Decentralized Crash-Resilient Runtime Verification

Shokoufeh Kazemlou

Department of Computing and Software
McMaster University

December 13, 2017

Overview

Presentation outline

Motivation

Traditional Verification

Exhaustive verification methods are extremely valuable to ensure system-wide correctness.

They often require developing an abstract model of the system and may suffer from the infamous state-explosion problem.

Motivation

Traditional Verification

Exhaustive verification methods are extremely valuable to ensure system-wide correctness.

They often require developing an abstract model of the system and may suffer from the infamous state-explosion problem.

Runtime Verification

Runtime verification (RV) refers to a technique, where a monitor checks at run time whether or not the execution of a system under inspection satisfies a given correctness property.

RV complements exhaustive verification techniques as well as underapproximated methods such as testing and tracing.

Motivation

RV in Distributed Systems

Designing a decentralized runtime monitor for a distributed system is an especially difficult task since it deals with

- computing global snapshots at run time, and
- estimating the total order of events

in order for the monitor to reason about the temporal behavior of the system.

Presentation outline

3-Valued LTL (LTL$_3$) [Bauer, Leucker, Schallhart 11]

3-valued LTL evaluates LTL formulas for finite words with an eye on possible future extensions.

Three Truth Values

The set of truth values is $\mathbb{B}_3 = \{\top, \bot, ?\}$, where

- $\top$: the formula is permanently satisfied no matter how the current execution extends,
- $\bot$: the formula is permanently violated no matter how the current execution extends
- ?: denotes an unknown verdict; i.e., there exist extensions that can falsify or make true the formula.

3-Valued LTL

LTL$_3$ Semantics

Let $u \in \Sigma^*$ be a finite word. The truth value of an LTL$_3$ formula φ with respect to u, denoted by $[u \models_3 \varphi]$, is defined as follows:

$$[u \models_3 \varphi] = \begin{cases} \top & \text{if} \quad \forall w \in \Sigma^\omega : uw \models \varphi \\ \bot & \text{if} \quad \forall w \in \Sigma^\omega : uw \not\models \varphi \\ ? & \text{otherwise.} \end{cases}$$

3-Valued LTL

LTL$_3$ Monitor

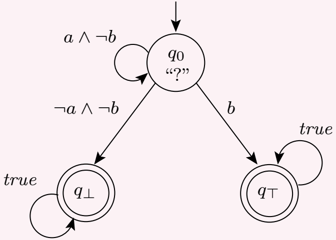
Let φ be an LTL formula. The LTL$_3$ monitor of φ is the unique deterministic finite state machine $\mathcal{M}_3^\varphi = (\Sigma, Q, q_0, \delta, \lambda)$, where Q is a set of states, q_0 is the initial state, $\delta \subseteq Q \times \Sigma \times Q$ is the transition relation, and $\lambda : Q \to \mathbb{B}_3$, is a function such that:

$$\lambda(\delta(q_0, u)) = [u \models_3 \varphi]$$

for every finite word $u \in \Sigma^*$. $\square$

Example

LTL$_3$ monitor for $a\,\mathbf{U}\,b$

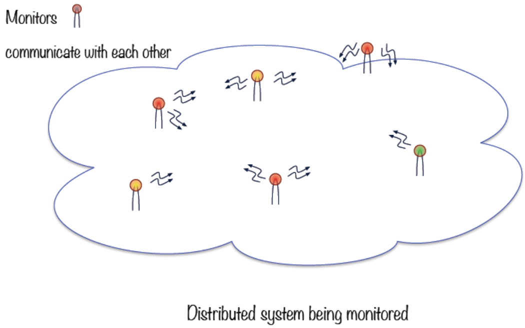


Presentation outline

Problem Statement

Distributed Monitors

Let $\mathcal{M} = \{M_1, M_2, \ldots, M_n\}$ be a set of distributed monitors monitoring an underlying system.



