Aprroximate Distributed Monitoring under Partial Asynchrony: Balancing Speed with Accuracy

Borzoo Bonakdarpour[1], Anik Momtaz[1], Dejan Ničković[2], and N. Ege Saraç[3]

[1] Michigan State University
[2] AIT Austrian Institute of Technology
[3] Institute of Science and Technology Austria (ISTA)

Abstract. In distributed systems with processes that do not share a global clock, *partial asynchrony* is achieved by clock synchronisation that guarantees bounded clock skew among all applications. Existing solutions for distributed runtime verification under partial asynchrony against temporal logic specifications are exact but suffer from significant computational overhead. In this paper, we propose an *approximate* distributed monitoring algorithm for Signal Temporal Logic (STL) that mitigates this issue by abstracting away potential interleaving behaviors. This conservative abstraction enables a significal speedup of the distributed monitors, albeit with a trade-off in accuracy. We address this trade-off with a methodology that combines our approximate monitor with its exact counterpart, resulting in enhanced monitoring efficiency without sacrifying precision. We validate our approach with multiple experiments, showcasing its effectiveness and efficacy on both a real-world application and synthetic examples.

You can leave notes using the command \firstName{note}.
Page limit: 16 + 4 (appendix)

1 Introduction

Distributed systems are networks of independent agents that work together to achieve a common objective. Distributed systems are everywhere around us and come in many different forms. For example, cloud computing uses distribution of resources and services over the internet to offer to their users a scalable infrastructure with transparent on-demand access to computing power and storage. Swarms of drones represent another family of distributed systems where individual drones collaborate to accomplish tasks like surveillance, search and rescue, or package delivery. While each drone operates independently, it also communicates and coordinates with others to succesfully achieve their common objectives. The individual agents in a distributed system typically do not share a global clock. To coordinate actions across multiple agents, clock synchonisation is often needed.

While perfect clock synchronisation is impractical due to network latency and node failures, algorithms such as the Network Time Protocol (NTP) allow agents to maintain a *bounded skew* between the synchronised clocks. In that case, we say that a distributed system has *partial asynchrony*.

Formal verification of distributed system is a notoriously hard problem, due to the combinatorial explosion of all possible interleavings in the behaviors collected from individual agents. *Runtime verification (RV)* provides a more pragmatic approach, in which a monitor observes a hehavior of a distributed system and checks its correctness against a formal specification. The problem of distributed RV under partial asynchrony assumption has been studied for Linear Temporal Logic (LTL) and Signal Temporal Logic (STL) specification languages. The proposed solutions use Satisfiability-Modulo-Theory (SMT) solving to provide sound and complete distributed monitoring procedures. Although distributed RV monitors consume only a single distributed behavior at a time, this behavior can nevertheless have an excessive number of possible interleavings. Hence, the exact distributed monitors from the literature can still suffer from significant computational overhead.

To mitigate this issue, we present an approach for *approximate* RV of STL specifications under partial asynchrony. In essence, we abstract away potential inteleavings in distributed behaviors in a conservative manner, resulting in an effective over-approximation of global behaviors. This abstraction simplifies the representation of distributed behaviors and the monitoring operations required to evaluate temporal specifications. There is an inevitable trade-off in approximate RV – gains in the monitoring speed-up may result in reduced accuracy. For some applications, reduced accuracy may not be acceptable. Therefore, we propose a methofology that cobines our approximate monitors with their exact counterparts, with the aim to benefit from the enhanced monitoring efficiency without sacrificing precision. We implemented our approach in a prototype tool and performed thorough evaluations on both synthetic and real-world case studies. We first demonstrated that our approximate monitors achieve speed-ups of several orders of magnitudes compared to the exact SMT-based distributed RV solution. We empirically characterized the classes of specifications and behaviors for which our approximate monitoring approach achieves good precision. We finally showed that by combining exact and approximate ditributed RV, there is still a significant efficiency gain on average without the sacrifice of the precision, even in cases where approximate monitors have low accuracy.

2 Preliminaries

TODO: Check and simplify.

We define boolean domain $\mathbb{B} = \{\bot, \top\}$ as the set of boolean truth values, where $\bot < \top$ and they complement each other, i.e., $\overline{\bot} = \top$ and $\overline{\top} = \bot$. We denote by $\mathbb{R}$ the set of reals, $\mathbb{R}_{\geq 0}$ the set of nonnegative reals, and $\mathbb{R}_{>0}$ the set of positive reals. An interval $I \subseteq \mathbb{R}$ of reals with the end points $a < b$ has length $|b - a|$.

