMC $D = \langle S_D, s_0, P \rangle$ where:

- $S_D = (S \times S_M) \cup \{ s_0 \}$
- $s_0$ is the initial state

- the trans. probability $P$ with

$$P\left(\bar{s}_0, (s_i, m_{init})\right) = \pi_i, \text{ with } 1 \leq i \leq |S|$$

$$P\left((s_i, s_{j_1}), (s_{j_2}, s_{j_3})\right) = A_{i,j_2} \sum_{v_k \in V} b_{j_2}(v_k)$$

We extend $D$ with a reward function $\rho(\bar{s}) = 1$
Predictive Analysis of Criticality

1) Compose (offline) an HMM with a DFSM to get a DTMC

2) Compute (offline) the expected distance $\text{ExpDist}$

We check in PRISM:
$R=? [F m=4]= 3242.81$
• Lock discipline: `btrfs_space_info`

• *Protected* fields are always accessed with a lock

• All `btrfs_space_info` objects can be accessed from *any* thread

```c
struct btrfs_space_info {
    /* Unprotected */
    u64 flags;
    /* Protected */
    u64 total_bytes;
    spinlock_t lock;
};
```
Monitor

Protected, Unprotected, Unlock

Protected, Unprotected, Lock

Unprotected, Unlock

Lock

Unlock

Protected

(All events)
Instrumentation

- Access instrumented using GCC plug-ins
- Function-body duplication used to enable/disable instrumentation
Hardware Supervision

• 100% monitoring for a limited number of objects
  – One per thread in our prototype
• Provided by hardware debug registers
• Used to monitor highest priority objects
  – Highest priority = closest to error state (criticality)
## Results

<table>
<thead>
<tr>
<th>Sampling Prob.</th>
<th>No supervision</th>
<th>Random supervision</th>
<th>Adaptive supervision</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>FalseAlarm</td>
<td>ErrDet</td>
<td>FalseAlarm</td>
</tr>
<tr>
<td>50%</td>
<td>30.3</td>
<td>23.0%</td>
<td>11.7</td>
</tr>
<tr>
<td>75%</td>
<td>47</td>
<td>31.2%</td>
<td>36</td>
</tr>
<tr>
<td>85%</td>
<td>5502</td>
<td>34.1%</td>
<td>5606</td>
</tr>
</tbody>
</table>
## Results

| Sampling Prob. | No supervision | | Random supervision | | Adaptive supervision | |
|----------------|----------------|----------------|-------------------|-------------------|-------------------|
|                | FalseAlarm     | ErrDet         | FalseAlarm        | ErrDet            | FalseAlarm        | ErrDet            |
| 50%            | 30.3           | 23.0%          | 11.7              | 57.4%             | 12                | 50.1%             |
| 75%            | 47             | 31.2%          | 36                | 69.3%             | 17                | 79.4%             |
| 85%            | 5502           | 34.1%          | 5606              | 72.3%             | 5449              | 85.1%             |

**FalseAlarm:**
For a run with no errors, how many objects had a high error probability (> 80%) at the end of the run?

These are false positives.
## Results

<table>
<thead>
<tr>
<th>Sampling Prob.</th>
<th>No supervision</th>
<th>Random supervision</th>
<th>Adaptive supervision</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>FalseAlarm</td>
<td>ErrDet</td>
<td>FalseAlarm</td>
</tr>
<tr>
<td>50%</td>
<td>30.3</td>
<td>23.0%</td>
<td>11.7</td>
</tr>
<tr>
<td>75%</td>
<td>47</td>
<td>31.2%</td>
<td>36</td>
</tr>
<tr>
<td>85%</td>
<td>5502</td>
<td>34.1%</td>
<td>5606</td>
</tr>
</tbody>
</table>

**ErrDet:**
Consider an error “detected” if the object had a high error probability (> 80%) after the error occurred.

We ran this test with manually inserted errors and measured what percent were detected.
## Results

<table>
<thead>
<tr>
<th>Sampling Prob.</th>
<th>No supervision</th>
<th></th>
<th>Random supervision</th>
<th></th>
<th>Adaptive supervision</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>FalseAlarm</td>
<td>ErrDet</td>
<td>FalseAlarm</td>
<td>ErrDet</td>
<td>FalseAlarm</td>
<td>ErrDet</td>
</tr>
<tr>
<td>50%</td>
<td>30.3</td>
<td>23.0%</td>
<td>11.7</td>
<td>57.4%</td>
<td>12</td>
<td>50.1%</td>
</tr>
<tr>
<td>75%</td>
<td>47</td>
<td>31.2%</td>
<td>36</td>
<td>69.3%</td>
<td>17</td>
<td>79.4%</td>
</tr>
<tr>
<td>85%</td>
<td>5502</td>
<td>34.1%</td>
<td>5606</td>
<td>72.3%</td>
<td>5449</td>
<td>85.1%</td>
</tr>
</tbody>
</table>

- With high enough sampling, Adaptive Supervision does better than supervising randomly.
- We are still investigating why false positive rates get high for high sampling rates.