You can leave notes using the command \firstName{note}.

1 Preliminaries

We define $\mathbb{B} = \{\bot, \top\}$ 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.

1.1 Signal Model

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.

Definition 1. *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.*

We consider an asynchronous and loosely-coupled message-passing system of $N \geq 2$ reliable agents, denoted $A_1, \ldots, A_N$. The agents produce a set of $n \geq 2$ 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 function $\pi : \{x_1, \ldots, x_n\} \to \{A_1, \ldots, A_N\}$ maps each signal to the agent it is produced by. **Borzoo: Check later if this is really needed.**

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.

We represent the local clock of the agent A_i 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 $c_i(T)$ of A_i. 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., $|c_i(T) - T| < \varepsilon$ for all $1 \leq i \leq N$ and $T \in \mathbb{R}_{\geq 0}$, where $\varepsilon \in \mathbb{R}_{>0}$ is the maximum clock skew. **Borzoo: Another way is** $|c(i) - c(j)| < \epsilon$ **Anik:** $|c_i(T) - c_j(T)| < \epsilon$ **for all** $1 \leq i, j \leq N$ **and** $T \in \mathbb{R}_{\geq 0}$

An *event* of a given signal x_i is a pair $(t, x_i(t))$, where t is a local clock value. We denote by V_i the set of events of signal x_i. An *edge* of x_i is an event $(t, x_i(t))$ such that $\lim_{s \to t^-} x_i(s) \neq \lim_{s \to t^+} x_i(s)$. In particular, it is a *rising* edge if $\lim_{s \to t^-} x_i(s) < \lim_{s \to t^+} x_i(s)$, and a *falling* edge otherwise. We respectively denote by $E_i^{\uparrow}$ and $E_i^{\downarrow}$ the sets of rising and falling edges of x_i, and we let $E_i = E_i^{\uparrow} \cup E_i^{\downarrow}$.

We assume that signals have *bounded variability* with respect to the global clock: for each signal x_i and every pair $(t, x_i(t)), (t', x_i(t')) \in E_i$ of edges, we have $|c_j^{-1}(t) - c_j^{-1}(t')| \geq \delta$ **Anik: for all** $1 \leq j \leq N$, where $A_j = \pi(x_i)$ is the agent that produces x_i and $\delta \in \mathbb{R}_{\geq 0}$ is the bounded variability constant. **Borzoo: Do we really need the agent here?**

Definition 2. *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* $1 \leq i, j \leq n$ **Anik: Use either 'n' or 'N' consistently** *with* $\pi(x_i) = \pi(x_j)$ *and all* $(t, x_i(t)) \in V_i$ *and* $(t', x_j(t')) \in V_j$*, if* $t < t'$ *then* $(t, x_i(t)) \rightsquigarrow (t', x_j(t'))$*.* **Borzoo: I do think** π **is unnecessary. We can assume each agent produces a distinct signal.**
- *Every pair of events whose timestamps are at least* 2ε *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* $t + 2\varepsilon \leq t'$ *then* $(t, x_i(t)) \rightsquigarrow (t', x_j(t'))$*.*

Example 3. TODO: distributed signal, happened-before relation

Definition 4. *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 5. *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 $1 \leq i \leq n$ where $A_j = \pi(x_i)$, if $T < T'$, then for every pair of events $(c_j(T), x_i(c_j(T))) \in \mathsf{ccf}(T)$ and $(c_j(T'), x_i(c_j(T'))) \in \mathsf{ccf}(T')$ we have $(c_j(T), x_i(c_j(T))) \rightsquigarrow (c_j(T'), x_i(c_j(T')))$. We denote by $\mathsf{CCF}(S, \rightsquigarrow)$ the set of all consistent cut flows of the distributed signal $(S, \rightsquigarrow)$.

Example 6. TODO: consistent cut, frontier, consistent cut flow

1.2 Signal Temporal Logic

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* is a vector of synchronous signals of finite duration. 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. Note that given a trace over a temporal domain $[0, D)$ and intervals $I, J \subseteq \mathbb{R}_{\geq 0}$, we write $I \oplus J = \{i + j \mid i \in I \wedge j \in J\}$. We use the shorthand notation T for the singleton set $\{T\}$. **Dejan: We are talking both about $x_i(t)$ and $x_i(T)$ and these two things have a different meaning. I do not know whether we should say something more about it. $x_i(t)$ is the trace observation on the agent A_i and its local clock, while $x_i(T)$ is the (unknown) real value of the signal wrt the global time. Ege: I try to use $x_i(T)$ only when the signals are synchronous (i.e., for talking about traces). When there is asynchrony, I write $x_i(c_j(T)))$ where A_j is the agent producing x_i.**

Below we recall the finite-trace qualitative semantics of STL defined over $\mathbb{B}$. 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 $T \in [0, D)$.

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

We simply write $[w \models \varphi]$ for $[w, 0 \models \varphi]$. We additionally use the following standard abbreviations: $\mathtt{false} = p \wedge \neg p$, $\mathtt{true} = \neg\mathtt{false}$, $\varphi_1 \vee \varphi_2 = \neg(\neg\varphi_1 \wedge \neg\varphi_2)$, $\Diamond_I \varphi = \mathtt{true} \mathcal{U}_I \varphi$, 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., $\varphi_1 \mathcal{U} \varphi_2 = \varphi_1 \mathcal{U}_{(0,\infty)} \varphi_2$. Note that our interpretation of the untimed until operator is strict. The non-strict variant can be defined in terms of the strict one as follows: $\varphi_1 \underline{\mathcal{U}} \varphi_2 = \varphi_2 \vee (\varphi_1 \wedge (\varphi_1 \mathcal{U} \varphi_2))$.

2 Problem Statement

Let $(S, \rightsquigarrow)$ be a distributed signal of n signals $x_1, \ldots, x_n$ with the temporal domain $[0, d)$ where $d \in \mathbb{R}_{>0}$. 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 $[0, D_{\mathsf{ccf}})$ where $D_{\mathsf{ccf}} \in \mathbb{R}_{>0}$ such that $(c_j(T), x_i(c_j(T))) \in \mathsf{ccf}(T)$ implies $x'_i(T) = x_i(c_j(T))$ for all $1 \leq i \leq n$ and $T \in [0, D_{\mathsf{ccf}})$, where $A_j = \pi(x_i)$. The set of traces of $(S, \rightsquigarrow)$ is given by $\mathsf{Tr}(S, \rightsquigarrow) = \{w_{\mathsf{ccf}} \mid \mathsf{ccf} \in \mathsf{CCF}(S, \rightsquigarrow)\}$. Moreover, we write $\mathsf{Tr}(S, \rightsquigarrow, \delta)$ for the largest subset of $\mathsf{Tr}(S, \rightsquigarrow)$ where all signals of all traces satisfy the bounded variability assumption with respect to the constant δ.

