Logical Compression: Source Coding for Distributed Temporal Logic Monitoring *

Valdemar Tang Evans(✉), Daniel Lucani, Hei Victor Cheng, Thomas Wright, and Claudio Gomes

Department of Electrical and Computer Engineering, Aarhus University, Aarhus, Denmark
{valdemar.tang,daniel.lucani,hvc,thomas.wright,claudio.gomes}@ece.au.dk

Abstract. We introduce source coding as a method for compressing intermediate Signal Temporal Logic (STL) verdicts in a distributed system with a centralized monitor. Additionally, we introduce stateful and event-triggered source coding to further enhance compression. We theoretically analyze the effects of allocation and aggregation of properties on compressibility, and establish conditions where aggregation guarantees better compression. Furthermore, we outline an algorithm exploiting logical associativity laws and property re-arrangement to increase aggregation and potentially compression. We test our approaches on industry-related STL specifications and investigate the effects of aggregation and number of allocated properties under our proposed schemes. We demonstrate that we can achieve compression ratios up to 5.96×10^3 and 32.4 compared to raw signals and uncompressed atomic verdicts, respectively.

Keywords: Runtime Monitoring · Compression · Temporal Logic

1 Introduction

With advances in processing and communication technology, a wide range of applications for (distributed) cyber-physical systems (CPS) become feasible. Regardless of whether the system is distributed over a large geographical area or more compact, communication technologies have limited bandwidth. In wireless systems the available bandwidth is physically limited or subject to licensing fees, and in dedicated mediums, real-time protocols limit the throughput to provide specific guarantees [1]. In any case, there is a clear motivation to occupy as little bandwidth as possible, and save resources for other systems or cut costs. Many CPSs perform safety-critical tasks, e.g. health-monitoring [7], control autonomous vehicles [29] or operate nuclear power plants [32]. Thus, ensuring correct operation is of paramount importance in many CPSs. A lightweight method

* This work was supported in part by the GROWLean Project by the Independent Research Fund Denmark under grant 2035-00229B, in part by the Lighthouse of Water Technologies Ph.D. School Project under EU REACT, in part by the Villum Foundation under grant VIL 500075, and in part by Horizon Europe and the RoboSAPIENS project under grant 101133807.

to verify whether a CPS is safe and/or performant, is runtime verification (RV) [2]. In RV, the observable system output is evaluated against a formally defined specification at runtime to determine whether the system has (or is on track to) satisfy the specification.

Related work. RV in distributed CPSs is challenging for a number of reasons, including lack of perfect synchronization, communication delays and partial system failures [16]. Existing work in this area deals with issues related to clock skew and monitoring algorithm correctness in distributed Linear Temporal Logic [27] (LTL) monitoring [9,14,18], Metric Temporal Logic (MTL) [19,21], as well as STL [25]. Message complexity or communication efficiency is considered in several works for distributed LTL monitoring algorithms [4,5,6,9,13,15,17]. The proposed communication schemes are explicitly optimized by e.g. only communicating properties that are true [9], only communicating properties whose verdict has changed since the previous communication time [6], only communicating information unknown to the receiver [4], strictly communicating the least information necessary to reach a verdict [5], or communicating a symbolic view representing the possible monitor states given the locally observed information [17]. Importantly, the number of messages and/or message size communicates updates verbatim and generally scales with the number of monitored atomic properties or the size of the overall property as seen in [9].

A paper that truly considers compression for STL monitoring is [3], where the authors perform a Fourier analysis of temporal logic robustness monitors using Volterra approximations. The authors analyze the impact of specific input signal frequencies on the output robustness, enabling the application of filters to discard frequencies which have little impact on robustness computation. This facilitates monitoring-safe frequency domain compression and the integration of STL monitoring into standard signal processing tool chains. This approach is however offline and centralized, and thus does not consider issues related to runtime monitoring in distributed systems.

Our approach to compression distinguishes itself from existing work, as we consider an online setting and decouple the message size from the number of atomic properties or property size, and instead efficiently encode messages based on their probability of occurrence. This means that our approach is complimentary to existing approaches rather than competing, as it may be applied on top of existing algorithms.

Contributions: We approach the problem of reducing communication for monitoring of temporal logic properties (with a focus on Signal Temporal Logic [23] (STL)) in distributed CPSs using two methods; source coding (an online compression technique), and property allocation/aggregation for better compression. For simplicity, we assume the existence of a global clock and a fixed centralized network topology. To encode and compress the partial verdicts, we present 4 variants of source coding with two conceptual underlying techniques; stateful (and stateless) source coding, and event-triggered (and periodic) transmission strategies. Furthermore, we theoretically analyze property aggregation in terms of how it may benefit compressibility for different aggregation operations. Finally, we

evaluate the compression and aggregation synergies in an industry-related case study [22].

Overview: In section 2, we introduce temporal logic and distributed monitoring preliminaries followed by the system architecture and problem statement in section 2.3 and compression preliminaries in sections 2.4 and 2.5. We introduce the 4 variants of source coding in section 3 and in section 4, we discuss and analyze property allocation and aggregation in terms of compression. In section 5 we empirically evaluate our proposed methods using an industry-inspired Simulink model of an automatic transmission system and several automatically generated STL properties.

2 Preliminaries and Problem Formulation

2.1 Past-Time Signal Temporal Logic (STL)

STL [23] is a formal specification language used to express real-time properties over real-valued signals. For simplicity, we adopt the discrete-time signal model of [31] and assume that samples are obtained and transmitted with some period Δ. Therefore, we can represent the time domain as an interval of integers $\mathbb{T} = \{0, ..., z\}$ where $z \in \mathbb{N}$. In this paper, we strictly consider the past bounded-time fragment of STL (**bpSTL**), and remark that a future-time STL property can be transformed to an equivalent past-time property using a pastification procedure [31]. The past-time fragment of STL is defined by the following recursive syntax

$$\varphi ::= \text{True} \mid \varphi^p \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \mathbf{S}_{[a,b]} \varphi_2 \tag{1}$$

where φ^p is an atomic property of the form $p(\mathbf{y}_k) \sim c$ where $\sim \in \{>, <, \geq, \leq, =, \neq\}$, $\mathbf{y}_k \in \mathbb{R}^m$ is the system output vector at time t_k, c is a constant, $\neg$ is the negation operator, $\wedge$ is the logical *and* operator and $\varphi_1 \mathbf{S}_{[a,b]} \varphi_2$ is the *since* operator, where $a, b \in \mathbb{N}$ and $a < b$. The operators in eq. (1) can be used to construct additional useful operators; the *or* operator $\varphi_1 \vee \varphi_2 \equiv \neg(\neg\varphi_1 \wedge \neg\varphi_2)$, the *once* operator ($\mathbf{O}_{[a,b]}\varphi \equiv \text{True}\ \mathbf{S}_{[a,b]}\varphi$), the *historically* operator ($\mathbf{H}_{[a,b]}\varphi \equiv \neg\mathbf{O}_{[a,b]}\neg\varphi$) and the *implication* operator ($\varphi_1 \rightarrow \varphi_2 \equiv \neg\varphi_1 \vee \varphi_2$). The formal satisfaction relation is provided in definition 1. s

Definition 1 (STL Property Satisfaction [23]). *The Boolean satisfaction of an STL property is defined using the notation* $(\mathbf{y}, t_k) \models \varphi$ *indicating that the system trace* $\mathbf{y}$ *satisfies the property* φ *at time* t_k *where,* $k \in \mathbb{T}$ *:*

$$\begin{aligned}
&(\mathbf{y}, t_k) \models \text{True} && \Leftrightarrow \text{True} \\
&(\mathbf{y}, t_k) \models p(\mathbf{y}_k) \sim c && \Leftrightarrow p(\mathbf{y}_k) \sim c \\
&(\mathbf{y}, t_k) \models \neg\varphi && \Leftrightarrow \neg((\mathbf{y}, t_k) \models \varphi) \\
&(\mathbf{y}, t_k) \models \varphi_1 \wedge \varphi_2 && \Leftrightarrow ((\mathbf{y}, t_k) \models \varphi_1) \wedge ((\mathbf{y}, t_k) \models \varphi_2) \\
&(\mathbf{y}, t_k) \models \varphi_1 \mathbf{S}_{[a,b]} \varphi_2 && \Leftrightarrow (\exists j \in [k-b, k-a] \cap \mathbb{T}\ s.t.\ (\mathbf{y}, t_j) \models \varphi_2) \wedge (\forall i \in [j, k](\mathbf{y}, t_i) \models \varphi_1)
\end{aligned}$$

Based on a property φ, an efficient monitoring program (referred to as the monitor) can be synthesized to evaluate the property against a signal trace [2],

or alternatively, a trace of verdicts of atomic properties. The approach of using verdicts of atomic properties is similar to monitoring Metric Temporal Logic [21]. For a given signal trace, we denote the verdict of a property φ at the k-th timestep as:

$$V_k(\varphi) = \begin{cases} \text{True} & \text{if } (\mathbf{y}, t_k) \models \varphi \\ \text{False} & \text{otherwise.} \end{cases} \tag{2}$$

We remark that this approach prohibits non-constant interpolation between sampling points and prevents a more thorough monitoring procedure as present in some STL monitoring algorithms [8,12]. Moreover, although we assume a fixed sampling frequency, our approach is not generally limited to this assumption except for the step in section 3.1. Furthermore, dealing with aspects of asynchronocity is out of scope for this paper, since this is an issue related to monitoring and not communication or compression. We emphasize that our approach is not limited to synchronous systems, although we assume that the system synchronized for simplicity.