Synchronous Monitoring

Local Monitor Algorithm

Data: LTL formula φ and state s_j

Result: a verdict from $\mathbb{B}_3$

Let $\mathcal{S}_i^{s_j}$ be the initial concrete local state of the monitor

$LS_i^1 \leftarrow \mu(\mathcal{S}_i^{s_j}, \varphi)$

for $r = 1, 2, \cdots$ **do**

- ***Send:*** broadcasts its current abstract local state LS_i^r
- ***Receive:*** let $\Pi_i^r = \{LS_j^r\}_{j\in[1,n]}$ be the set of all messages received at round r.
- ***Computation:*** $LS_i^{r+1} \leftarrow LC(\Pi_i^r)$

emits a verdict from $\mathbb{B}_3$

Problem Statement

A non-faulty monitor should compute and emit a verdict that a centralized monitor that has global view of the system would compute. Formally:

$$\forall i \in [1, n] : M_i \text{ is non-faulty } \rightarrow \nu_i = [\alpha \models_3 \varphi]$$

Presentation outline

Related Work

Central Monitor

- H. Chauhan and V. K. Garg and A. Natarajan and N. Mittal. A Distributed Abstraction Algorithm for Online Predicate Detection (SRDS 2013).
- N Mittal and V. K. Garg. Techniques and applications of computation slicing (Distributed Computing 2005).
- V. A. Ogale and V. K. Garg. Detecting Temporal Logic Predicates on Distributed Computations (DISC 2007).

Related Work

Fault-free Setting

- A. Bauer and Y. Falcone. Decentralised LTL monitoring (FMSD 2016).
- C. Colombo and Y. Falcone. Organising LTL monitors over distributed systems with a global clock (FMSD 2016).
- M. Mostafa, B. Bonakdarpour. Decentralized Runtime Verification of LTL Specifications in Distributed Systems. (IPDPS 2015).

Fault-tolerant Distributed Monitoring

B. Bonakdarpour, P. Fraigniaud, S. Rajsbaum, D. A. Rosenblueth, C. Travers. Decentralized Asynchronous Crash-Resilient Runtime Verification (CONCUR 2016).

Presentation outline

Contributions

Contributions

- An automata-based distributed LTL monitoring algorithm for the decentralized crash-resilient synchronous monitoring.
- Reducing the message size overhead from $|AP|$ per message, to $\log(m_q)$, where m_q is the number of outgoing transitions from the current monitor state in each local monitor's automaton.
- Introducing an Extended LTL_3 monitor for synchronous/asynchronous crash-resilient monitoring.

Presentation outline

Synchronous Monitoring

Distributed Synchronous Setting

Finite trace $\alpha = s_0 s_1 \cdots s_k$;
Set of synchronous monitors $\mathcal{M} = \{M_1, M_2, \cdots, M_n\}$;
Correctness property expressed by an LTL formula φ.

Algorithm Sketch

1. takes a sample from state s_j;
2. broadcasts a message containing its current observation, and receives messages from other monitors;
3. performs a local computation and updates its current observation;
4. emits a truth value from $\mathbb{B}_3$.

Synchronous Monitoring

Uniform Consensus

Eeach process proposes a value, and the processes have to collectively agree on the same value.

- **Validity:** A decided value is a proposed value.
- **Agreement:** No two processes decide different values.
- **Termination:** Every correct process decides.

Validity Specification in Synchronous Monitoring

The decided value must be the same value that a centralized monitor with full view of the system would compute.

Number of Rounds

The lower bound on the number of rounds required to consistently monitor the system is $f + 1$, where f is the total number of crashes the system can tolerate.