Let Σ be a finite *alphabet*. We denote by Σ^* the set of finite words over Σ and by ϵ the empty word. For $u \in \Sigma^*$, we respectively write $\mathsf{prefix}(u)$ and $\mathsf{suffix}(u)$ for the sets of nonempty prefixes and suffixes of u. We also let $\mathsf{infix}(u) = \{v \in \Sigma^* \mid \exists x, y \in \Sigma^* : u = xvy \wedge v \neq \epsilon\}$. For a nonempty word $u \in \Sigma^*$ and $1 \leq i \leq |u|$, we denote by $u[i]$ the ith letter of u, by $u[..i]$ the prefix of u of length i, and by $u[i..]$ the suffix of u of length $|u| - i + 1$. Given $u \in \Sigma^*$ and $\ell \geq 1$, we denote by u^ℓ the word obtained by concatenating u by itself $\ell - 1$ times. Moreover, given $L \subseteq \Sigma^*$, we define $\mathsf{first}(L) = \{u[0] \mid u \in L\}$.

We define the function $\mathsf{destutter} : \Sigma^* \to \Sigma^*$ inductively as follows. For all $\sigma \in \Sigma \cup \{\epsilon\}$, let $\mathsf{destutter}(\sigma) = \sigma$. For all $u \in \Sigma^*$ such that $u = \sigma_1\sigma_2 v$ for some $\sigma_1, \sigma_2 \in \Sigma$ and $v \in \Sigma^*$, let (i) $\mathsf{destutter}(u) = \mathsf{destutter}(\sigma_2 v)$ if $\sigma_1 = \sigma_2$, and (ii) $\mathsf{destutter}(u) = \sigma_1 \cdot \mathsf{destutter}(\sigma_2 v)$ otherwise. By extension, for a set $L \subseteq \Sigma^*$ of finite words, we write $\mathsf{destutter}(L) = \{\mathsf{destutter}(u) \mid u \in L\}$. Given a tuple $(u_1, \ldots, u_m)$ of finite words of the same length, we write $\mathsf{destutter}(u_1, \ldots, u_m)$ for the extension of $\mathsf{destutter}$ defined as expected: requiring the equality condition in (i) to hold for all the words in the tuple.

Moreover, given an integer $k \geq 0$, we define $\mathsf{stutter}_k : \Sigma^* \to \Sigma^*$ such that $\mathsf{stutter}_k(u) = \{v \in \Sigma^* \mid |v| = k \wedge \mathsf{destutter}(v) = \mathsf{destutter}(u)\}$ if $k \geq |\mathsf{destutter}(u)|$, and $\mathsf{stutter}_k(u) = \emptyset$ otherwise.

2.1 Signal Temporal Logic

Let $A, B \subset \mathbb{R}$. A function $f : A \to B$ is *right-continuous* iff $\lim_{a \to c^+} f(a) = f(c)$ for all $c \in A$, and *non-Zeno* iff for every bounded interval $I \subseteq A$ there are finitely many $a \in I$ such that f is not continuous at a.

A *signal* is a right-continuous, non-Zeno, piecewise-constant function $x : [0, d) \to \mathbb{R}$ where $d \in \mathbb{R}_{>0}$ is the duration of x and $[0, d)$ is its temporal domain. Let $x : [0, d) \to \mathbb{R}$ be a signal. An *event* of x is a pair $(t, x(t))$ where $t \in [0, d)$. An *edge* of x is an event $(t, x(t))$ such that $\lim_{s \to t^-} x(s) \neq \lim_{s \to t^+} x(s)$. In particular, an edge is *rising* if $\lim_{s \to t^-} x(s) < \lim_{s \to t^+} x(s)$, and it is *falling* otherwise. A signal $x : [0, d) \to \mathbb{R}$ can be represented finitely by its initial value and edges: if x has m edges, then $x = (t_0, v_0)(t_1, v_1) \ldots (t_m, v_m)$ such that $t_0 = 0$, $t_{i-1} < t_i$, and (t_i, v_i) is an edge of x for all $1 \leq i \leq m$.