2.2 Notation

For a sensor s^i with $i = 1, ..., q$, let $\mathcal{P}^i = \{\varphi^i_j\}_{j=1}^{n^i}$ denote the set of *arbitrary STL* properties to be monitored where n^i is the number of properties to be monitored by s^i. The collection of verdicts of the properties in $\mathcal{P}^i$ at a given time t_k is denoted by $V_k(\mathcal{P}^i)$. We can represent $V_k(\mathcal{P}^i)$ as a binary string, e.g. $V_k(\mathcal{P}^i) = 100011$ if $|\mathcal{P}^i| = 6$ where the j-th bit corresponds to $V_k(\varphi^i_j)$ in $\mathcal{P}^i$ and True $\equiv 1$, False $\equiv 0$. We denote a verdict trace over $\mathbb{T}$ as $\mathbf{V}(\varphi) = \langle V_0(\varphi), V_1(\varphi), ..., V_k(\varphi), ... \rangle$, and define $\mathbf{V}(\mathcal{P})$ in the same manner. Furthermore, let $\mathcal{V}(\mathcal{P}^i) \subseteq \mathbb{B}^{|\mathcal{P}^i|}$ denote the set of all possible combinations of verdicts that the properties in $\mathcal{P}^i$ can take for any time t_k. When not necessary, we remove the sensor i superscript for $\mathcal{P}^i$ and the k subscript from $V_k(\mathcal{P})$ when the timestep is not important. Later, we treat $V(\mathcal{P})$ as a discrete random variable. Consequently, we denote the probability of $V(\mathcal{P})$ taking a value $v \in \mathcal{V}(\mathcal{P})$ as $P(V(\mathcal{P}) = v)$ or $P(v)$ when brevity is preferred. We can then form the discrete probability mass function, denoted as $f(V(\mathcal{P}))$. Although we consider $V(\mathcal{P})$ a single random variable, $f(V(\mathcal{P}))$ is a joint distribution of the verdicts of all properties in $\mathcal{P}$.

2.3 Assumptions, System Architecture and Problem Formulation

Assumption 1 *We assume that the information necessary to evaluate the satisfaction of an atomic property is available at a single sensor.*

Assumption 2 *We assume that all defined STL properties are well-formed, meaning that there exists one or more traces that can either satisfy or violate a given property.*

Importantly, assumption 1 allows the sensors to transmit the verdicts of atomic properties (or potentially aggregated sub-properties) instead of the raw signals. While assumption 1 restricts the applicability of this approach (e.g. we do no allow specifying properties such as $x + y > 5$ if x and y are observed by different sensors), assumption 2 is mild, and is often assumed in other work [18,26].

Two traditional architectures related to orchestration monitoring and compression are shown in fig. 1 (top and middle respectively) in addition to our proposed approach (bottom). In a traditional orchestration monitoring approach (fig. 1 top), the atomic sub-properties of the global property are allocated to the sensors, which then transmit the uncompressed verdicts to the centralized monitor. In a traditional compression setup (fig. 1 middle), sensors compress and transmit raw signal values, which can then be used as input in a traditional monitoring setup after decompression. Conversely, our proposed approach employs source coding to compress the verdicts of the allocated properties. Furthermore, instead of allocating atomic properties for monitoring, we investigate how property aggregation and allocation may benefit compression.

Problem Definition and Contribution Overview. We state our overall goal as *reducing the communication necessary to perform distributed monitoring of temporal logic properties*. We approach the overall goal by breaking it into two sub-problems; a compression problem (**CP**) and an allocation problem (**AP**):

CP Given a set of properties $\mathcal{P}$ to be monitored by a sensor, how can we efficiently compress the verdicts $V(\mathcal{P})$, for transmission to the centralized monitor?

AP Given a property φ to be monitored by the centralized monitor and a sensor-to-atomic property mapping, how do we identify the set of properties $\mathcal{P}$ to be monitored by each sensor to achieve good compression, while ensuring that we can monitor φ?

We make two key contributions: 1) to address **CP**, we show how to implement four variants of source coding exploiting time-varying behavior and bounded variability (section 3), and 2) to address **AP**, we theoretically analyze the effects of property aggregation and allocation and outline a strategy to increase aggregation (section 4). Thus, we do not provide a full solution for **AP**, but characterize its theoretical foundations. We choose to present the contributions in this order since the various compression and communication techniques will aid us in the theoretical analysis of property aggregation. We remark that while the contributions related to **CP** are generally applicable across monitoring architectures, the contributions related to **AP** are only applicable to the orchestration monitoring architecture in its current form. The reader is referred to [13] for a comparison of different monitoring architectures. Next, we provide necessary background information to understand the proposed compression techniques.

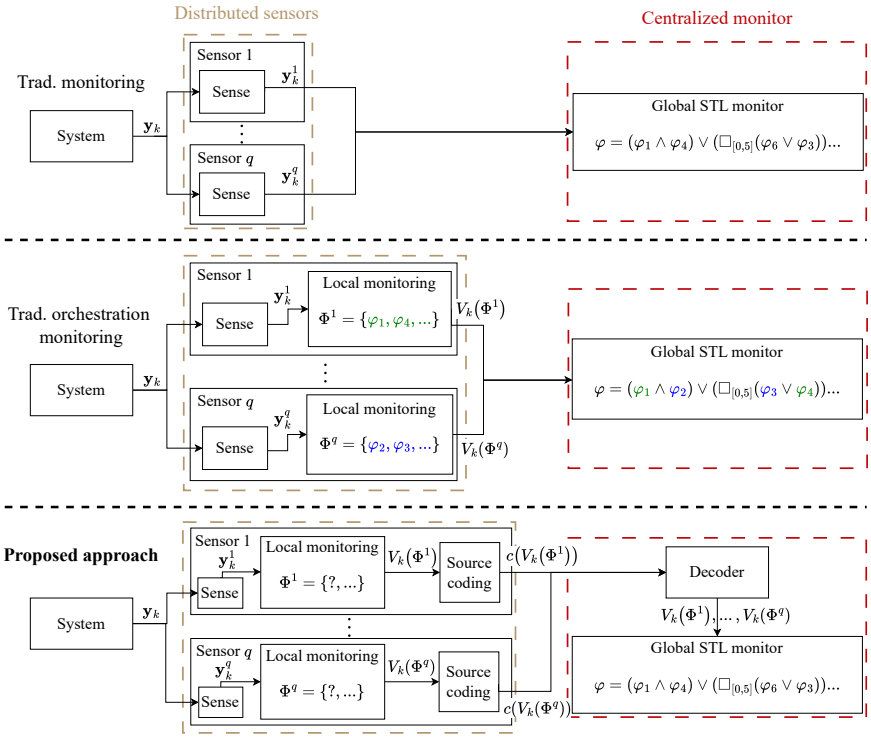


Fig. 1: The conceptual system architecture and data flow for the proposed method (bottom), traditional orchestration-based monitoring with verdicts (middle) and a traditional monitoring (top). The notation related to source coding used in this figure is introduced in section 2.5.

2.4 Event-Triggered Communication

The term event-triggered communication covers approaches where an event is used to trigger communication between two entities. In our setup, the system outputs are sampled at discrete timesteps where the sampled values are used to incrementally monitor a set of properties locally on a sensor. A change of verdict constitutes an event, expressed formally as the condition

$$V_k(\mathcal{P}) \neq V_{k-1}(\mathcal{P}). \tag{3}$$

2.5 Entropy and Source Coding

Compression refers to the process of finding a shorter length encoding or representation of the original data (possibly with a loss of information). We define the compression ratio (CR) as

$$CR = \frac{\text{Uncompressed size}}{\text{Compressed size}} \tag{4}$$

meaning that a higher compression ratio is preferable. Source coding is the process of encoding data for maximum compression based on the Shannon Entropy (or just entropy). The entropy determines the information content needed to represent the outcome of discrete random variable on average and identifies the limit of lossless compression. For a discrete random variable X, the entropy is defined as:

$$H(X) = -\sum_{x \in \mathcal{X}} P(x) \log_2 P(x) \tag{5}$$

where $\mathcal{X}$ is the set of possible values that X can take and $P(x)$ is the probability of $X = x$. Furthermore, it follows from eq. (5) that $H(X) \geq 0$. A notable property of eq. (5) is that a uniform distribution maximizes entropy, while an uneven distribution decreases entropy. Additionally, the joint entropy of 2 random variables, X, Y is defined by

$$H(X,Y) = -\sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} P(x,y) \log_2 P(x,y) = H(X) + H(Y|X) \tag{6}$$

$$\text{where } H(X|Y) = -\sum_{y \in \mathcal{Y}} P(y) H(X|Y = y). \tag{7}$$

Eqs. (6)-(7) may be extended to an arbitrary number of random variables [10]. Additionally, we note that

$$H(g(X,Y)) \leq H(X,Y) \tag{8}$$

where $g(X,Y)$ is any deterministic function of X, Y. We remark that entropy can be thought of as a measure of uncertainty, where a larger entropy corresponds to larger uncertainty. Thus, we may introduce a deterministic variable without changing the entropy:

$$H(X,Y,g(X,Y)) = H(X,Y) + H(g(X,Y) \,|\, X,Y) = H(X,Y) \tag{9}$$

since $H(g(X,Y) \,|\, X,Y) = 0$ when $g(X,Y)$ is any deterministic function of X, Y.