Synchronous Monitoring

Local Monitor Algorithm

Data: LTL formula φ and state s_j

Result: a verdict from $\mathbb{B}_3$

Let $\mathcal{S}_i^{s_j}$ be the initial concrete local state of the monitor

$LS_i^1 \leftarrow \mu(\mathcal{S}_i^{s_j}, \varphi)$

for $r = 1, 2, \cdots$ **do**

- ***Send:*** broadcasts its current abstract local state LS_i^r
- ***Receive:*** let $\Pi_i^r = \{LS_j^r\}_{j \in [1,n]}$ be the set of all messages received at round r.
- ***Computation:*** $LS_i^{r+1} \leftarrow LC(\Pi_i^r)$

emits a verdict from $\mathbb{B}_3$

Challenges in Synchronous Monitoring

Example

Let $\varphi = \mathbf{F}(a \wedge b)$, $AP = \{a, b\}$, and $\mathcal{M} = \{M_1, M_2, M_3, M_4\}$. Suppose $s = \{a, b\}$ is the current global state of the system, and the initial samples of the monitors are as follows:

sample

	a	b
M_1	*true*	$\natural$
M_2	$\natural$	*true*
M_3	$\natural$	*true*
M_4	$\natural$	*true*

round 1

	a	b
M_1	crashed	crashed
M_2	*true*	*true*
M_3	$\natural$	*true*
M_4	$\natural$	*true*

round 2

	a	b
M_1	crashed	crashed
M_2	crashed	crashed
M_3	*true*	*true*
M_4	$\natural$	*true*

round 3

	a	b
M_1	crashed	crashed
M_2	crashed	crashed
M_3	*true*	*true*
M_4	*true*	*true*

Synchronous Monitoring

Challenge

If each monitor broadcasts its sample $\Rightarrow$ the message size is $|AP|$

Reducing the Message Size Overhead

We introduce an algorithm which decreases the message size from $|AP|$ to $\log(m_q)$ where m_q is the number of outgoing transitions from monitor state q.

Automata-based Synchronous Monitoring

The General Idea

- Each local monitor M_i evaluates the input formula and computes a possible set of verdicts;

$$V_i = \{\delta(q, s') \mid s' \in E(\mathcal{S}_i^s)\}$$

- At each round, each monitor M_i broadcasts its verdict set V_i, and computes a new verdict set by applying the intersection function on the verdict sets received from other monitors;

$$V_i^{r+1} = LC(\Pi_i^r) = \bigcap_{j \in [1,n]} \{V_j^r\} = \bigcap_{j \in [1,n]} \{V_j^r\}$$

- After $f + 1$ rounds of communication, each monitor emits the verdict that a centralized monitor that has the global view of the system would compute.

Automata-based Synchronous Monitoring

Local Monitor Algorithm

Data: LTL_3 monitor $\mathcal{M}^{\varphi}$ and state s_j

Result: a verdict from $\mathbb{B}_3$

Let $\mathcal{S}_i^{s_j}$ be the initial concrete local state of the monitor

$LS_i^1 \leftarrow \mu_2(\mu_1(\mathcal{S}_i^{s_j}, \mathcal{M}_{\varphi})) = V_i^1$

for $r = 1, \cdots, f+1$ **do**

- ***Send:*** broadcasts its current abstract local state $LS_i^r = V_i^r$
- ***Receive:*** let $\Pi_i^r = \{V_j^r\}_{j \in [1,n]}$ be the set of all messages received at round r.
- ***Computation:*** $LS_i^{r+1} \leftarrow LC_i(\Pi_i^r) = \bigcap_{j \in [1,n]} \{V_j^r\}$

emit $\lambda_e(v_i)$

Automata-based Synchronous Monitoring

Example

Let $\varphi = \mathbf{F}(a \wedge b)$.

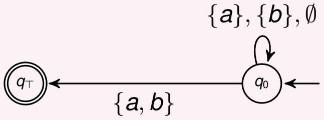


Figure: LTL$_3$ monitor of $\varphi = \mathbf{F}(a \wedge b)$.