We are interested in how the satisfaction signals of the traces of $(S, \rightsquigarrow)$ evolve over time. Let φ be an STL formula and $w \in \mathsf{Tr}(S, \rightsquigarrow)$ be a trace of length D. We denote by $y_w^\varphi : [0, D) \to \{0, 1\}$ the signal defined as $y_w^\varphi(t) = 1$ if $[w, t \models \varphi]_{\mathsf{STL}} = \top$ and $y_w^\varphi(t) = 0$ if $[w, t \models \varphi]_{\mathsf{STL}} = \bot$ for all $t \in [0, D)$. Given a signal x and the set $E_x = \{(t_1, x(t_1)), \ldots, (t_k, x(t_k))\}$ of its edges, we denote by $\mu(x)$ the sequence of values of its edges, i.e, $\mu(x) = v_0 v_1 \ldots v_k$ where $v_0 = x(0)$ and $v_i = x(t_i)$ for all $1 \leq i \leq k$.

Given a distributed signal $(S, \rightsquigarrow)$, an STL formula φ, a maximum clock skew $\varepsilon > 0$, and a bounded variability constant $\delta > 0$, we aim to compute $\mathsf{Goal}(\varphi, S, \rightsquigarrow, \varepsilon, \delta) = \{\mu(y_w^\varphi) \mid w \in \mathsf{Tr}(S, \rightsquigarrow, \delta)\}$, i.e., the set of nonstuttering sequences of truth values obtained from evaluating φ on the traces of $(S, \rightsquigarrow)$ using the finite-trace semantics of STL.

3 Our Approach

A precise solution to the monitoring problem at hand would include considering all interleavings of the concurrent events in the given distributed signal. The partial synchrony assumption helps reduce the number of such interleavings, however, the state-of-the-art leveraging this assumption relies on off-the-shelf SMT solvers and thus are not able to provide theoretical guarantees on running time and memory consumption [1,3]. In order to design a dedicated monitoring algorithm with provable guarantees **Borzoo: Do we do that?!**, we introduce the bounded variability assumption and a method to overapproximate the set of behaviors of the distributed signal. We assume a central monitor that knows the values of ε and δ as well as the initial values of the signals.

3.1 Uncertainty Regions and Canonical Segmentations

Consider a signal $x : [0, d) \to \mathbb{R}$ with a single rising edge from value 2 to 3 at local time t. The monitor observing this signal needs to take into account

how the local clock of the agent producing x relates with the global clock: 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 2 to 3. To systematically reason about uncertainty regions of signals, 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. **Borzoo: Are these consecutive intervals?**

Let $x : [0, d) \to \mathbb{R}$ be a signal and $E_x = \{(t_1, x(t_1)), \ldots, (t_m, x(t_m))\}$ be the set of edges of x, given in an increasing order of local clock values. For $1 \leq i \leq m$, we let

$$\theta_{\text{lo}}(x, t_i) = \max_{1 \leq j \leq i} t_j - \varepsilon + (i-j)\delta, \text{ and}$$
$$\theta_{\text{hi}}(x, t_i) = \min_{i \leq j \leq m} t_j + \varepsilon - (j-i)\delta.$$

Intuitively, θ_{lo} and θ_{hi} give us the lower and upper bounds on the value of the global clock for a given edge, i.e., the bounds on the uncertainty region of the 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 $F_i = \{\theta_{\text{lo}}(x_i, t_j) \mid (t_j, x_i(t_j)) \in E_i\} \cup \{\theta_{\text{hi}}(x_i, t_j) \mid (t_j, x_i(t_j)) \in E_i\}$ be the set of bounds on 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 global clock.**Borzoo: why global clock?** We define $F_S = \{0, d'\} \cup \bigcup_{i=1}^{n} F_i$ and let $(a_j)_{1 \leq j \leq |F|}$ be an increasing sequence of clock values corresponding to the elements of F. We define the *canonical segmentation* of $(S, \rightsquigarrow)$ as $G_S = \{I_1, \ldots, I_{|F|-1}\}$ where $I_j = [a_j, a_{j+1})$ for all $1 \leq j < |F|$.

Example 7. Let $(S, \rightsquigarrow)$ be a distributed signal with $S = (x_1, x_2)$ over the temporal domain $[0, 8)$ such that $\pi(x_1) \neq \pi(x_2)$.**Borzoo: π is unnecessary.** Suppose $x_1(0) = 2$ and $x_2(0) = 3$, and let $\varepsilon = 2$ and $\delta = 3$. Consider the case where x_1 has a rising edge $(3, 5)$ and a falling edge $(5, 3)$ while x_2 has a rising edge $(2, 7)$ and a falling edge $(5, 2)$. For x_1, taking into account only the clock skew would give us the uncertainty regions $(1, 5)$ and $(3, 7)$.**Borzoo: We should this by a figure.** However, the computation of uncertainty regions take into account also the bounded variability. Intuitively, if the rising edge of x_1 occurs at global time 1, considering bounded variability, its falling edge occurs at the earliest at global time 4 instead of 3. Conversely, if the falling edge occurs at global time 7 then its rising edge occurs at the latest at global time 4 instead of 5. Then, we obtain $F_1 = \{1, 4, 7\}$. For x_2, the uncertainty regions are $(0, 4)$ and $(3, 7)$, which gives us $F_2 = \{0, 3, 4, 7\}$; and therefore, $F_S = \{0, 1, 3, 4, 7, 8\}$. This leads to the canonical segmentation $G_S = \{[0, 1), [1, 3), [3, 4), [4, 7), [7, 8)\}$.

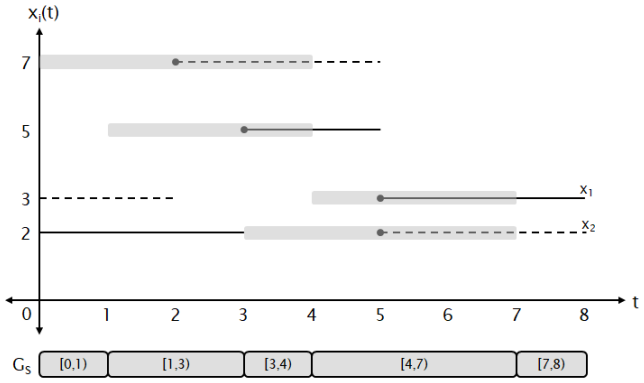

Fig. 1. The values of x_1 (solid) and x_2 (dashed) over time. The edges are marked with black balls and their uncertainty regions are given as light gray boxes around the edges. The resulting canonical segmentation G_S is shown below the graphical representation of the signals.

3.2 Value Expressions

In uncertainty regions, although the monitor is not able to determine the exact value of a signal at a time point, it knows how the signal values change. To capture this information, we use short sequences of possible signal values capturing how a signal behaves within a segment and several operations on them to propagate the information from real-valued signals to truth values of subformulas.

Let Σ be a finite alphabet corresponding to a set of signal values. A *value expression* is a finite word in $\mathsf{destutter}(\Sigma^*)$. Given value expressions u_1 and u_2, we define their *asynchronous product* to encode their potential interleavings as follows:

$$u_1 \otimes u_2 = \{\mathsf{destutter}(v_1, v_2) \mid v_i \in \mathsf{stutter}_k(u_i), \text{ where } k = |u_1| + |u_2| - 1\}$$

Given $L_1, L_2 \subseteq \mathsf{destutter}(\Sigma^*)$, we write $L_1 \otimes L_2 = \{u_1 \otimes u_2 \mid u_i \in L_i\}$.