In the context of source coding, the set of possible values $\mathcal{X}$ is known as the source symbols. Given $\mathcal{X}$, we identify codewords (CWs) (bitstrings that represent specific source symbols) and define a mapping between the source symbols and CWs, e.g. $c(x) = 10111$ encodes that $X = x$. The set of CWs is called the codebook (CB), denoted as $\mathcal{C}(X)$. The length of a CW $c(x)$ is given by the number of bits and is denoted as $l(c(x))$. The average CW length can be computed as $E_l\left[\mathcal{C}(X)\right] = \sum_{x \in \mathcal{X}} P(x) l(c(x))$. Finally, based on the discrete probability distribution of X, we can construct an optimal CB and an optimal mapping from CWs to source symbols such that

$$H(X) \leq E_l\left[\mathcal{C}(X)\right] < H(X) + 1 \tag{10}$$

using Huffman coding [20]. Due to space constraints, we refer the reader to [24] for an introduction to Huffman coding. We remark that to properly decode

messages, the transmitter and receiver must agree upon the CB(s) during or before starting operation.

Later in section 3, we present different communication schemes and source coding approaches to both compress and reduce communicated source symbols and in section 4, we utilize the definition of entropy to analyze the impact of aggregating STL properties on the entropy of their verdicts.

3 Source Coding (CP)

For now, we assume that for every sensor s^i, $\mathcal{P}^i$ has been defined. We return to the allocation problem in section 4. To implement source coding, we must first identify the set of source symbols $\mathcal{V}(\mathcal{P})$. One may be tempted to define the source symbols as all possible binary combinations of verdicts resulting in $2^{|\mathcal{P}^i|}$ source symbols. However, we note that the satisfaction of one property imposes constraints on the value of the associated signals, which in turn imposes constraints on the verdict of other properties associated with the same signal(s). For example, consider the case $\mathcal{P}^i = \{x > 10, x < 5\}$. Both properties cannot possibly be satisfied at the same time, thus reducing the size of the set of possible source symbols.

3.1 Identifying Source Symbols ($\mathcal{V}(\mathcal{P})$)

We propose to identify the source symbols $\mathcal{V}(\mathcal{P})$ using a Satisfiability Modulo Theory (SMT) solver [11]. SMT solvers allow the user to input a set of constraints, and outputs (if decidable) whether the constraints are satisfiable. Such a constraint encoding essentially expands temporal operators into disjunctions and conjunctions over time and has been previously defined for model checking problems [28]. In our setup, one constraint is produced for every sampling timestep over the temporal extent of a given property. For example, the following constraints are produced by $\mathbf{O}_{[a,b]}\varphi_1^p : \bigvee_{j=a}^{b} \zeta(\varphi_1^p, t_{k-j})$ where $\zeta(\varphi_1^p, t_{k-j})$ denotes the constraints imposed by the satisfaction of φ_1^p at time t_{k-j}. To determine $\mathcal{V}(\mathcal{P})$, we task the SMT solver with evaluating all possible verdict combinations involving $2^{|\mathcal{P}|}$ tests. For example, for two properties we test if the constraints induced by the following cases are satisfiable: $\varphi_1 \wedge \varphi_2, \varphi_1 \wedge \neg\varphi_2, \neg\varphi_1 \wedge \varphi_2, \neg\varphi_1 \wedge \neg\varphi_2$. The satisfiable cases can then be translated to source symbols, where a 1 represents the satisfaction of a property, and a 0 represents the violation of a property, e.g. if $\varphi_1 \wedge \neg\varphi_2 \wedge \varphi_3$ is satisfiable, 101 is a valid $v \in \mathcal{V}(\mathcal{P})$.

If $\mathcal{P}$ is large and if the temporal horizons are long, evaluating $\mathcal{V}(\mathcal{P})$ may be computationally expensive. Although the evaluations can be parallelized, there are other ways or circumventing this issue. One option is to not include temporal operators with long horizons in the SMT tests and include any possible combination of satisfaction and violation of the excluded properties as possible source symbols. Alternatively, the CB may be constructed using all possible $|\mathcal{P}|^2$ source symbols. State combinations which are never observed because they are impossible will have a probability of zero, and thus be assigned a long CW. In a truly

online setting, however, the perhaps cleanest way is to employ dynamic Huffman coding [30], which constructs the CB and adapts source symbol to CW mappings on-the-fly. This does however incur an overhead whenever previously unobserved states occur. This approach naturally avoids impossible verdicts combinations since they are never observed and thus do not become part of the CB.

The SMT encoding in this section is tied to the assumption of a fixed sampling frequency and does not consider signal interpolation. This assumption and limitation is however also present in other online monitoring algorithms [31].

3.2 Source Coding

In this section, we detail how to implement source coding for distributed STL monitoring and present two additional techniques (stateful source coding and event-triggered communication) to increase compression. The additional techniques can be implemented separately or in unison. While our analysis focuses on the entropy, recall that the entropy and compression are related by eq. (10), i.e. a smaller entropy implies smaller upper and lower bounds in eq. (10).

Stateless Source Coding. To generate an optimal CB, we need to identify the discrete probability mass function of $V(\mathcal{P})$, $f(V(\mathcal{P}))$. In practice, $f(V(\mathcal{P}))$ can be estimated by monitoring traces from a training dataset. Based on $f(V(\mathcal{P}))$, we employ a CB generation technique and define a source symbol to CW mapping using e.g. Huffman coding. Then, at each timestep, the sensors transmit their encoded verdicts $c(V(\mathcal{P}))$ to the centralized monitor, where the verdicts are used as inputs to monitor the overall property as depicted in fig. 1. This procedure results in a low-complexity and straight-forward way to apply source coding to distributed STL monitoring. However, with this approach, we model the probability of a specific verdict under the assumption of a time-invariant system. While the time-invariant assumption may be true in some cases, this is not the case for systems where signals have bounded variability (i.e. are Lipschitz continuous). Thus, by introducing stateful behavior at the expense of memory, we may be able to achieve better compression by exploiting time-variability.

Stateful Source Coding. We motivate stateful source coding in Example 1.

Example 1. Assume $\mathcal{P} = \{x > 2, x > 5, x > 10\}$, and that $V_{k-1}(\mathcal{P}) = 100$. We argue that transitioning to $V_k(\mathcal{P}) = 110$ or remaining in the same state is more probable than transitioning directly to $V_k(\mathcal{P}) = 111$ due to continuity of signals. Thus, conditioning the next state on the state history likely reduces the entropy, corroborated by the well-known relation $H(X|Y) \leq H(X)$. Consequently, we may achieve better compression indicated by smaller bounds in eq. (10).

Implementing stateful source coding requires a CB to be identified for each possible version of the history. Therefore, the number of required CBs scales

with $|\mathcal{V}(\mathcal{P})|^h$, where $h \in \mathbb{Z}^+$ is the length of the history. For this reason, we base our prediction on a 1-step history characterized by

$$f(V_k \,|\, V_{k-1}) = \begin{bmatrix} P(V_k = v_1 \,|\, V_{k-1} = v_1) & \dots & P(V_k = v_m \,|\, V_{k-1} = v_1) \\ \vdots & \ddots & \vdots \\ P(V_k = v_1 \,|\, V_{k-1} = v_m) & \dots & P(V_k = v_m \,|\, V_{k-1} = v_m) \end{bmatrix} \tag{11}$$

where the V_k denotes the verdict of $V(\mathcal{P})$ at time t_k for brevity and $v_i \in \mathcal{V}(\mathcal{P})$ for $i = 1, ..., m$. Utilizing the information in $f(V_k \,|\, V_{k-1})$ to improve compression requires a stateful source coding approach where each state has its own associated CB and source symbol to CW mapping for the next state. To construct the CB for a state v, we first obtain the conditional distribution $f(V_k \,|\, V_{k-1} = v)$ from $f(V_k \,|\, V_{k-1})$, i.e. sum over each column in eq. (11). Then, based on $f(V_k \,|\, V_{k-1} = v)$, we construct the optimal CB as in the stateless approach. We perform this procedure for every $v \in \mathcal{V}(\mathcal{P})$ to define the full set of m CBs. During runtime at each timestep, the sensors and centralized monitor select the CB(s) corresponding to the previous state(s), and encode/decode the current state(s) based on the selected CB(s). While employing this stateful source coding approach does not guarantee an increase in compression (recall that $H(X|Y) \leq H(X)$), in our empirical evaluation in section 5 we find that the stateful approach typically does increase compression, although the compression gains vary between properties.