Let AP be a set of atomic propositions. The syntax is given by the following grammar where $p \in \mathsf{AP}$ and $I \subseteq \mathbb{R}_{\geq 0}$ is an interval.

$$\varphi := p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \mathcal{U}_I \varphi$$

A *trace* $w = (x_1, \ldots, x_n)$ is a finite vector of signals. We express atomic propositions as functions of trace values at a time point t, i.e., a proposition $p \in \mathsf{AP}$ over a trace $w = (x_1, \ldots, x_n)$ is defined as $f_p(x_1(t), \ldots, x_n(t)) > 0$ where $f_p : \mathbb{R}^n \to \mathbb{R}$ is a function. Given intervals $I, J \subseteq \mathbb{R}_{\geq 0}$, we define $I \oplus J = \{i + j \mid i \in I \wedge j \in J\}$, and we simply write t for the singleton set $\{t\}$.

Below we recall the finite-trace qualitative semantics of STL defined over $\mathbb{B}$ [1]. Let $d \in \mathbb{R}_{>0}$ and $w = (x_1, \ldots, x_n)$ with $x_i : [0, d) \to \mathbb{R}$ for all $1 \leq i \leq n$. Let φ_1, φ_2 be STL formulas and let $t \in [0, d)$.

$$
\begin{aligned}
[w, t \models p]_{\mathsf{STL}} &\iff f_p(x_1(t), \ldots, x_n(t)) > 0 \\
[w, t \models \neg\varphi_1]_{\mathsf{STL}} &\iff \overline{[w, t \models \varphi_1]_{\mathsf{STL}}} \\
[w, t \models \varphi_1 \wedge \varphi_2]_{\mathsf{STL}} &\iff [w, t \models \varphi_1]_{\mathsf{STL}} \wedge [w, t \models \varphi_2]_{\mathsf{STL}} \\
[w, t \models \varphi_1 \mathcal{U}_I \varphi_2]_{\mathsf{STL}} &\iff \exists t' \in (t \oplus I) \cap [0, d) : \\
&\qquad [w, t' \models \varphi_2]_{\mathsf{STL}} \wedge \forall t'' \in (t, t') : [w, t'' \models \varphi_1]_{\mathsf{STL}}
\end{aligned}
$$

We simply write $[w \models \varphi]$ for $[w, 0 \models \varphi]$. We additionally use the following standard abbreviations: $\texttt{false} = p \wedge \neg p$, $\texttt{true} = \neg\texttt{false}$, $\varphi_1 \vee \varphi_2 = \neg(\neg\varphi_1 \wedge \neg\varphi_2)$, $\Diamond_I \varphi = \texttt{true}\mathcal{U}_I\varphi$, and $\Box_I\varphi = \neg\Diamond_I\neg\varphi$. Moreover, the untimed temporal operators are defined through their timed counterparts on the interval $(0, \infty)$, e.g., $\varphi_1 \mathcal{U} \varphi_2 = \varphi_1 \mathcal{U}_{(0,\infty)} \varphi_2$.

2.2 Distributed Semantics of Signal Temporal Logic

We consider an asynchronous and loosely-coupled message-passing system of $n \geq 2$ reliable agents producing a set of signals $x_1, \ldots, x_n$, where for some $d \in \mathbb{R}_{>0}$ we have $x_i : [0, d) \to \mathbb{R}$ for all $1 \leq i \leq n$. The agents do not share memory or a global clock. Only to formalize statements, we speak of a *hypothetical* global clock and denote its value by T. For local time values, we use the lowercase letters t and s.

For a signal x_i, we denote by V_i the set of its events, by $E_i^{\uparrow}$ the set of its rising edges, and by $E_i^{\downarrow}$ that of falling edges. Moreover, we let $E_i = E_i^{\uparrow} \cup E_i^{\downarrow}$. We represent the local clock of the ith agent as an increasing and divergent function $c_i : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ that maps a global time T to a local time $c_i(T)$. We denote by c_i^{-1} the inverse of the local clock function c_i.

We assume that the system is *partially synchronous*: the agents use a clock synchronization algorithm that guarantees a bounded clock skew with respect to the global clock, i.e., $|c_i(T) - c_j(T)| < \varepsilon$ for all $1 \leq i, j \leq N$ and $T \in \mathbb{R}_{\geq 0}$, where $\varepsilon \in \mathbb{R}_{>0}$ is the maximum clock skew.