Borzoo: The definition if destutter is a unary function. The above is a binary function. Also, I think you should add $i \in \{1, 2\}$ For value expressions u_1, u_2 and a function $f : \Sigma^2 \to \mathbb{R}$, we refer to the set of value expressions that can be obtained by applying f to their asynchronous product as follows: $f(u_1, u_2) = \mathsf{destutter}(\{(f(v_1[i], v_2[i]))_{1 \le i \le k} \mid (v_1, v_2) \in u_1 \otimes u_2 \text{ and } k = |v_1|\})$. Moreover, for $c \in \mathbb{R}$, we write $f(u_1, u_2) > c$ to denote the set $\mathsf{destutter}(\{(f(v_1[i], v_2[i]) > c)_{1 \le i \le k} \mid (v_1, v_2) \in u_1 \otimes u_2 \text{ and } k = |v_1|\})$ where, for each i, the value of $f(v_1[i], v_2[i]) > c$ is 1 if the inequality holds, and it is 0 otherwise. We naturally extend these definitions to functions of arbitrary (finite) arity.

Example 8. Recall the distributed signal $(S, \rightsquigarrow)$ in Example 7. To encode the behavior of x_1 and x_2, let $\Sigma = \{2, 3, 5, 7\}$. One possible behavior of these signals is that the rising edge of x_1 and the falling edge of x_2 occur in the segment

$[3,4)$. We represent this case by taking the value expression $u_1 = 2 \cdot 5$ for x_1 and $u_2 = 7 \cdot 2$ for x_2 in the segment $[3,4)$. To capture the potential interleavings of these edges, we compute the asynchronous product $u_1 \otimes u_2$ of these value expressions. Stuttering u_1 and u_2 up to length 3 and taking their product gives us $(2 \cdot 2 \cdot 5, 7 \cdot 7 \cdot 2)$, $(2 \cdot 2 \cdot 5, 7 \cdot 2 \cdot 2)$, $(2 \cdot 5 \cdot 5, 7 \cdot 7 \cdot 2)$, and $(2 \cdot 5 \cdot 5, 7 \cdot 2 \cdot 2)$. Observe that the first and the last pairs are equivalent after destuttering and they encode the case when the rising and falling edges occur exactly at the same time with respect to the global clock. On the other hand, the second (resp. third) pair encodes the case when the falling edge of x_2 occurs before (resp. after) the rising edge of x_1.

Now, to compute the function $u_2 - u_1$ we apply the function letter by letter to the pairs in their asynchronous product. For example, applying it to $(2 \cdot 5, 7 \cdot 2)$ results in $(7-2) \cdot (2-5) = 5 \cdot -3$. Similarly, we get $5 \cdot 0 \cdot -3$ and $5 \cdot 2 \cdot -3$ from the remaining pairs. Finally, if we take the constraint $u_2 - u_1 > 0$, all the expressions coincide after destuttering and result in $1 \cdot 0$.

Borzoo: I think we can just skip all the above and start with $\Sigma = \{0,1\}$. People understand how we respresent arithmetic predicates with Booleans. No? As we will detail in the next section, after computing the value expressions of real-valued signals, we compute those of satisfaction signals of atomic propositions and use these boolean value expressions in the evaluation of the STL formula. Therefore, we assume a binary alphabet in the sequel.

Let $\Sigma = \{0,1\}$. Given $u, v \in \Sigma^*$ of the same length, we respectively denote by $u \,\&\, v$ the bitwise-and operation, and by $\sim u$ the bitwise-negation operation. In addition, given nonempty words $u, v \in \Sigma^*$ of length ℓ, we define auxiliary bitwise-temporal operators as follows:

$$
\begin{aligned}
\mathsf{E}u &= \left(\max_{i \le j \le \ell} u[j] \right)_{1 \le i \le \ell} \\
\mathsf{A}u &= \left(\min_{i \le j \le \ell} u[j] \right)_{1 \le i \le \ell} \\
u \mathsf{U}^0 v &= \left(\max_{i \le j \le \ell} \left(\min \left(v[j], \min_{i \le k \le j} u[k] \right) \right) \right)_{1 \le i \le \ell} \\
u \mathsf{U}^1 v &= \left(\max \left(u[i..] \mathsf{U}^0 v[i..], \mathsf{A}u[i..] \right) \right)_{1 \le i \le \ell}
\end{aligned}
$$

Note that the operators E and A respectively correspond to the *eventually* and the *always* operators of temporal logics. **Borzoo: This is pretty much like robustness semantics of STL. Should we go with that?** Moreover, U^0 corresponds to the standard (strong) *until* operator while U^1 is the *weak until*. As usual, $\mathsf{A}u = u \mathsf{U}^1 0^\ell$ and $\mathsf{E}u = 1^\ell \mathsf{U}^0 u$. This distinction will be useful later when we evaluate value expressions segment by segment.

A single boolean value expression encodes how a satisfaction signal evolves through an interval. We use sets of such expressions, i.e., subsets of $\mathsf{destutter}(\Sigma^*)$,

to represent the potential behaviors of satisfaction signals arising from different interleavings of events. The following grammar encodes these sets.

$$\psi := 0 \mid 1 \mid \neg\psi \mid \psi \sqcap \psi \mid \psi \cdot \psi \mid \psi \mathcal{U}^0 \psi \mid \psi \mathcal{U}^1 \psi$$

We denote by Ψ the set of all expressions generated by the above grammar. Letting $\psi_1, \psi_2 \in \Psi$, we inductively define the semantics for our encoding.

$$\begin{aligned}
[\![0]\!] &= \{0\} \\
[\![1]\!] &= \{1\} \\
[\![\neg\psi_1]\!] &= \{\sim u \mid u \in [\![\psi_1]\!]\} \\
[\![\psi_1 \sqcap \psi_2]\!] &= \mathsf{destutter}(\{v_1 \,\&\, v_2 \mid (v_1, v_2) \in [\![\psi_1]\!] \otimes [\![\psi_2]\!]\}) \\
[\![\psi_1 \cdot \psi_2]\!] &= \mathsf{destutter}(\{u_1 u_2 \mid u_1 \in [\![\psi_1]\!], u_2 \in [\![\psi_2]\!]\}) \\
[\![\psi_1 \mathcal{U}^0 \psi_2]\!] &= \mathsf{destutter}(\{v_1 \mathsf{U}^0 v_2 \mid (v_1, v_2) \in [\![\psi_1]\!] \otimes [\![\psi_2]\!]\}) \\
[\![\psi_1 \mathcal{U}^1 \psi_2]\!] &= \mathsf{destutter}(\{v_1 \mathsf{U}^1 v_2 \mid (v_1, v_2) \in [\![\psi_1]\!] \otimes [\![\psi_2]\!]\})
\end{aligned}$$

We additionally define $\psi_1 \sqcup \psi_2 = \neg(\neg\psi_1 \sqcap \neg\psi_2)$, $\Diamond\psi = 1\mathcal{U}^0\psi$, and $\Box\psi = \psi\mathcal{U}^1 0$, and Moreover, one can easily verify that $\mathsf{destutter}(\Sigma^*)$ is closed under these operations.