And suppose the initial samples of the monitors are as follows:

sample

	a	b	V_i^1
M_1	*true*	$\natural$	$\{q_0, q_\top\}$
M_2	$\natural$	*true*	$\{q_0, q_\top\}$
M_3	$\natural$	$\natural$	$\{q_0, q_\top\}$
M_4	$\natural$	$\natural$	$\{q_0, q_\top\}$

Automata-based Synchronous Monitoring

Example

round 1

	V_i^2
M_1	crashed
M_2	$\{q_0, q_\top\}$
M_3	$\{q_0, q_\top\}$
M_4	$\{q_0, q_\top\}$

round 2

	V_i^3
M_1	crashed
M_2	crashed
M_3	$\{q_0, q_\top\}$
M_4	$\{q_0, q_\top\}$

round 3

	V_i^4
M_1	crashed
M_2	crashed
M_3	$\{q_0, q_\top\}$
M_4	$\{q_0, q_\top\}$

As we see $|V_3| = |V_4| > 1$. Therefore, they cannot emit the correct verdict $\top$ as we have $[\{a, b\} \models_3 \mathbf{F}(a \wedge b)] = \top$.

Insufficiency of LTL3 Monitor

The LTL3 monitor of $\varphi = \mathbf{F}(a \wedge b)$ is not sufficient to distinguish the correct verdict when local monitors have partial view of the system.

Automata-based Synchronous Monitoring

Extended LTL$_3$ Monitor

Input: LTL$_3$ monitor $\mathcal{M}^{\varphi} = \{\Sigma, Q, q_0, \delta, \lambda\}$
output: Extended LTL$_3$ monitor $\mathcal{M}^{\varphi}_e = \{\Sigma, Q_e, q_0, \delta_e, \lambda_e\}$
Where,

$Q \subseteq Q_e$, q_0
q_0 is the initial state
$\delta_e : Q_e \times \Sigma \rightarrow 2^{Q_e}$ is a transition function
$\lambda_e : Q_e \rightarrow \mathbb{B}_3$ is a mapping function, such that:

1. for every non-empty finite trace $\alpha \in \Sigma^*$, we have $\lambda_e(\delta_e(q_0, \alpha)) = \lambda(\delta(q_0, \alpha))$.
2. at every $q \in Q_e$ we have $|\mathcal{I}^q| = 1$.

Extended LTL$_3$ Monitor Construction

Transition

A transition t^i_j from monitor state q_i to monitor state q_j is defined as follows:

$$t^i_j = \{s \in \Sigma \mid \delta(q_i, s) = q_j\}$$

Indistinguishable Transitions

We say a transition t_1 is indistinguishable from another transition t_2, and denote it by *indisting*?(t_1, t_2), if the following holds:

$$\exists s \in t_2.\ \mathit{covered}?(s, t_1)$$

Covered State

We say state s is covered by transition t, and we denote it by *covered*?(s, t), if we have:

$$\forall ap \in AP.\ \exists s' \in t.\ (ap \in s \Leftrightarrow ap \in s')$$

Extended LTL$_3$ Monitor Construction

Input: $\mathcal{M}^{\varphi} = \{\Sigma, Q, q_0, \delta, \lambda\}$

Output: $\mathcal{M}_e^{\varphi} = \{\Sigma, Q_e, q_0, \delta_e, \lambda_e\}$

$Q_e \leftarrow Q$

for *every* $q_i \in Q$ **do**

  Obtain the set of outgoing transitions T_i from monitor state q_i

  for *every* $t_j^i \in T_i$ **do**

    /* $t_j^i = \{s \in \Sigma \mid \delta(q_i, s) = q_j\}$ */

    /* *N_j denotes the number of transitions from which t_j^i is indistinguishable, and K_j denotes the number of transitions indistinguishable from t_j^i* */