Definition 1. *A* distributed signal *is a pair $(S, \rightsquigarrow)$, where $S = (x_1, \ldots, x_n)$ is a vector of signals and $\rightsquigarrow$ is the happened-before relation between events in signals extended with the partial synchrony assumption as follows.*

- *For every agent, the events of its signals are totally ordered, i.e., for all $1 \leq i \leq n$ and all $(t, x_i(t)), (t', x_i(t')) \in V_i$, if $t < t'$ then $(t, x_i(t)) \rightsquigarrow (t', x_i(t'))$.*
- *Every pair of events whose timestamps are at least ε apart is totally ordered, i.e., for all $1 \leq i, j \leq n$ and all $(t, x_i(t)) \in V_i$ and $(t', x_j(t')) \in V_j$, if $t + \varepsilon \leq t'$ then $(t, x_i(t)) \rightsquigarrow (t', x_j(t'))$.*

Example 2. TODO: distributed signal, happened-before relation

Definition 3. *Let $(S, \rightsquigarrow)$ be a distributed signal of n signals, and $V = \bigcup_{i=1}^{n} V_i$ be the set of its events. A set $C \subseteq V$ is a* consistent cut *iff for every event in C, all events that happened before it also belong to C, i.e., for all $e, e' \in V$, if $e \in C$ and $e' \rightsquigarrow e$, then $e' \in C$.*

We denote by $\mathbb{C}(T)$ the (infinite) set of consistent cuts at global time T. Given a consistent cut C, its *frontier* $\mathsf{front}(C) \subseteq C$ is the set consisting of the last events in C of each signal, i.e., $\mathsf{front}(C) = \bigcup_{i=1}^{n} \{(t, x_i(t)) \in V_i \cap C \mid \forall t' > t : (t', x_i(t')) \notin V_i \cap C\}$.

Definition 4. *A* consistent cut flow *is a function $\mathsf{ccf} : \mathbb{R}_{\geq 0} \to 2^V$ that maps a global clock value T to the frontier of a consistent cut at time T, i.e., $\mathsf{ccf}(T) \in \{\mathsf{front}(C) \mid C \in \mathbb{C}(T)\}$.*

For all $T, T' \in \mathbb{R}_{\geq 0}$ and $1 \leq i \leq n$, if $T < T'$, then for every pair of events $(c_i(T), x_i(c_i(T))) \in \mathsf{ccf}(T)$ and $(c_i(T'), x_i(c_i(T'))) \in \mathsf{ccf}(T')$ we have $(c_i(T), x_i(c_i(T))) \rightsquigarrow (c_i(T'), x_i(c_i(T')))$. We denote by $\mathsf{CCF}(S, \rightsquigarrow)$ the set of all consistent cut flows of the distributed signal $(S, \rightsquigarrow)$. Observe that a consistent cut flow of a distributed signal induces a vector of synchronous signals which can be evaluated using the standard semantics described in Section 2.1.

Example 5. TODO: consistent cut, frontier, consistent cut flow

Let $(S, \rightsquigarrow)$ be a distributed signal of n signals $x_1, \ldots, x_n$. A consistent cut flow $\mathsf{ccf} \in \mathsf{CCF}(S, \rightsquigarrow)$ yields a trace $w_{\mathsf{ccf}} = (x'_1, \ldots x'_n)$ on a temporal domain $[0, D)$ where $D \in \mathbb{R}_{>0}$ such that $(c_i(T), x_i(c_i(T))) \in \mathsf{ccf}(T)$ implies $x'_i(T) = x_i(c_i(T))$ for all $1 \leq i \leq n$ and $T \in [0, D)$. The set of traces of $(S, \rightsquigarrow)$ is given by $\mathsf{Tr}(S, \rightsquigarrow) = \{w_{\mathsf{ccf}} \mid \mathsf{ccf} \in \mathsf{CCF}(S, \rightsquigarrow)\}$.

We define the satisfaction of an STL formula φ by a distributed signal $(S, \rightsquigarrow)$ over a three-valued domain $\{\top, \bot, ?\}$.

$$[(S, \rightsquigarrow) \models \varphi]_{\mathsf{STL}} = \begin{cases} \top & \text{if } \forall w \in \mathsf{Tr}(S, \rightsquigarrow) : [w \models \varphi]_{\mathsf{STL}} \\ \bot & \text{if } \forall w \in \mathsf{Tr}(S, \rightsquigarrow) : [w \models \neg\varphi]_{\mathsf{STL}} \\ ? & \text{otherwise} \end{cases}$$

If the set of synchronous traces $\mathsf{Tr}(S, \rightsquigarrow)$ defined by a distributed signal $(S, \rightsquigarrow)$ is contained in the set of traces allowed by the formula φ, then $(S, \rightsquigarrow)$ satisfies φ. Similarly, if $\mathsf{Tr}(S, \rightsquigarrow)$ has an empty intersection with the set of traces Φ defines, then $(S, \rightsquigarrow)$ violates φ. Otherwise, the evaluation is inconclusive since some traces satisfy the property and some violate it.