Example 9. **Borzoo: I think we can again use a fig to explain this example.** Recall the distributed signal $(S, \rightsquigarrow)$ in Example 7. Using the propositions $p = (x_1 > 3)$ and $q = (x_2 > 6)$, we show how one computes $\mathcal{U}^0$ and $\mathcal{U}^1$. In Example 8, we focused on the scenario in which the rising edge of x_1 and the falling edge of x_2 occurs in the segment $[3, 4)$. In this scenario, we represent the behavior of the satisfaction signal of p in the segment $[3, 4)$ by the value expression 01, and that of q by 10. For p, the only other option is that the rising edge of x_1 occurs earlier, which leads to the value expression 1. For q, the alternatives include the expressions 01 and 1, respectively encoding the scenarios in which the rising edge of x_2 occurs in $[3, 4)$, and no edges of x_2 occur in $[3, 4)$. Let us represent these potential behaviors by the sets $L_p = \{01, 1\}$ and $L_q = \{10, 01, 1\}$.

To compute $L_p\mathcal{U}^0 L_q$ and $L_p\mathcal{U}^1 L_q$, we compute the asynchronous product $L_p \otimes L_q$ and apply the corresponding bitwise-until operators on each pair in the product. Note that the bitwise-until operators can be applied bit by bit, in the same spirit as the standard recursive definition $p\mathcal{U}q = q \vee (p \wedge \bigcirc(p\mathcal{U}q))$ for LTL, where $\mathcal{U}^0$ assumes the value of the undefined next bit is 0 whereas $\mathcal{U}^1$ assumes it is 1. To demonstrate, let us consider the pair $(11, 10) \in L_p \otimes L_q$, encoding the case where the rising edge of x_1 occurs before the segment $[3, 4)$ and the falling edge of x_2 occurs in $[3, 4)$. Then, we have (i) $11\mathsf{U}^0 10 = 10$, and (ii) $11\mathsf{U}^1 10 = 11$ which gives us 1 after destuttering. These expressions respectively correspond to the behaviors of the satisfaction signal of $p\mathcal{U}q$ in $[3, 4)$ in this scenario where the next segment starts with a behavior that (i) violates the formula, and (ii) satisfies it. Repeating these for each pair in the asynchronous product, we obtain $L_p\mathcal{U}^0 L_q = \{1, 01, 10\}$ and $L_p\mathcal{U}^1 L_q = \{1, 01, 101\}$.

3.3 Approximate Evaluation

Given a distributed signal $(S, \rightsquigarrow)$, our logic maps a segmentation of $(S, \rightsquigarrow)$ to a set of value expressions. Using these, first we compute the value expressions for atomic propositions, and then inductively evaluate the subformulas. We assume that, at any segment, the signals can realize any value expression of the segment. In particular, we treat any combination of value expressions from consecutive segments as a valid encoding of a potential behavior. This leads to an overapproximation of the set of possible behaviors since we lose the relationship between value expressions with respect to the signals' edges.

Example 10. Recall the distributed signal $(S, \rightsquigarrow)$ and its canonical segmentation G_S in Example 7. The signal x_1 has a rising edge with an uncertainty interval of $(1, 4)$, and the canonical segmentation contains the segments $[1, 3)$ and $[3, 4)$, effectively splitting the uncertainty interval of this edge. To cover all possible behaviors of the distributed signal, we allow the encoding of the rising edge of x_1 both in the segment $[1, 3)$ and in $[3, 4)$. This results in the value expression $2 \cdot 5$ being contained in both the set of value expressions of x_1 for $[1, 3)$ and for $[3, 4)$. However, when we reason on possible the behaviors of x_1, we simply take the product of the sets of possible behaviors for each segment and neglect the knowledge that x_1 has a single rising edge changing its value from 2 to 5. In particular, our method considers the value expression $2 \cdot 5 \cdot 2 \cdot 5$ as a valid potential behavior of x_1 in the interval $[1, 4)$ although this is not possible. While each of these value expressions are valid in isolation (i.e., x_1 can display the behavior 2 in $[1, 3)$ and $2 \cdot 5$ in $[3, 4)$, or $2 \cdot 5$ in $[1, 3)$ and 5 in $[3, 4)$), allowing this combination (i.e., $2 \cdot 5 \cdot 2 \cdot 5$) results in an overapproximation by encoding a behavior that is not a potential behavior of the given signal. **Borzoo: I think the past couple of pages can be summarized just by this example.**

Approximate Evaluation of Real-Valued Signals Let $(S, \rightsquigarrow)$ be a distributed signal of n signals $x_1, \dots x_n$, and let G_S be its canonical segmentation. We want to compute the set of value expressions of each signal in each segment in the canonical segmentation G_S. We denote by $\gamma(I, i)$ the value expression capturing the behavior of x_i in the interval $I \in G_S$ and describe its computation below.

For each x_i, let $X_i = \{J_{i,j} \mid 1 \leq j < |E_i|\}$ be the set of its uncertainty regions, where each $J_{i,j} = [l_{i,j}, h_{i,j})$ is such that $l_{i,j} = \theta_{\text{lo}}(x_i, t_j)$ and $h_{i,j} = \theta_{\text{hi}}(x_i, t_j)$, and t_j is the jth edge of x_i in increasing order of local clock values. These intervals have the value expression $v_{i,j} = \nu_1\nu_2$ where $\nu_1 = \lim_{s \to t_j^-} x_i(s)$ and $\nu_2 = \lim_{s \to t_j^+} x_i(s)$, and the remaining intervals have constant value expressions.

First, we "synchronize" the uncertainty regions of the signals in S by splitting their value expressions accordingly. Let $G_S = \{I_1, \dots, I_K\}$ where $I_k = [s_k, s_{k+1})$ for all $1 \leq k \leq K$. For each $1 \leq i \leq n$ and $1 \leq k \leq K$, let $Y_{k,i} = \{J_{i,j} \in X_i \mid I_k \subseteq J_{i,j}\}$ be the uncertainty regions of x_i that contain the interval I_k where $J_{i,j}$ are as above. Let $M = |Y_{k,i}|$. Now, we compute the value expressions of signals for the canonical segmentation.

- For each $1 \leq i \leq n$ in increasing order and for each $1 \leq k \leq K$ in increasing order, compute the set $Z_{k,i} = \mathrm{op}_1(v_{i,1}) \cdot \ldots \cdot \mathrm{op}_M(v_{i,M})$ where, for $1 \leq j \leq M$,
  - if $l_{i,j} = s_k$ and $s_{k+1} = h_{i,j}$ then $\mathrm{op}_j(v_{i,j}) = \{v_{i,j}\}$,
  - if $l_{i,j} = s_k$ and $s_{k+1} < h_{i,j}$ then $\mathrm{op}_j(v_{i,j}) = \mathsf{prefix}(v_{i,j})$,
  - if $l_{i,j} < s_k$ and $s_{k+1} = h_{i,j}$ then $\mathrm{op}_j(v_{i,j}) = \mathsf{suffix}(v_{i,j})$,
  - if $l_{i,j} < s_k$ and $s_{k+1} < h_{i,j}$ then $\mathrm{op}_j(v_{i,j}) = \mathsf{infix}(v_{i,j})$,
  - if $j \geq 2$, then $\mathrm{op}_j(v_{i,j}) = \mathrm{op}_j(v_{i,j}) \cup \{\epsilon\}$.
- Let $\gamma(I_k, i) = \mathsf{destutter}(Z_{k,i})$.