Source Coding under Event-Triggered Communication. For seldom changing verdicts, periodic transmission may convey redundant information. Assuming that the centralized monitor and distributed sensors are synchronized and agree on a specific sampling frequency, using the event-triggering scheme in eq. (3) to trigger message transmission, the centralized monitor can infer that the current verdict is identical to that of the previous time-step if no message is received. We note that the event-triggered communication scheme changes the distribution of the source symbols and should be accounted for to ensure optimal compression. Fortunately, the event-triggering distribution in both cases can be easily obtained from the non event-triggering distribution in eq. (11) by setting the probabilities on the diagonal equal to 0 (i.e. probabilities of remaining in the same state) and normalizing. For the stateful approach, the conditional distributions are defined as:

$$f(V_k \,|\, V_{k-1} = v_i)^{ET} = \left[\frac{P(V_k = v_1 \,|\, V_{k-1} = v_i)}{1 - P(V_k = v_i \,|\, V_{k-1} = v_i)} \dots 0 \dots \frac{P(V_k = v_m \,|\, V_{k-1} = v_i)}{1 - P(V_k = v_i \,|\, V_{k-1} = v_i)} \right]$$

where the 0 entry corresponds to the case where $V_k = v_i$. For the stateless approach, the distribution becomes:

$$f(V(\mathcal{P}))^{ET} = \begin{bmatrix} \sum_{v \in \mathcal{V}(\mathcal{P}) \setminus \{v_1\}} P(V_k = v_1 | V_{k-1} = v) \\ \vdots \\ \sum_{v \in \mathcal{V}(\mathcal{P}) \setminus \{v_m\}} P(V_k = v_m | V_{k-1} = v) \end{bmatrix}^\intercal .$$

Then, based on $f(V_k \mid V_{k-1} = v)^{ET}$ or $f(V(\mathcal{P}))^{ET}$ we can apply stateful or stateless source coding respectively. This marks the end of our contribution to address problem **CP**. Next, we consider the property allocation problem **AP**.

4 Property Allocation (AP)

As entropy determines the limit for compression, we start by theoretically analyzing how property aggregation affects the entropy $H(V(\mathcal{P}))$ in section 4.1. Later in section 4.2 we outline the practical problem of identifying possible property allocations.

4.1 The Effects of Property Aggregation on Compressibility

We start by motivating property aggregation with an intuitive example.

Example 2. Consider the property $\varphi = \varphi_1^1 \vee (\varphi_1^2 \wedge \varphi_2^2)$ where $\varphi_1^1 = y > 10$, $\varphi_1^2 = x < 10, \varphi_2^2 = x > 2$. By allocating the atomic properties to their respective sensors we have $\mathcal{P}^1 = \{\varphi_1^1\}, \mathcal{P}^2 = \{\varphi_1^2, \varphi_2^2\}$. The set of possible verdicts then becomes $\mathcal{V}(\mathcal{P}^1) = \{0, 1\}, \mathcal{V}(\mathcal{P}^2) = \{10, 01, 11\}$. However, we realize that $\varphi_1^2 \wedge \varphi_2^2$ is entirely monitorable by s^2, so we can *aggregate* φ_1^2, φ_2^2 and allocate $\mathcal{P}^2 = \{\varphi_1^2 \wedge \varphi_2^2\}$ instead of $\mathcal{P}^2 = \{\varphi_1^2, \varphi_2^2\}$ as knowing $V(\varphi_1^2 \wedge \varphi_2^2)$ is sufficient to determine the overall verdict. Consequently, $\mathcal{V}(\{\varphi_1^2 \wedge \varphi_2^2\}) = \{0, 1\}$, reducing the number of possible verdicts and potentially increasing compression.

Example 2 shows that we can trade off explainability (what sub-property violation/satisfaction caused the property to be violated/satisfied) for potential compression by aggregating properties while maintaining monitorability. Before progressing, we give a brief overview of the contents in this section. Initially, we focus on how aggregation using binary propositional operators $\triangle^b \in \{\wedge, \vee\}$ affects entropy of the aggregated property (similar to the case in example 2), followed by how aggregation affects the joint entropy of the verdicts of the allocated properties $H(V(\mathcal{P}))$. Finally, we investigate the effects of aggregation of temporal operators $\triangle^t \in \{\mathbf{O}_{[a,b]}, \mathbf{H}_{[a,b]}, \mathbf{S}_{[a,b]}\}$ under stateless and stateful source coding, as well as event-triggered communication.

Binary Aggregation. In theorem 1 we establish conditions for when aggregation does not increase entropy $H(V(\varphi))$, and when it strictly reduces entropy.

Theorem 1. *Let $\varphi_1 \triangle^b \varphi_2$ denote the application of $\triangle^b$ to φ_1, φ_2. Then $H(V(\varphi_1 \triangle^b \varphi_2)) \leq H(V(\{\varphi_1, \varphi_2\}))$. Additionally, this inequality is strict if $\big(\varphi_1 \neq \varphi_2$ and $\forall v \in \mathcal{V}(\{\varphi_1, \varphi_2\})\ P(v) > 0\big)$ where $\varphi_1 \neq \varphi_2$ is semantic inequivalence.*

Proof. Due to space limitations, we outline the proof here and provide the full proof in Appendix A. We can prove $H(V(\varphi_1 \triangle^b \varphi_2)) \leq H(V(\{\varphi_1, \varphi_2\}))$, by applying eq. (8). To prove $H(V(\varphi_1 \triangle^b \varphi_2)) < H(V(\{\varphi_1, \varphi_2\}))$, we use the insight that if the two properties are not equivalent, and for all $v \in \mathcal{V}(\{\varphi_1, \varphi_2\}) P(v) > 0$, then two or more outcomes of $V(\{\varphi_1, \varphi_2\})$ will map to a single outcome of $V(\varphi_1 \triangle^b \varphi_2)$, thus strictly reducing the entropy. □

Corollary 1. *Repeated aggregation decreases or maintains entropy for binary propositional operators. Under the necessary conditions for the strict inequality in theorem 1, the decrease in entropy is strict.*

Binary Aggregation and $H(V(\mathcal{P}))$. In theorem 2, we establish that binary aggregation never decreases the joint entropy, and identify conditions for when it strictly decreases joint entropy.

Theorem 2. *Binary aggregation of properties in $\mathcal{P}$ never increases $H(\mathcal{P})$. Additionally, binary aggregation decreases entropy $\big(H(V(\{\varphi_1 \triangle^b \varphi_2, \varphi_3, ...\})) < H(V(\{\varphi_1, \varphi_2, \varphi_3, ...\}))\big)$, if $H(V(\{\varphi_1, \varphi_2\}) \,|\, V(\{\varphi_1 \triangle^b \varphi_2, \varphi_3, ...\})) > 0$.*

Proof. By introducing $\varphi_1 \triangle^b \varphi_2$ (as per eq. (9)) and applying eq. (6), we have

$$H(V(\{\varphi_1, \varphi_2, \varphi_1 \triangle^b \varphi_2, \varphi_3, ...\})) = \\ H(V(\{\varphi_1 \triangle^b \varphi_2, \varphi_3, ...\})) + H(V(\{\varphi_1, \varphi_2\}) \,|\, V(\{\varphi_1 \triangle^b \varphi_2, \varphi_3, ...\})).$$

Since entropy is strictly positive, $H(V(\{\varphi_1, \varphi_2\}) \,|\, V(\{\varphi_1 \triangle^b \varphi_2, \varphi_3, ...\})) \geq 0$, thus proving that binary aggregation never increases entropy. If $H(V(\{\varphi_1, \varphi_2\}) \,|\, V(\{\varphi_1 \triangle^b \varphi_2, \varphi_3, ...\})) > 0$, then necessarily $H(V(\{\varphi_1 \triangle^b \varphi_2, \varphi_3, ...\})) < H(V(\{\varphi_1, \varphi_2, \varphi_3, ...\}))$, concluding the proof. □

Intuitively, $H(V(\{\varphi_1, \varphi_2\}) \,|\, V(\{\varphi_1 \triangle^b \varphi_2, \varphi_3, ...\})) > 0$ implies that knowing $V(\{\varphi_1 \triangle^b \varphi_2, \varphi_3, ...\})$ is not sufficient to deterministically infer $V(\{\varphi_1, \varphi_2\})$. Thus, a decrease in $H(\mathcal{P})$ depends on what properties are aggregated, and what properties are in $\mathcal{P}$.

Aggregation of Temporal Operators and their Effect on Entropy. As described in section 3.2, we may estimate $f(V(\mathcal{P}))$ based on system traces, from which we can estimate $H(\mathcal{P})$. When estimating the entropy in this manner, a strict relationship between the entropy before and after temporal property aggregation cannot be established. We show such a scenario in Example 3.