    $N_j \leftarrow 0$, $K_j \leftarrow 0$

    for *every* $t_k^i \in T_i \backslash \{t_j^i\}$ **do**

      if *indisting?*(t_j^i, t_k^i) **then**

        $N_j \leftarrow N_j + 1$

      if *indisting?*(t_k^i, t_j^i) **then**

        $K_j \leftarrow K_j + 1$

    if $N_j > 0$ **then**

      $\{t_{j1}^i, t_{j2}^i\} \leftarrow SPLIT(t_j^i, N_j, K_j, T_i)$

      $T_i \leftarrow \{t_{j1}^i, t_{j2}^i\} \cup T_i \backslash \{t_j^i\}$

      $Q_e \leftarrow \{q_{j1}, q_{j2}\} \cup (Q_e \backslash \{q_j\})$

      if $i \neq j$ **then**

        for *every* $t_k^i \in T_i$ **do**

          $\delta(q_i, s) = q_k$ for every $s \in t_k^i$

        $\delta(q_{j1}, s) \leftarrow \delta(q_i, s)$ for every $s \in \Sigma$

        $\delta(q_{j2}, s) \leftarrow \delta(q_i, s)$ for every $s \in \Sigma$

      if $i = j$ **then**

        for *every* $t_k^i \in T_i$ **do**

          $\delta(q_{j1}, s) = q_k$ for every $s \in t_k^i$

        $\delta(q_{j2}, s) \leftarrow \delta(q_{j1}, s)$ for every $s \in \Sigma$

      $\lambda_e(q_{j1}) \leftarrow \lambda(q_j)$

      $\lambda_e(q_{j2}) \leftarrow \lambda(q_j)$

    else

      $\delta_e(q_i, s) \leftarrow q_j$ for every $s \in t_j^i$

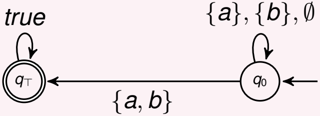
      $\lambda_e(q_j) \leftarrow \lambda(q_j)$

Algorithm 3: Extended LTL$_3$ monitor Construction

Automata-based Synchronous Monitoring

Example

Consider the LTL_3 monitor for $\varphi = \mathbf{F}(a \wedge b)$.

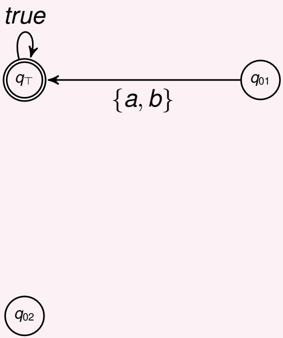


The set of outgoing transitions from monitor state q_0 is $T_0 = \{t_0^0, t_\top^0\}$ where:

$t_0^0 = \{\{a\}, \{b\}, \emptyset\}$
$t_\top^0 = \{\{a, b\}\}$

We can verify that t_0^0 is indistinguishable from $t_\top^0$. Therefore we split transition t_0^0 into two transitions $t_{01}^0 = \{\{a\}\}$ and $t_{02}^0 = \{\{b\}, \emptyset\}$.

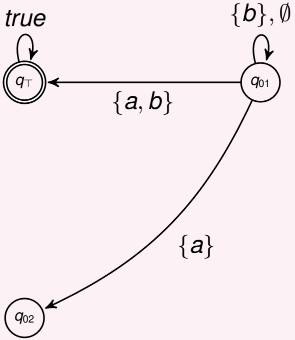
Example



Note that we have

$$\lambda(q_{01}) = \lambda(q_0) = ?$$
$$\lambda(q_{02}) = \lambda(q_0) = ?$$

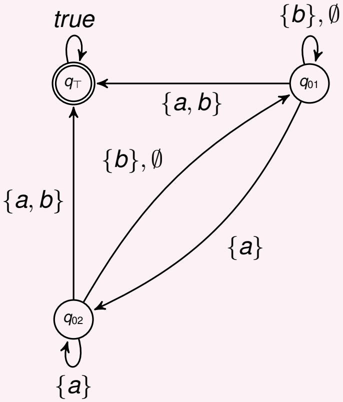
Example



Note that we have

$$\lambda(q_{01}) = \lambda(q_0) = ?$$
$$\lambda(q_{02}) = \lambda(q_0) = ?$$

Example



Note that we have

$$\lambda(q_{01}) = \lambda(q_0) = ?$$
$$\lambda(q_{02}) = \lambda(q_0) = ?$$

Automata-based Synchronous Monitoring

Example

Suppose $s = \{a, b\}$ is the current global state of the system and Let $\varphi = \mathbf{F}(a \wedge b)$. The initial state of the monitors is as follows:

sample

	a	b	LS_i^1
M_1	*true*	$\natural$	$\{q_{02}, q_\top\}$
M_2	$\natural$	*true*	$\{q_{01}, q_\top\}$
M_3	$\natural$	$\natural$	$\{q_{01}, q_{02}, q_\top\}$
M_4	$\natural$	$\natural$	$\{q_{01}, q_{02}, q_\top\}$

round 1

	LS_i^2
M_1	crashed
M_2	$\{q_\top\}$
M_3	$\{q_{01}, q_\top\}$
M_4	$\{q_{01}, q_\top\}$

round 2

	LS_i^3
M_1	crashed
M_2	crashed
M_3	$\{q_\top\}$
M_4	$\{q_{01}, q_\top\}$

round 3

	LS_i^4
M_1	crashed
M_2	crashed
M_3	$\{q_\top\}$
M_4	$\{q_\top\}$

Presentation outline

Asynchronous Monitoring

Asynchronous Monitoring

The system under inspection produces a finite trace $\alpha = s_0 s_1 \cdots s_k$, and is inspected with respect to an LTL formula φ by a set $\mathcal{M} = \{M_1, M_2, \cdots, M_n\}$ of asynchronous distributed monitors.

Local Monitor Algorithm

Data: Extended LTL$_3$ monitor $\mathcal{M}_e^{\varphi} = \{\Sigma, Q_e, q_0, \delta_e, \lambda_e\}$ and state s_j

Result: a set of monitor states

Let $\mathcal{S}_i^{s_j}$ be the initial concrete local state of the monitor ;

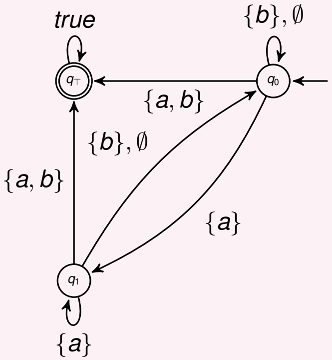
$Snap_i^{s_j} \leftarrow \mathcal{S}_i^{s_j}$

emit $V_i^{s_j} = \mu_2(\mu_1(Snap_i^{s_j}, \mathcal{M}^{\varphi}))$

Asynchronous Monitoring

Example

Let $\varphi = \mathbf{F}(a \wedge b)$ whose Extended LTL$_3$ monitor is given below. Suppose monitors are at monitor state q_0, and let $s = \{a, b\}$ be the global state of the system.



Example

The following tables represent each monitor M_i's initial local snapshot $Snap_i^s$ and its verdict set V_i calculated based on only $Snap_i^s$.

$Snap_1^s$

	M_1	M_2	M_3
a	$true$	$\natural$	$\natural$
b	$\natural$	$\natural$	$\natural$
V_1	$\{q_1, q_\top\}$		

$Snap_2^s$

	M_1	M_2	M_3
a	$\natural$	$\natural$	$\natural$
b	$\natural$	$true$	$\natural$
V_2	$\{q_0, q_\top\}$		

$Snap_3^s$

	M_1	M_2	M_3
a	$\natural$	$\natural$	$\natural$
b	$\natural$	$\natural$	$\natural$
V_3	$\{q_0, q_1, q_\top\}$		

$V_1 \cap V_2 \cap V_3 = q_\top$.

Presentation outline

Conclusion

Conclusion

- We proposed a synchronous monitoring algorithm that copes with f crash failures in a distributed setting. The algorithm solves the synchronous monitoring problem in $f+1$ rounds of communication and reduces the message size overhead from $|AP|$ to $\log(m_q)$.
- We proposed an algorithm for distributed crash-resilient asynchronous RV that consistently monitors the system under inspection with no communication between monitors.

Presentation outline

Future Work

Futur Work

- To address more severe faults, e.g., Byzantine failures.
- To have monitors observe, communicate, and emit verdicts between any two global states.
- To extend our results to the case where the input to the monitors is a sequence of global states and each monitor produces a sequence of verdict sets, one per each global state