Example 11. Recall the distributed signal in Example 7. Initially, before we compute γ for each segment and signal, the signals have the value expressions as demonstrated in Figure 2a. Considering the canonical segmentation, the computed values of γ are as in Figure 2b. For example, the value $\gamma([1,3),1) = \mathsf{destutter}(\mathsf{prefix}(2 \cdot 5)) = \{2, 2 \cdot 5\}$ because $[1,3)$ is a subset of the uncertainty interval $[1,4)$ of x_1 and only their left end points coincide. Similarly, $\gamma([3,4),2) = \mathsf{destutter}(\mathsf{suffix}(3 \cdot 7) \cdot (\mathsf{prefix}(7 \cdot 2) \cup \{\epsilon\})) = \{3 \cdot 7, 3 \cdot 7 \cdot 2, 7, 7 \cdot 2\}$ because the right (resp. left) end points of $[3,4)$ and the uncertainty interval $[0,4)$ (resp. $[3,7)$) of x_2 coincide.

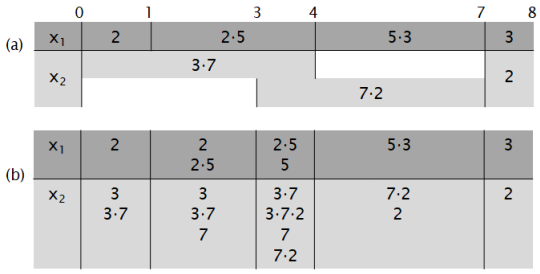

Fig. 2. (a) Uncertainty regions and their value expressions before the segmentation. (b) The canonical segmentation of the signal and the corresponding value expressions.

Approximate Evaluation of Satisfaction Signals Let $\tau_S : [0,d) \to G_S$ be a function that maps each clock value to the interval in the canonical segmentation that it belongs to, i.e., $\tau_S(t) = [t', t'')$ where $t' \leq t < t''$ and $[t', t'') \in G_S$. We say that two intervals $I, J \subseteq [0,d)$ with the end points $a_i < b_i$ and $a_j < b_j$ are of the *same type* iff the following holds: $(a_i \in I \iff a_j \in J) \wedge (b_i \in I \iff b_j \in J)$.

Let $(S, \rightsquigarrow)$ be a distributed signal of n signals, $t \in [0,d)$ be a time value such that $\tau_S(t) = [t', t'')$. Let $I \subseteq \mathbb{R}_{\geq 0}$ be a time interval with the end points $a < b$. Note that timed until can be expressed using timed eventually, timed always, and untimed until operators [2]. We provide the semantics of our logic following this observation.

$$
\begin{aligned}
[S,t \models p] &= f_p(\gamma(\tau_S(t),1),\ldots,\gamma(\tau_S(t),n)) > 0\\
[S,t \models \neg\varphi] &= \neg[S,t \models \varphi]\\
[S,t \models \varphi_1 \wedge \varphi_2] &= [S,t \models \varphi_1] \sqcap [S,t \models \varphi_2]\\
[S,t \models \varphi_1 \mathcal{U} \varphi_2] &= \mathsf{destutter}(\{[S,t \models \varphi_1]\mathcal{U}^b[S,t \models \varphi_2] \mid \text{ if } t'' < d \text{ then } b \in \mathsf{first}([S,t'' \models \varphi_1\mathcal{U}\varphi_2]) \text{ else } b = 0\})\\
[S,t \models \Diamond_I \varphi] &= \mathsf{destutter}(\{\mathsf{E}(Pr_1)\cdot\ldots\cdot\mathsf{E}(Pr_M)\}) \text{ where } Pr_i \in \mathsf{profilesBetween}(I,S,\varphi,[t',t'')) \text{ for all } 1 \le i \le M\\
[S,t \models \Box_I \varphi] &= \mathsf{destutter}(\{\mathsf{A}(Pr_1)\cdot\ldots\cdot\mathsf{A}(Pr_M)\}) \text{ where } Pr_i \in \mathsf{profilesBetween}(I,S,\varphi,[t',t'')) \text{ for all } 1 \le i \le M
\end{aligned}
$$

The *profile* of an interval I with respect to S and φ captures the value expressions for the satisfaction of φ in I. Then, $\mathsf{profilesBetween}(I,S,\varphi,J)$ is the set of profiles, with respect to S and φ, of intervals of the same length and type as I that coincide with the segments in $I \oplus J$. Intuitively, we aim to compute how the satisfaction signal of φ behaves in I while we slide I over $I \oplus J$. Observe that, although there are infinitely many such windows due to denseness of time, there are only finitely many ways these windows can "relate" with the segments in $I \oplus J$, resulting in finitely many profiles for them. We formalize and demonstrate this below.

Let $I \subseteq [0,d)$ be an interval with the end points $a < b$. Given $t \in [0,d)$, we let $\mathsf{end}(t,S)$ be true iff $t = d$ or $\tau_S(t) = [t,t')$ for some $t' > t$. Let Q be the number of intervals in G_S whose endpoints are strictly between a and b. Let $(H_k)_{1\le k\le Q}$ be a (possibly empty) sequence of intervals of the canonical segmentation whose endpoints are strictly between a and b, i.e., $H_k = [d_k, d_{k+1}) \in G_S$ and $a < d_k < d_{k+1} < b$ for all $1 \le i \le Q$. We denote by $\kappa(S,a,b)$ the concatenation of the value expressions of φ for the intervals in this sequence, i.e., $\kappa(S,a,b) = [S,s_1 \models \varphi]\cdot\ldots\cdot[S,s_Q \models \varphi]$. We define the *profile* of $I \subseteq [0,d)$ with the end points $a < b$ with respect to S and φ as follows.