Example 3. Assume $\mathbf{V}(\varphi_1) = \langle \text{True}, \text{False}, \text{False}, \text{False} \rangle$ and let $\varphi = \mathbf{O}_{[0,1]}\varphi_1$. We then have $\mathbf{V}(\mathbf{O}_{[0,1]}\varphi_1) = \langle \text{True}, \text{True}, \text{False}, \text{False} \rangle$. Thus, $H(V(\mathbf{O}_{[0,1]}\varphi_1)) = 1$ and $H(V(\varphi_1)) = -((1/4)\log_2(1/4) + (3/4)\log_2(3/4)) = 0.81$, showing that temporal property aggregation may increase entropy. Conversely, if $\mathbf{V}(\varphi_1) = \langle \text{True}, \text{True}, \text{False}, \text{False} \rangle$, then $H(V(\mathbf{O}_{[0,1]}\varphi_1)) = 0.81$, while $H(V(\varphi_1)) = 1$. Thus, temporal property aggregation may also decrease entropy.

We note that a similar scenario to the one in example 3 exists for the remaining temporal operators, however with a notable difference for the $\varphi_1 \mathbf{S}_{[a,b]} \varphi_2$ operator: if $H(V(\{\varphi_1, \varphi_2\})) > 1$, then aggregation strictly decreases entropy since $0 \leq H(V(\varphi_1 \mathbf{S}_{[a,b]} \varphi_2)) \leq 1$, since the entropy of a binary random variable is at maximum 1. From this point on, we only treat the $\mathbf{O}_{[a,b]}$ and $\mathbf{H}_{[a,b]}$ operators and leave further analysis of the $\mathbf{S}_{[a,b]}$ operator for future work, due to the large number of edge cases and space limitations.

Despite the potentially poor prospects of aggregating temporal properties under the stateless source coding approach indicated by example 3, the $\mathbf{O}_{[a,b]}$ and $\mathbf{H}_{[a,b]}$ temporal operators exhibit deterministic behavior and bounded variability that can be exploited by stateful source coding and event-triggered communication to reduce increases in entropy.

Conditional Entropy of Unary Temporal Properties. To analyze the entropy of the $\mathbf{O}_{[a,b]}$ and $\mathbf{H}_{[a,b]}$ operators under property aggregation for the stateful source coding approach section 3.2, we consider the three different cases depicted in fig. 2. In the following, we only explicitly consider the $\mathbf{O}_{[a,b]}$ operator, but we note that an identical analysis can be conducted for the $\mathbf{H}_{[a,b]}$ operator by considering the occurrences of $V_k = 0$ instead of $V_k = 1$.

In general the conditional entropy of $V(\varphi)$ of any property φ is given by:

$$\begin{aligned} &H(V_k|V_{k-1}) = \\ &\quad P(V=0)H(V_k|V_{k-1}=0)+ \\ &\quad P(V=1)H(V_k|V_{k-1}=1) \end{aligned} \tag{12}$$

where $P(V=0)$ and $P(V=1)$ are the probabilities that $P(V(\varphi)=0)$ or $P(V(\varphi)=1)$ respectively, at any time.

Let $T(\varphi)$ denote the duration in the past needed to evaluate the verdict of φ at the current time. For example, $T(\mathbf{O}_{[0,3]}\varphi_1) = 3$.

Assume that $T(\varphi)$ is a multiple of the sampling interval T_s, and let $l = T(\varphi)/T_s + 1$, where l indicates the number of samples for the duration $T(\varphi)$. Additionally, let m denote fraction of samples where $V(\varphi) = 1$ and let n denote the trace length.

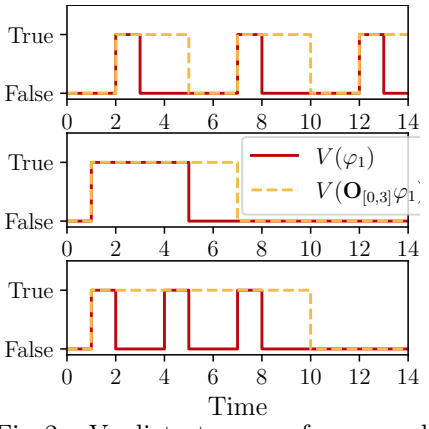


Fig. 2: Verdict traces of φ_1 and $\mathbf{O}_{[0,3]}\varphi_1$ for case 1 (top), case 2 (middle) and case 3 (bottom).

Case 1: Any isolated occurrence of $V(\varphi_1) = 1$ (referred to as spikes, see red line fig. 2 (top)) causes $V(\mathbf{O}_{[a,b]}\varphi_1) = 1$ to remain true for at least l samples. In general, if spikes are at least $l+1$ apart they cause $V(\mathbf{O}_{[a,b]}\varphi_1)$ to transition from 1 to 0, before again transitioning back to 1. We let $H_b(p) = -p\log p - (1-p)\log(1-p)$ and adapt eq. (12) in eqs. (13) and (14):

$$H(V_k(\mathbf{O}_{[a,b]}\varphi_1)|V_{k-1}(\mathbf{O}_{[a,b]}\varphi_1)) = \underbrace{\frac{n-lm}{n}}_{P(V=0)} \underbrace{H_b\left(\frac{m}{n-lm}\right)}_{H(V_k|V_{k-1}=0)} + \underbrace{\frac{lm}{n}}_{P(V=1)} \underbrace{H_b\left(\frac{m}{lm}\right)}_{H(V_k|V_{k-1}=1)}, \tag{13}$$

$$H(V_k(\varphi_1)|V_{k-1}(\varphi_1)) = \frac{n-m}{n} H_b\left(\frac{m}{n-m}\right) + \frac{m}{n} H_b(0) \tag{14}$$

where $l \geq 2$, $n \geq l+1$ and $1 \leq m \leq \frac{n}{l+1}$. The last entropy term in eq. (14) is always 0, because the verdict is restricted to always transition back to 0 after transitioning to 1 (i.e. a spike). In fig. 3 we plot eqs. (13) and (14) and their difference

for a fixed n, and various values of l and m. By numerical evaluation, we obtain the worst-case increase in entropy to be ≈ 0.314 bits for the case where $l = 2$. Additionally, we find that for any m, we have $H(V_k(\mathbf{O}_{[a,b]}\varphi_1)|V_{k-1}(\mathbf{O}_{[a,b]}\varphi_1)) > H(V_k(\varphi_1)|V_{k-1}(\varphi_1))$.

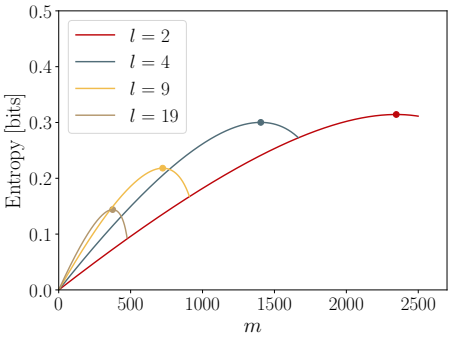


Fig. 3: Increase in conditional entropy for case 1 after temporal aggregation, for different values of l and m. Maximums are indicated with dots.

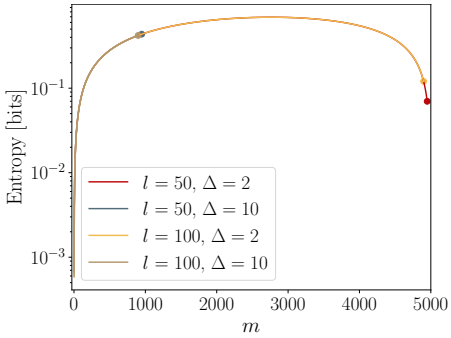


Fig. 4: Reduction in conditional entropy for case 3 after temporal aggregation, for various values of l, Δ and m. End points due to constraints are marked with dots.

Case 2: Due to limitation of space, we provide the specific instantiations of eq. (12) in section B.1. Using numerical evaluation, we find that $P(V(\varphi_1)1) \geq \frac{\frac{n}{2} - \lceil \frac{l}{2} \rceil + 1}{n} \Rightarrow H(V_k(\mathbf{O}_{[a,b]}\varphi_1)|V_{k-1}(\mathbf{O}_{[a,b]}\varphi_1)) < H(V_k(\varphi_1)|V_{k-1}(\varphi_1))$. However, we note that the entropy of both the aggregated and non-aggregated scenarios are nearly indistinguishable with a negligible increase/decrease in entropy.

Case 3: Due to limitation of space, we refer to section B.2 for the entropy expressions. Let Δ denote the fixed time difference between two isolated occurrences of $V(\varphi_1) = 1$ which map to the interval where $V(\mathbf{O}_{[a,b]}\varphi_1) = 1$. We plot the decrease in the entropy after aggregation in fig. 4, which clearly shows that in case 3, the conditional entropy after aggregation is strictly decreased, often by a significant amount. In addition, l and Δ have little impact on the difference in entropy for the same m.

In conclusion, our analysis shows that aggregating temporal properties may reduce or increase the conditional entropy depending on the trace of the subproperty. The trace may however contain variants or combinations of the cases in fig. 2 making it difficult to draw general conclusions.