3 Overapproximation of Synchronous Traces

TODO: summary. main idea is to reduce bookkeeping as much as possible to improve scalability. note we use bool signals for convenience of demonstration, but the methods are general unless otherwise stated.

STL$^+$ has the same syntax as STL and the below semantics. $\mathsf{Tr}^+(S, \rightsquigarrow)$ is the overapproximation we compute.

$$[(S, \rightsquigarrow) \models \varphi]_{\mathsf{STL}^+} = \begin{cases} \top & \text{if } \forall w \in \mathsf{Tr}^+(S, \rightsquigarrow) : [w \models \varphi]_{\mathsf{STL}} \\ \bot & \text{if } \forall w \in \mathsf{Tr}^+(S, \rightsquigarrow) : [w \models \neg\varphi]_{\mathsf{STL}} \\ ? & \text{otherwise} \end{cases}$$

Theorem 6. *For every STL formula φ and every distributed signal $(S, \rightsquigarrow)$, if $[(S, \rightsquigarrow) \models \varphi]_{\mathsf{STL}^+}$ then $[(S, \rightsquigarrow) \models \varphi]_{\mathsf{STL}}$.*

3.1 Uncertainty Regions and Canonical Segmentations

Consider a signal $x : [0, d) \to \mathbb{B}$ with a rising edge at local time t. Due to clock skew, the rising edge of x occurs in the range $(t - \varepsilon, t + \varepsilon)$ according to the global clock. This range is called an *uncertainty region* because in the interval $(t - \varepsilon, t + \varepsilon)$ the monitor cannot tell the value of x precisely, but only that it changes from $\bot$ to $\top$. To systematically reason about this uncertainty, we use the notion of segmentation of temporal domains of signals.

Given a temporal domain $I = [0, d) \subset \mathbb{R}_{\geq 0}$, a *segmentation* of I is a partition of I into finitely many intervals $I_1, \ldots, I_k$ of the form $I_j = [t_j, t_{j+1})$ such that $t_j < t_{j+1}$ for all $1 \leq j \leq k$. By extension, a segmentation of a collection of signals with the same temporal domain I is a segmentation of I.

Let $x : [0, d) \to \mathbb{R}$ be a signal and $(t, x(t))$ be an edge of x. We define $\theta_{\text{lo}}(x, t) = \max(0, t - \varepsilon)$ and $\theta_{\text{hi}}(x, t) = t_i + \varepsilon$. Intuitively, θ_{lo} and θ_{hi} give us the lower and upper bounds on the value of the monitor's clock for a given edge, i.e., the end points of the uncertainty region of the given edge. We use these to describe a canonical segmentation of a distributed signal.

Let $(S, \rightsquigarrow)$ be a distributed signal of n signals. For each signal x_i, let F_i be the set of end points of its uncertainty regions. Let $d' = \max(d, \max(\bigcup_{i=1}^{n} F_i))$, which corresponds to the duration of the distributed signal with respect to the monitor's clock. We define $F = \{0, d'\} \cup \bigcup_{i=1}^{n} F_i$ and let $(a_j)_{1 \leq j \leq |F|}$ be a nondecreasing sequence of clock values corresponding to the elements of F. Then, the *canonical segmentation* of $(S, \rightsquigarrow)$ is $G_S = \{I_1, \ldots, I_{|F|-1}\}$ where $I_j = [a_j, a_{j+1})$ for all $1 \leq j < |F|$. We present an example in **??**

Fig. 1. TODO

3.2 Value Expressions

TODO

3.3 Overapproximate Evaluation

TODO – for each signal concat the valexpr sets. choose one elt from each. stutter to the right length. this belongs to the set.

4 Monitoring Algorithm

TODO

5 Experimental Evaluation

TODO

6 Conclusion

TODO

References

1. Maler, O., Nickovic, D.: Monitoring properties of analog and mixed-signal circuits. Int. J. Softw. Tools Technol. Transf. **15**(3), 247–268 (2013). https://doi.org/10.1007/s10009-012-0247-9