$$
\mathsf{profile}(I,S,\varphi) = \begin{cases}
\mathsf{prefix}([S,a \models \varphi]) & \text{if } \tau_S(a) = \tau_S(b) \wedge \mathsf{end}(a,S)\\
\mathsf{infix}([S,a \models \varphi]) & \text{if } \tau_S(a) = \tau_S(b) \wedge \neg\mathsf{end}(a,S)\\
[S,a \models \varphi]\cdot\kappa(S,a,b)\cdot\mathsf{first}([S,b \models \varphi]) & \text{if } \tau_S(a) \neq \tau_S(b) \wedge \mathsf{end}(a,S) \wedge \mathsf{end}(b,S) \wedge b \in I\\
[S,a \models \varphi]\cdot\kappa(S,a,b) & \text{if } \tau_S(a) \neq \tau_S(b) \wedge \mathsf{end}(a,S) \wedge \mathsf{end}(b,S) \wedge b \notin I\\
[S,a \models \varphi]\cdot\kappa(S,a,b)\cdot\mathsf{prefix}([S,a \models \varphi]) & \text{if } \tau_S(a) \neq \tau_S(b) \wedge \mathsf{end}(a,S) \wedge \neg\mathsf{end}(b,S)\\
\mathsf{suffix}([S,a \models \varphi])\cdot\kappa(S,a,b)\cdot\mathsf{first}([S,b \models \varphi]) & \text{if } \tau_S(a) \neq \tau_S(b) \wedge \neg\mathsf{end}(a,S) \wedge \mathsf{end}(b,S) \wedge b \in I\\
\mathsf{suffix}([S,a \models \varphi])\cdot\kappa(S,a,b) & \text{if } \tau_S(a) \neq \tau_S(b) \wedge \neg\mathsf{end}(a,S) \wedge \mathsf{end}(b,S) \wedge b \notin I\\
\mathsf{suffix}([S,a \models \varphi])\cdot\kappa(S,a,b)\cdot\mathsf{prefix}([S,b \models \varphi]) & \text{if } \tau_S(a) \neq \tau_S(b) \wedge \neg\mathsf{end}(a,S) \wedge \neg\mathsf{end}(b,S)
\end{cases}
$$

Example 12. Recall the distributed signal in Example 7. Let $p = (x_1 > 3)$ and $q = (x_2 > 6)$ be atomic propositions as in Example 9.

First, we consider the STL formula $p\mathcal{U}q$. The value expressions of the satisfaction signals of p and q are respectively computed by bitwise comparison of the value expressions of x_1 with 3 and x_2 with 6. The resulting expressions for q are given in Figure 3. We compute the values of $p\mathcal{U}q$ starting from the last segment $[7,8)$, which gives us $0\mathsf{U}^0 0 = 0$. After handling the segment $[4,7)$ and obtain $\{10\mathsf{U}^0 10, 10\mathsf{U}^0 0\} = \{10,0\}$, we move to the segment $[3,4)$. Note that $\mathsf{first}(\{10,0\}) = \{1,0\}$, therefore we compute the union of $\{01,1\}\mathcal{U}^0\{01,010,1,10\}$ and $\{01,1\}\mathcal{U}^1\{01,010,1,10\}$ for the segment $[3,4)$, and continue similarly with $[1,3)$ and $[0,1)$.

Now, we consider the STL formula $\Diamond_{[0,1.5]}q$. We show the evaluation of the value expressions for the segment $[1,3)$. Intuitively, to compute $\Diamond_{[0,1.5]}q$ in the segment $[1,3)$, one can consider a sliding window over the interval $[1,4.5)$ and compute how the behavior of the satisfaction signal of q changes as the window slides.

First, we need to compute the set $\mathsf{profilesBetween}([0,1.5],(x_1,x_2),p,[1,4.5))$. We show in Figure 3 the profiles corresponding to how intervals of the same length and type as $[0,1.5]$ can intersect with the segments covering $[0,1.5]\oplus[1,3)$.

Let us denote by R_t the set of value expression of the segment starting at time t, given in Figure 3. Then, the six intervals in Figure 3 correspond to the following profiles in order:

1. $\mathsf{Pr}_1 = \mathsf{prefix}(R_1)$
2. $\mathsf{Pr}_2 = \mathsf{infix}(R_1)$
3. $\mathsf{Pr}_3 = \mathsf{suffix}(R_1) \cdot \mathsf{first}(R_3)$
4. $\mathsf{Pr}_4 = \mathsf{suffix}(R_1) \cdot \mathsf{prefix}(R_3)$
5. $\mathsf{Pr}_5 = \mathsf{suffix}(R_1) \cdot R_3 \cdot \mathsf{first}(R_4)$
6. $\mathsf{Pr}_6 = \mathsf{suffix}(R_1) \cdot R_3 \cdot \mathsf{prefix}(R_4)$

Recall the sliding window analogy: One can imagine that we obtain these profiles by sliding a closed window of length 1.5 over the segment $[1,3)$. Although there are infinitely many such windows, there are finitely many profiles; for example, $[1.2,2.7]$ and $[1.4,2.9]$ are equivalent for our purposes. This is because they are both strictly contained in $[1,3)$, meaning that they capture the events of q that may occur strictly between the interval's endpoints. More specifically, they are captured by Pr_2, as with any $t \in (1,1.5)$. Similarly, the profiles Pr_1 and Pr_3 capture respectively $t = 1$ and $t = 1.5$; the profile Pr_4 captures $t \in (1.5,2.5)$; the profile Pr_5 captures $t = 2.5$; and the profile Pr_6 captures $t \in (2.5,3)$.

Applying the bitwise-eventually operator E to each profile, concatenating the resulting sets, and finally destuttering gives us the set of value expressions for $\Diamond_{[0,1.5]}q$ in the segment $[1,3)$. For example, the value expression 010101010 is in this set because the first two profiles each contain 0 and the last four profiles each contain an expression equivalent to 10 after destuttering.

We remark that this example also shows how our method overapproximates the set of possible behaviors by neglecting the bookkeeping of edges. For example, in reality, if x_2 has its rising edge in $[1,3)$, then it cannot have its falling edge in $[3,4)$ due to bounded variability constant $\delta = 3$.

4 The Offline Algorithm

Let φ be an STL formula and $(S,\rightsquigarrow)$ be a distributed signal of n signals $x_1,\dots x_n$ over the temporal domain $[0,d)$.

1. Enumerate the subformulas of φ such that each formula has an enumeration number smaller than the numbers of all its subformulas. Let $\varphi_1,\dots,\varphi_m$ be such an enumeration. Note that $\varphi_1 = \varphi$.

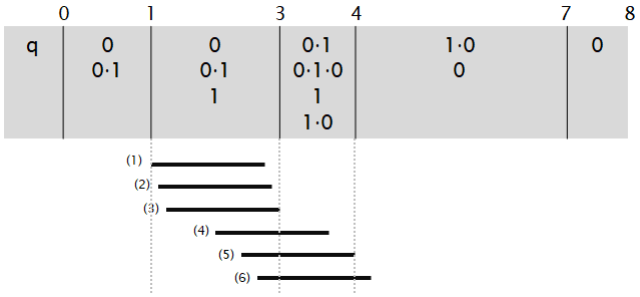

Fig. 3. The intervals representing the profiles needed to compute $\Diamond_{[0,1.5]} q$ in $[1, 3)$. The six black lines at the bottom show how the interval $[0, 1.5]$ "slides" over $[1, 3)$.

2. Compute the canonical segmentation $G_S = \{[t_1, t_2), \ldots, [t_k, t_{k+1})\}$ of $(S, \rightsquigarrow)$.
3. Compute the value expressions of signals with respect to the canonical segmentation, i.e., for each $1 \leq j \leq k$ and $1 \leq i \leq n$, compute $\gamma([t_j, t_{j+1}), i)$.
4. Compute the value expressions of subformulas with respect to the canonical segmentation, i.e., for each $1 \leq j \leq k$ and $1 \leq \ell \leq m$, compute $[S, t_j \models \varphi_\ell]$.
5. Output the set $\mathsf{Out}(\varphi, S, \rightsquigarrow, \varepsilon, \delta) = \mathsf{destutter}([S, t_1 \models \varphi_1] \cdot \ldots \cdot [S, t_k \models \varphi_1])$ of value expressions.