Temporal Properties and Event-Triggering. Once again, we analyze the cases shown in fig. 2. For case 1, both the aggregated and non-aggregated verdicts will trigger same number of events and for case 2, the number of triggered events also remains the same. In case 3, temporal property aggregation strictly reduces the number of triggered events. In conclusion, the number of events triggered

by $\mathbf{O}_{[a,b]}\varphi_1$ is less than or equal than the number of events triggered by φ_1. An identical conclusion may be drawn for the $\mathbf{H}_{[a,b]}$ operator.

4.2 Property Allocation Problem (AP)

A property allocation $PA(\varphi) = \{\mathcal{P}^i\}_{i=1}^{q}$, is a function which maps an STL property φ to a set containing indexed sets of properties allocated to sensors $1, ..., q$, respectively. As stated in section 2.3, we use verdict traces ($\mathbf{V}(\mathcal{P}) = \langle V_1(\mathcal{P}), ..., V_{\text{end}}(\mathcal{P})\rangle$) as inputs which requires the allocated properties to be sub-properties of the original property φ. To ensure that we *maintain monitorability* of φ, we must also ensure that each sub-property of φ is monitored by at least one sensor.

By assumption 1, every atomic property of φ is monitorable by one sensor. Thus, a simple allocation scheme is to allocate atomic properties to their respective sensors, which preserves monitorability. We call this property allocation $\boldsymbol{PA^p(\varphi)}$. We can use $PA^p(\varphi)$ with any combination of source coding and communication strategies described in section 3.2. However, as we showed in section 4.1, $PA^p(\varphi)$ is not guaranteed to provide the optimal basis for compression.

Property Allocation with Aggregation and Re-Arrangement. To solve **AP**, we must identify the best compressible property allocation, which is one of all possible property allocations that maintain monitorability. While a detailed description of a solution to both these problems are out of scope for this paper, we briefly outline how such a problem could be addressed.

To identify all possible property allocations, we can exploit the semantics of STL and the distributivity and associativity laws of the $\wedge$ and $\vee$ operators, which can be extended to the unary temporal operators. These laws determine how the property can be rearranged, while preserving semantic equivalence. As we are only interested in determining the overall verdict of the property, it suffices to monitor an equivalent property with sub-properties which are more amenable to compression or trigger fewer events. We can re-arrange the property such that sub-properties of the original property which are monitorable by a single sensor, are as large as possible. For example, $\varphi = \varphi_1^1 \wedge ((\mathbf{H}_{[a,b]}\varphi_2^1) \wedge \varphi_1^2)$ can be re-arranged to $(\varphi_1^1 \wedge (\mathbf{H}_{[a,b]}\varphi_2^1)) \wedge \varphi_1^2$, indicating that we can allocate $\mathcal{P}^1 = \{(\varphi_1^1 \wedge \mathbf{H}_{[a,b]}\varphi_2^1)\}$ instead of $\{\mathbf{H}_{[a,b]}\varphi_2^1, \varphi_1^1\}$.

However, as demonstrated in section 4, maximum aggregation does not necessarily correspond to maximum compression. To explore the set of possible property allocations, the sub-properties only containing atomic properties monitorable by a single sensor may be split or kept as a single property. In section 5, we provide results for a property allocation scheme which maximizes aggregation ($\boldsymbol{PA^*(\varphi)}$) and a version without temporal property aggregation.

5 Experimental Evaluation

Setup and Baselines. To evaluate our proposed approaches, we test them on a set of STL properties [22] originally used for falsification testing of a Simulink

automatic transmission system (ATS) model. We generate a dataset containing 1024 unique traces using the ATS model and apply $PA^*(\varphi)$ (property allocation with re-arrangement and maximum aggregation) and $PA^p(\varphi)$ (atomic property allocation) and remove duplicate properties and properties which have the same verdict for all timesteps over all traces. We apply the SMT solver to identify the set of source symbols $\mathcal{V}(\mathcal{P}^i)$ for all sensors, and monitor the allocated properties to determine the conditional distribution from eq. (11) for the generated dataset.

We use two baselines as the reference uncompressed size: the original data size (fig. 1 top) and the size of the raw verdict representations (fig. 1 middle). To compute the original data size, we multiply the trace length with the number of bits needed to represent each value assuming 32-bit representations for floating point values, and for discrete signals, we use $\lceil \log_2(u) \rceil$ bits where u is the number of different values the signal can take. We compute the size of the uncompressed verdicts as the number of properties in each $\mathcal{P}$ resulting from $PA^p(\varphi)$, multiplied by the trace length. However, in this case, event-triggered communication is also employed, if employed in source coding. To refer to specific combinations of reference data size and source coding scheme, we define the function $CR(\mathrm{R}, \mathrm{C})$ which computes the CR (eq. (4)) for a given configuration, where $R \in \{\mathrm{Raw}, \mathrm{Ver}\}$ denotes the reference data size for the raw data and verdicts respectively, and $C \in \{\mathrm{SL}, \mathrm{SF}, \mathrm{SLE}, \mathrm{SFE}\}$ denotes the communication/source coding scheme where SL is periodic communication with stateless source coding, SF is periodic communication with stateful source coding, SLE is event-triggered communication with stateless source coding and SFE is event-triggered communication with stateful source coding. Additionally, we define the degree of average aggregation as $\overline{PA^*}(\varphi) = \frac{1}{\sum_{\mathcal{P} \in PA^*(\varphi)} |\mathcal{P}|} \sum_{\mathcal{P} \in PA^*(\varphi)} \sum_{\varphi_m \in \mathcal{P}} |\varphi_m|$, where $|\varphi|$ is the number of atomic properties and operators in φ.

Numerical Results. We start with investigating results related to **CP** by computing the compression ratios for all combinations of reference data sizes and communication schemes. In fig. 5, we show the CR for all tested properties and configurations. Figure 5 clearly shows that for $CR(\mathrm{Raw}, \cdot)$, the largest compression gains typically come from implementing event-triggered communication, resulting in $CR(\mathrm{Raw}, \mathrm{SLE}), CR(\mathrm{Raw}, \mathrm{SFE}) \in [428, 5960]$ while $CR(\mathrm{Raw}, \mathrm{SL}) \in [11.9, 29.1]$ and $CR(\mathrm{Raw}, \mathrm{SF}) \in [11.9, 32.4]$. This indicates that relatively few events are triggered in the traces. However, for $CR(\mathrm{Ver}, \mathrm{SLE})$ and $CR(\mathrm{Ver}, \mathrm{SFE})$, the compression gains are often smaller under the event-triggering schemes ($CR(\mathrm{Ver}, \mathrm{SLE}) \in [1.0, 2.9]$ and $CR(\mathrm{Ver}, \mathrm{SFE}) \in [1.0, 7.1]$) than schemes without event-triggering. This is because event-triggering already compresses the trace significantly before source coding is applied. Figure 5 also shows that introducing stateful source coding often increases both $CR(\mathrm{Raw}, \cdot)$ and $CR(\mathrm{Ver}, \cdot)$ compared to stateless source coding, although the increase varies between properties.

Figure 6 investigates the correlation between the size of $\mathcal{P}$ ($|\mathcal{P}|$) and $CR(\mathrm{Ver}, \cdot)$ under the 4 presented communication schemes and shows a clear positive correlation between $CR(\mathrm{Ver}, \cdot)$ and $|\mathcal{P}|$. Interestingly, the SL and SLE approaches

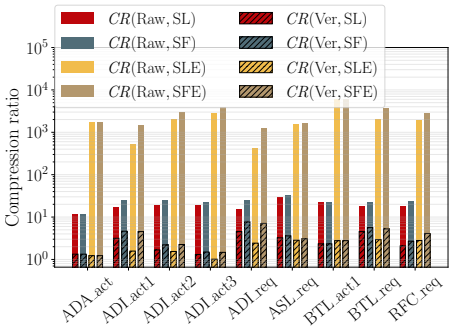


Fig. 5: $CR(\text{Raw}, \cdot)$ and $CR(\text{Ver}, \cdot)$ for all properties under $PA^*(\varphi)$.

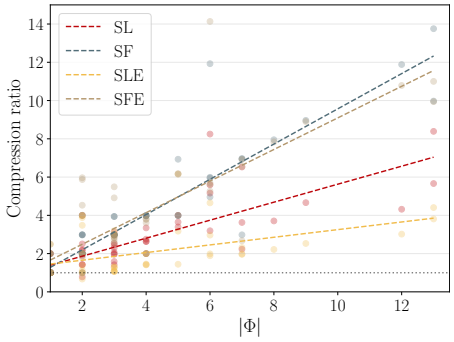


Fig. 6: CR_{ver} per $\mathcal{P}$ in $PA^*(\varphi)$ for all properties and linear fit.

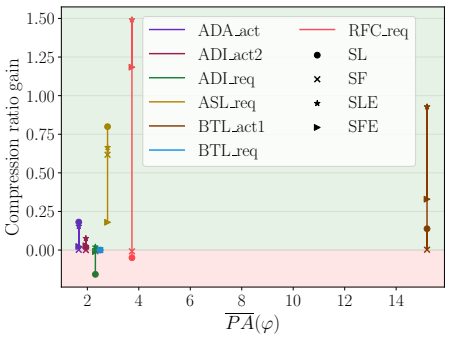


Fig. 7: Difference in $CR(\text{Ver}, \cdot)$ between $PA^*(\varphi)$ and $PA^p(\varphi)$, excluding properties without aggregation. Negative values mean that $PA^p(\varphi)$ results in better compression.

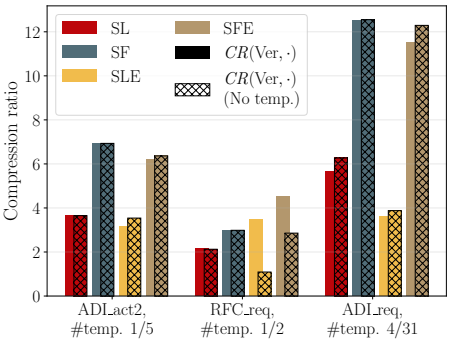


Fig. 8: $CR(\text{Ver}, \cdot)$ with and without temporal property aggregation. Only $\mathcal{P}$ with temporal properties are included.

achieve a smaller CR for larger $|\mathcal{P}|$ compared to the stateful approaches, which are increasingly preferable as $|\mathcal{P}|$ increases.

Next, we present results related to **AP**. In fig. 7, we empirically investigate the effects of property aggregation on compression using the uncompressed verdicts as a baseline. Figure 7 shows that aggregation may result in worse $CR(\text{Ver}, \cdot)$, although the penalty of aggregation is often minor. Conversely, aggregation may also increase $CR(\text{Ver}, \cdot)$, aligning with the theoretical results in section 4.1. Additionally, fig. 7 shows that the benefits of aggregation vary depending on the communication scheme and property, e.g. the SL scheme does not strictly benefit more from aggregation than the SF scheme. All properties with negative gains in fig. 7 contain allocations with temporal properties.

The effects of temporal property aggregation are further investigated in fig. 8. We modify the property allocation such that the immediate sub-properties of the allocated temporal properties, are allocated instead, and compute $CR(\text{Ver}, \cdot)$ values only for $\mathcal{P}$ containing temporal properties. For two of the properties,

aggregation strictly decreases $CR(\text{Ver}, \cdot)$ regardless of scheme, while aggregation increases $CR(\text{Ver}, \cdot)$ for the remaining property under SLE and SFE.

6 Concluding Remarks

We have introduced 4 variants of source coding-based compression for distributed runtime monitoring of temporal logic properties and theoretically analyzed how property aggregation impacts compression. Additionally, we have outlined a property aggregation procedure using logical associativity laws. Among the 4 compression variants, two techniques were introduced to enhance compression; stateful source coding and event-triggered communication, as well as their combination. The proposed methods were evaluated empirically on a set of industry-inspired STL properties, where compression ratios up to 5.96×10^3 were achieved compared to raw data transmission, and up to 32.4 compared to raw verdict transmissions. With raw data, the largest compression gains were associated with event-triggered communication, while the stateful source coding provided the largest compression gains against raw verdicts. We also found a positive correlation between compression ratio on raw verdicts and the number of properties allocated to a sensor, where the stateful source coding approaches showed the strongest correlation. Additionally, we empirically investigated the impact of property aggregation on compression and showed that aggregation often results in compression gains, and in some cases small negative gains under temporal property aggregation, aligning with the theoretical analysis.

While our work focused on STL, we conjecture that the core methods proposed can be adapted to other distributed monitoring techniques with finite-valued monitoring traffic, where we believe source coding is the most generally applicable. Other future work includes formally defining and verifying a property aggregation algorithm to maximize compression.

References

1. ISO 11898-1:2024. https://www.iso.org/standard/86384.html
2. Bartocci, E., Falcone, Y., Francalanza, A., Reger, G.: Introduction to Runtime Verification. In: Bartocci, E., Falcone, Y. (eds.) Lectures on Runtime Verification: Introductory and Advanced Topics, pp. 1–33. Lecture Notes in Computer Science, Springer International Publishing, Cham (2018). https://doi.org/10.1007/978-3-319-75632-5_1
3. Basnet, N., Abbas, H.: Logical Signal Processing: A Fourier Analysis of Temporal Logic. In: Deshmukh, J., Ničković, D. (eds.) Runtime Verification. pp. 359–382. Springer International Publishing, Cham (2020). https://doi.org/10.1007/978-3-030-60508-7_20
4. Bataineh, O., Rosenblum, D.S., Reynolds, M.: Efficient Decentralized LTL Monitoring Framework Using Tableau Technique. ACM Transactions on Embedded Computing Systems **18**(5s), 1–21 (Oct 2019). https://doi.org/10.1145/3358219
5. Bataineh, O., Rosenblum, D.S., Reynolds, M.: A Novel Decentralized LTL Monitoring Framework Using Formula Progression Table. In: Biondi, F., Given-Wilson, T., Legay, A. (eds.) Model Checking Software. pp. 38–55. Springer International Publishing, Cham (2019). https://doi.org/10.1007/978-3-030-30923-7_3

6. Bauer, A., Falcone, Y.: Decentralised LTL Monitoring. In: Giannakopoulou, D., Méry, D. (eds.) FM 2012: Formal Methods. pp. 85–100. Springer, Berlin, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32759-9_10
7. Bufo, S., Bartocci, E., Sanguinetti, G., Borelli, M., Lucangelo, U., Bortolussi, L.: Temporal Logic Based Monitoring of Assisted Ventilation in Intensive Care Patients. In: Margaria, T., Steffen, B. (eds.) Leveraging Applications of Formal Methods, Verification and Validation. Specialized Techniques and Applications, vol. 8803, pp. 391–403. Springer Berlin Heidelberg, Berlin, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45231-8_30
8. Chalupa, M., Henzinger, T.A., Saraç, N.E., Yu, E.: Quantitative Monitoring of Signal First-Order Logic. In: Sampaio, A., Stoelinga, M. (eds.) Formal Methods. pp. 214–233. Springer Nature Switzerland, Cham (2026). https://doi.org/10.1007/978-3-032-26220-2_11
9. Colombo, C., Falcone, Y.: Organising LTL monitors over distributed systems with a global clock. Formal Methods in System Design **49**(1), 109–158 (Oct 2016). https://doi.org/10.1007/s10703-016-0251-x
10. Cover, T.M., Thomas, J.A.: Elements of Information Theory. Wiley-Interscience, Hoboken, N.J, 2nd ed edn. (2006)
11. De Moura, L., Bjørner, N.: Z3: An efficient SMT solver. In: Proceedings of the Theory and Practice of Software, 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems. pp. 337–340. TACAS'08/ETAPS'08, Springer-Verlag, Berlin, Heidelberg (Mar 2008)
12. Donzé, A., Ferrère, T., Maler, O.: Efficient Robust Monitoring for STL. In: Sharygina, N., Veith, H. (eds.) Computer Aided Verification. pp. 264–279. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39799-8_19
13. El-Hokayem, A., Falcone, Y.: On the Monitoring of Decentralized Specifications: Semantics, Properties, Analysis, and Simulation. ACM Transactions on Software Engineering and Methodology **29**(1), 1–57 (Jan 2020). https://doi.org/10.1145/3355181
14. Falcone, Y.: On Decentralized Monitoring. In: Nouri, A., Wu, W., Barkaoui, K., Li, Z. (eds.) Verification and Evaluation of Computer and Communication Systems. pp. 1–16. Springer International Publishing, Cham (2022). https://doi.org/10.1007/978-3-030-98850-0_1
15. Falcone, Y., Cornebize, T., Fernandez, J.C.: Efficient and Generalized Decentralized Monitoring of Regular Languages. In: Ábrahám, E., Palamidessi, C. (eds.) Formal Techniques for Distributed Objects, Components, and Systems. pp. 66–83. Springer, Berlin, Heidelberg (2014). https://doi.org/10.1007/978-3-662-43613-4_5
16. Francalanza, A., Pérez, J.A., Sánchez, C.: Runtime Verification for Decentralised and Distributed Systems. In: Bartocci, E., Falcone, Y. (eds.) Lectures on Runtime Verification, vol. 10457, pp. 176–210. Springer International Publishing, Cham (2018). https://doi.org/10.1007/978-3-319-75632-5_6
17. Ganguly, R., Kazemloo, S., Bonakdarpour, B.: Crash-Resilient Decentralized Synchronous Runtime Verification. IEEE Transactions on Dependable and Secure Computing **21**(3), 1017–1031 (May 2024). https://doi.org/10.1109/TDSC.2023.3265566
18. Ganguly, R., Momtaz, A., Bonakdarpour, B.: Distributed Runtime Verification Under Partial Synchrony. LIPIcs, Volume 184, OPODIS 2020 **184**, 20:1–20:17 (2021). https://doi.org/10.4230/LIPICS.OPODIS.2020.20