Theorem 13. *For all STL formulas φ and distributed signals $(S, \rightsquigarrow)$ of bounded clock skew ε and bounded variability δ, we have* $\mathsf{Goal}(\varphi, S, \rightsquigarrow, \varepsilon, \delta) \subseteq \mathsf{Out}(\varphi, S, \rightsquigarrow, \varepsilon, \delta)$.

Proof. Let φ be an STL formula, $(S, \rightsquigarrow)$ be a distributed signal of bounded clock skew ε and bounded variability δ with $S = (x_1, \ldots, x_n)$.

Recall that each element of $\mathsf{Goal}(\varphi, S, \rightsquigarrow, \varepsilon, \delta)$ is a boolean value expression for the satisfaction signal of φ for some trace of $(S, \rightsquigarrow)$. Let $v \in \mathsf{Goal}(\varphi, S, \rightsquigarrow, \varepsilon, \delta)$ be arbitrary. To conclude that $v \in \mathsf{Out}(\varphi, S, \rightsquigarrow, \varepsilon, \delta)$, we argue that the semantics of our logic explores all admissible interleavings (subject to $\rightsquigarrow$, ε, and δ) of the edges of S.

First, notice that the canonical segmentation $G_S = \{I_1, \ldots, I_k\}$ and the function γ that maps each segment in G_S and signal in S to a value expression capture all potential behaviors of the individual signals in S. In particular, we claim that for each trace $w = (x'_1, \ldots, x'_n) \in \mathsf{Tr}(S, \rightsquigarrow, \delta)$ and for each i, we have $\mu(x'_i) \in \mathsf{destutter}(\gamma(I_1, i) \ldots \gamma(I_k, i))$. This holds because (i) each signal x'_i has the same edges as x_i modulo the time stamps, and (ii) for each x_i and each uncertainty interval J of x_i, the value expression of J can be reconstructed by some choice of elements from the sequence of γ values corresponding to the value expressions of the canonical segments that are contained in J.

Now, we argue that the semantics of our logic preserve this property. One key observation is that the asynchronous product of two sets of value expressions captures all the admissible interleavings of the edges involved in each potential behavior encoded as a value expression these sets. Then, the cases of atomic propositions, negation, and conjunction are trivial. For untimed until,

the definitions of the bitwise-temporal operations coincide with their discrete counterparts, therefore they are correct for each interleaving. For timed eventually and always, in addition to the arguments above, observe that the function profilesBetween enumerates all possible relations between the operator's interval I and the segments we compute the satisfaction over.

Claim. TODO: The algorithm runs in ...

Proof. Let $(S, \rightsquigarrow)$ be a distributed signal of n signals of length d and a total of g edges. Let φ be an STL formula of size m. Let ε be the maximum clock skew and δ the bounded variability constant. Note that $g \leq n\lceil\frac{d+\varepsilon}{\delta}\rceil$.

1. **Enumerating the subformulas take $O(m)$ time and space.**
2. **Computing the canonical segmentation takes $O(g \log g)$ time and $O(g)$ space.**
   - Given the signals' edges, computing their uncertainty intervals takes $O(1)$ time for each edge. We store their end points in a list of size $O(g)$, which we sort in $O(g \log g)$ time.
3. **Computing the value expressions of signals takes $2^{O(\frac{\varepsilon}{\delta}+\log g+\log n)}$ time and space.**
   - For each segment I_k, we compute the sets $Y_{k,1}, \ldots Y_{k,n}$ by searching the end points of I_k in the uncertainty intervals of $x_1, \ldots, x_n$ one by one, which takes $O(g)$ time. Since there are $O(g)$ segments, computing all the sets $Y_{k,i}$ takes $O(g^2)$ time. Note that each set $Y_{k,i}$ has at most $\frac{4\varepsilon}{\delta}$ elements, so the computation takes $O(\frac{gn\varepsilon}{\delta})$ space.
   - Computing each set $\mathrm{op}(v)$ takes $O(1)$ time and space. Note that each $\mathrm{op}(v)$ contains at most 4 elements of length at most 2. Since each $Y_{k,i}$ contains $O(\frac{\varepsilon}{\delta})$ elements, there are $O(\frac{\varepsilon}{\delta})$ many $\mathrm{op}(v)$ sets for each k and i. Therefore, it takes $O(\frac{gn\varepsilon}{\delta})$ time and space to compute all the sets $\mathrm{op}(v)$.
   - To compute each set $Z_{k,i}$, we take concatenate and destutter the $\mathrm{op}(v)$ sets. The concatenation takes $2^{O(\frac{\varepsilon}{\delta})}$ time and space because each $\mathrm{op}(v)$ set has $O(1)$ elements of $O(1)$ length, and there are $O(\frac{\varepsilon}{\delta})$ such sets. Note that each set $Z_{k,i}$ has $2^{O(\frac{\varepsilon}{\delta})}$ elements of length $O(\frac{\varepsilon}{\delta})$.
   - For computing each set $\gamma(I_k, i)$ from $Z_{k,i}$, the desuttering can be done on-the-fly in the previous step by comparing the first and last elements of the concatenated expressions, which takes $O(1)$ time and space for each element of each $\mathrm{op}(v)$ set. Finally, since there are n signals and $O(g)$ segments, computing all the sets $\gamma(I_k, i)$ takes $2^{O(\frac{\varepsilon}{\delta}+\log g+\log n)}$ time and space.
4. **Computing the value expressions of subformulas takes $O(...)$ time and $O(...)$ space.**
   - For an atomic proposition p, we take as input the γ sets of each segment and the signals involved in p. Recall from the previous step that each γ contains $2^{O(\frac{\varepsilon}{\delta})}$ elements of length $O(\frac{\varepsilon}{\delta})$. To compute the corresponding n-ary function f_p in a given segment I_k, we need to compute the asynchronous product $\gamma(I_k, 1) \otimes \ldots \otimes \gamma(I_k, n)$. This computation takes $2^{O(\frac{n\varepsilon}{\delta})}$ time and space, and the product contains $2^{O(\frac{n\varepsilon}{\delta})}$ elements

of length $O(\frac{n\varepsilon}{\delta})$. We assume that, for a single-letter alignment of the tuples in the product, applying f_p and comparing the output with zero take $O(1)$ time, and thus $2^{O(\frac{n\varepsilon}{\delta})}$ time for a single segment. During these computations, we compute the boolean value expressions by destuttering on the fly. Repeating the steps for each segment takes $2^{O(\frac{n\varepsilon}{\delta}+\log g)}$ time, resulting in $O(g)$ sets of boolean value expressions, each containing $O(\frac{n\varepsilon}{\delta})$ elements of length $O(\frac{n\varepsilon}{\delta})$.

- We investigate the remaining operations where we take the inputs as sets of boolean value expressions of $O(\alpha)$ elements of length $O(\beta)$. We remark that a set of boolean value expressions can be represented as two bit vectors marking the positions corresponding to elements in the set. For example, the set $U = \{0, 01, 10, 101\}$ can be represented as $u_0 = 011$ and $u_1 = 110$ where u_σ encodes the elements that start with letter σ and a position i is marked with 1 iff U contains an element of length i that starts with σ.
- To compute the negation of a set of value expressions, we simply store a boolean flag that effectively swaps the names of the corresponding bit vectors. Since this flag applies to all segments at once, it takes $O(1)$ time and space in total.
- For the conjunction of two sets of value expressions, first observe the following: (i) for all $u, v \in \mathsf{destutter}(\Sigma^*)$, we can compute $u \wedge v$ in $O(|u|+|v|)$ time and space, and (ii) for all $u, u', v, v' \in \mathsf{destutter}(\Sigma^*)$ such that u and u' start and end with the same letter, v and v' start and end with the same letter, $|u'| \leq |u|$ and $|v'| \leq |v|$, we have $u' \wedge v' \subseteq u \wedge v$. These statements hold because the conjunction of two boolean value expressions cannot have more edges can the sum of the number of edges of the inputs, and they can be aligned to obtain an output with any smaller number of edges. **Ege: There are some corner cases to consider, but this is almost true.** Therefore, we only need to compute the conjunctions of the longest elements of each type in each set (if they exist), which takes $O(\beta_1 + \beta_2)$ time and space for each segment, and thus $O(g(\beta_1 + \beta_2))$ time and space in total. The resulting sets each contain $O(\beta_1 + \beta_2)$ elements of length $O(\beta_1 + \beta_2)$. Note that each conjunction potentially doubles the set size. **Ege: The explanation needs to be improved.**
- For the until operator, observe that (i) for all $u, v \in \mathsf{destutter}(\Sigma^*)$, we can compute $u \wedge v$ in $O(|v|)$ time and space, and (ii) for all $u, u', v, v' \in \mathsf{destutter}(\Sigma^*)$ such that u and u' start and end with the same letter, v and v' start and end with the same letter, $|u'| \leq |u|$ and $|v'| \leq |v|$, we have $u' \mathcal{U} v' \subseteq u \mathcal{U} v$. These hold for similar reasons as for conjunction, but now the number of edges in the output depends on the number of edges of v and not of u. **Ege: To be checked.** Therefore, the computation takes $O(\beta_2)$ time and space for each segment, and thus $O(g\beta_2)$ time and space in total. The resulting sets each contain $O(\beta_2)$ elements of length $O(\beta_2)$.

- For the bounded temporal operators, first note that there are $O(g)$ profiles to be considered. Computing a profile may include a concatenation of $O(g)$ sets of value expressions (due to κ when I is large), thus it takes $2^{O(g \log \alpha)}$ time, resulting in a set of $O(\alpha(g+\beta))$ elements of length $O(\alpha(g+\beta))$. Computing the bitwise temporal operators on such a set takes $O(\alpha(g+\beta))$ time, resulting in a set of $O(1)$ elements of length $O(1)$. Concatenating the resulting sets of $O(g)$ profiles thus takes $O(g)$ time and space, resulting in a set of $O(g)$ elements of length $O(g)$.
- TODO: explain the worst case and update the statement.
  Only conjunctions: $2^{O(\alpha+\log g+\log m)}$, output with $O(m\alpha)$ elements of length $O(m\alpha)$.

5. **Computing the output takes $2^{O(g \log X)}$ time and $O(gY)$ space.** We concatenate $O(g)$ sets of value expressions for the root formula, which contains in the worst case $O(X)$ elements of length $O(Y)$. This can be done together with on-the-fly destuttering in $2^{O(g \log X)}$ time and $O(gY)$ space. TODO: replace X and Y based on the worst case above.

5 Experimental Evaluation

TODO: the setting etc.

5.1 Optimizations

Data structures TODO: computing boolean value expressions fast (the box representation and the operations)

Preprocessing Restructuring of the subformulas and the input signal to improve the algorithm's precision.

1. Enumerate the subformulas of φ such that each formula has an enumeration number smaller than the numbers of all its subformulas. Let $\varphi_1, \ldots, \varphi_m$ be such an enumeration. Note that $\varphi_1 = \varphi$.
2. For each agent A_i, let P_i be the set of subformulas of φ such that for every $\psi \in P_i$ we have (i) $\pi(x) = A_i$ for each signal x that occurs in ψ, (ii) the temporal operators ψ contains, if any, are untimed, and (iii) for each subformula ψ' of φ such that ψ is a subformula of ψ', the first two conditions are violated. For each $\psi \in P_i$, if ψ is not an atomic proposition, define a fresh signal y such that $\pi(y) = A_i$. Let $y_1, \ldots, y_M$ be the fresh signals and let $\mathsf{f} : \{y_1, \ldots, y_M\} \to \{\varphi_1, \ldots, \varphi_m\}$ be function that maps each fresh signal to the corresponding subformula of φ. Then, $y_j(t) = [w, t \models \mathsf{f}(y_j)]_{\mathsf{STL}}$ for all $1 \leq j \leq M$ and $0 \leq t < d$, where w is the trace obtained from S by assuming complete synchrony.

3. For each fresh signal y_j, define a fresh proposition $q_j = (y_j > 0)$ and replace in φ the subformula $\mathsf{f}(q_j)$ with q_j. Let φ' be the obtained formula. Let $\varphi'_1, \ldots, \varphi'_{m'}$ be the subformulas of φ' satisfying the enumeration invariant given above. Note that φ'_1 and φ are semantically equivalent.
4. We define a new distributed signal appropriately extending $(S, \rightsquigarrow)$ with the fresh propositions. Let $(S', \rightsquigarrow')$ be a distributed signal with $S' = (x_1, \ldots, x_n, y_1, \ldots, y_M)$ and $\rightsquigarrow'$ the smallest extension of $\rightsquigarrow$ from S to S' satisfying Definition 2.

5.2 Results

TODO

References

1. Ganguly, R., Momtaz, A., Bonakdarpour, B.: Distributed runtime verification under partial synchrony. In: Bramas, Q., Oshman, R., Romano, P. (eds.) 24th International Conference on Principles of Distributed Systems, OPODIS 2020, December 14-16, 2020, Strasbourg, France (Virtual Conference). LIPIcs, vol. 184, pp. 20:1–20:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2020). https://doi.org/10.4230/LIPIcs.OPODIS.2020.20
2. 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
3. Momtaz, A., Abbas, H., Bonakdarpour, B.: Monitoring signal temporal logic in distributed cyber-physical systems. In: Mitra, S., Venkatasubramanian, N., Dubey, A., Feng, L., Ghasemi, M., Sprinkle, J. (eds.) Proceedings of the ACM/IEEE 14th International Conference on Cyber-Physical Systems, ICCPS 2023, (with CPS-IoT Week 2023), San Antonio, TX, USA, May 9-12, 2023. pp. 154–165. ACM (2023). https://doi.org/10.1145/3576841.3585937