19. Ganguly, R., Xue, Y., Jonckheere, A., Ljung, P., Schornstein, B., Bonakdarpour, B., Herlihy, M.: Distributed Runtime Verification of Metric Temporal Properties for Cross-Chain Protocols. In: 2022 IEEE 42nd International Conference on Distributed Computing Systems (ICDCS). pp. 23–33 (Jul 2022). https://doi.org/10.1109/ICDCS54860.2022.00012
20. Huffman, D.A.: A Method for the Construction of Minimum-Redundancy Codes. Proceedings of the IRE **40**(9), 1098–1101 (Sep 1952). https://doi.org/10.1109/JRPROC.1952.273898
21. Koymans, R.: Specifying real-time properties with metric temporal logic. Real-Time Systems **2**(4), 255–299 (Nov 1990). https://doi.org/10.1007/BF01995674
22. Lidén Eddeland, J., Donzé, A., Miremadi, S., Åkesson, K.: Industrial Temporal Logic Specifications for Falsification of Cyber-Physical Systems. In: ARCH20. 7th International Workshop on Applied Verification of Continuous and Hybrid Systems (ARCH20). pp. 267–258. https://doi.org/10.29007/r74f
23. Maler, O., Nickovic, D.: Monitoring Temporal Properties of Continuous Signals. In: Lakhnech, Y., Yovine, S. (eds.) Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems. pp. 152–166. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg (2004). https://doi.org/10.1007/978-3-540-30206-3_12
24. Moffat, A.: Huffman Coding. ACM Comput. Surv. **52**(4), 85:1–85:35 (Aug 2019). https://doi.org/10.1145/3342555
25. Momtaz, A., Abbas, H., Bonakdarpour, B.: Monitoring Signal Temporal Logic in Distributed Cyber-physical Systems. In: Proceedings of the ACM/IEEE 14th International Conference on Cyber-Physical Systems (with CPS-IoT Week 2023). pp. 154–165. ACM, San Antonio TX USA (May 2023). https://doi.org/10.1145/3576841.3585937
26. Mostafa, M., Bonakdarpour, B.: Decentralized Runtime Verification of LTL Specifications in Distributed Systems. In: 2015 IEEE International Parallel and Distributed Processing Symposium. pp. 494–503 (May 2015). https://doi.org/10.1109/IPDPS.2015.95
27. Pnueli, A.: The temporal logic of programs. In: 18th Annual Symposium on Foundations of Computer Science (Sfcs 1977). pp. 46–57 (Oct 1977). https://doi.org/10.1109/SFCS.1977.32
28. Raman, V., Maasoumy, M., Donzé, A.: Model predictive control from signal temporal logic specifications: A case study. In: Proceedings of the 4th ACM SIGBED International Workshop on Design, Modeling, and Evaluation of Cyber-Physical Systems. pp. 52–55. CyPhy '14, Association for Computing Machinery, New York, NY, USA (Apr 2014). https://doi.org/10.1145/2593458.2593472
29. Sahin, Y.E., Quirynen, R., Cairano, S.D.: Autonomous Vehicle Decision-Making and Monitoring based on Signal Temporal Logic and Mixed-Integer Programming. In: 2020 American Control Conference (ACC). pp. 454–459 (Jul 2020). https://doi.org/10.23919/ACC45564.2020.9147917
30. Vitter, J.S.: Design and analysis of dynamic Huffman coding. In: 26th Annual Symposium on Foundations of Computer Science (Sfcs 1985). pp. 293–302 (Oct 1985). https://doi.org/10.1109/SFCS.1985.18
31. Yamaguchi, T., Hoxha, B., Ničković, D.: RTAMT – Runtime Robustness Monitors with Application to CPS and Robotics. International Journal on Software Tools for Technology Transfer (Oct 2023). https://doi.org/10.1007/s10009-023-00720-3

32. Zhang, F., Kelly, K.: Overview and Recommendations for Cyber Risk Assessment in Nuclear Power Plants. Nuclear Technology **209**(3), 488–502 (Mar 2023). https://doi.org/10.1080/00295450.2022.2092356

A Proof of Theorem 1

We use Lemma 1 to prove Theorem 1.

Lemma 1. $2 \leq |\mathcal{V}(\{\varphi_1, \varphi_2\})| \leq 4$ *where* φ_1, φ_2 *are arbitrary well-formed STL properties.*

Proof. Under Assumption 2, φ_1 and φ_2 can be both violated and satisfied by at least one trace. If the two properties have no relation, e.g. the satisfaction of φ_1 does not restrict φ_2 in any way and vice versa, then we have at most $2^2 = 4$ possible combinations which proves the upper bound. If on the other hand we have that $\varphi_1 \leftrightarrow \varphi_2$, the two properties are equivalent and $\mathcal{V}(\{\varphi_1, \varphi_2\}) = \{00, 11\}$. Alternatively, it may be the case that $\neg\varphi_1 \rightarrow \varphi_2$ and $\neg\varphi_2 \rightarrow \varphi_1$ and consequently $\mathcal{V}(\{\varphi_1, \varphi_2\}) = \{01, 10\}$, which concludes the proof. □

Proof (Theorem 1). Any STL operator maps a specific verdict combination of its sub-properties to an aggregated binary verdict. For brevity, let $\mathcal{V}(\{\varphi_1, \varphi_2\})^{\models} = \{v : v \in \mathcal{V}(\{\varphi_1, \varphi_2\}) \wedge v \rightarrow V(\varphi_1 \triangle^b \varphi_2) = \text{True}\}$ and $\mathcal{V}(\{\varphi_1, \varphi_2\})^{\not\models} = \{v : v \in \mathcal{V}(\{\varphi_1, \varphi_2\}) \wedge v \rightarrow V(\varphi_1 \triangle^b \varphi_2) = \text{False}\}$ Then, we have

$$P(V(\varphi_1 \triangle^b \varphi_2) = 1) = \sum_{v \in \mathcal{V}(\{\varphi_1, \varphi_2\})^{\models}} P(V(\{\varphi_1, \varphi_2\}) = v),$$
$$P(V(\varphi_1 \triangle^b \varphi_2) = 0) = \sum_{v \in \mathcal{V}(\{\varphi_1, \varphi_2\})^{\not\models}} P(V(\{\varphi_1, \varphi_2\}) = v)$$

Applying the definition of entropy we have:

$$H(V(\varphi_1 \triangle^b \varphi_2)) = -\left[\sum_{v \in \mathcal{V}(\{\varphi_1, \varphi_2\})^{\models}} P(v)\right] \log\left[\sum_{v \in \mathcal{V}(\{\varphi_1, \varphi_2\})^{\models}} P(v)\right]$$
$$-\left[\sum_{v \in \mathcal{V}(\{\varphi_1, \varphi_2\})^{\not\models}} P(v)\right] \log\left[\sum_{v \in \mathcal{V}(\{\varphi_1, \varphi_2\})^{\not\models}} P(v)\right]$$

For both $\mathcal{V}(\{\varphi_1, \varphi_2\})^{\models}$ and $\mathcal{V}(\{\varphi_1, \varphi_2\})^{\not\models}$ we can factor the log terms as follows

$$\begin{aligned}\log[P(v_1) + ... + P(v_n)] &= \log\left[P(v_1)(1 + P(v_2)/P(v_1) + ... + P(v_n)/P(v_1))\right] \\ &= \log P(v_1) + \log\left[1 + P(v_2)/P(v_1) + ... + P(v_n)P(v_1)\right],\end{aligned}$$

where $n = |\mathcal{V}(\{\varphi_1, \varphi_2\})^{\models}|$ or $|\mathcal{V}(\{\varphi_1, \varphi_2\})^{\not\models}|$ depending on the scenario. For every term in the sum outside the log, we can factor out the corresponding term

in the sum inside the log expression, e.g.

$$P(v_1)(\log P(v_1) + \log\left[1 + P(v_2)/P(v_1) + ... + P(v_n)/P(v_1)\right]) \\ + ... + P(v_n)(\log P(v_n) + \log\left[1 + P(v_1)/P(v_n) + ...\right]).$$

Thus, we can factor out the entropy of $H(V(\{\varphi_1, \varphi_2\})) = \sum_{v \in \mathcal{V}(\{\varphi_1,\varphi_2\})} P(v) \log P(v)$ such that:

$$\begin{aligned} H(V(\varphi_1 \triangle^b \varphi_2)) = \; & H(V(\{\varphi_1, \varphi_2\})) \\ & - \sum_{v \in \mathcal{V}(\{\varphi_1,\varphi_2\})^{\models}} P(v) \log(1 + \frac{\sum_{v' \in \mathcal{V}(\{\varphi_1,\varphi_2\})^{\models} \setminus \{v\}} P(v')}{P(v)}) \\ & - \sum_{v \in \mathcal{V}(\{\varphi_1,\varphi_2\})^{\not\models}} P(v) \log(1 + \frac{\sum_{v' \in \mathcal{V}(\{\varphi_1,\varphi_2\})^{\not\models} \setminus \{v\}} P(v')}{P(v)}) \end{aligned} \tag{15}$$

Since the the sum of last two terms on the right-hand side are ≤ 0 in any case, we have proven $H(\mathcal{V}(\varphi_1 \triangle^b \varphi_2)) \leq H(\mathcal{V}(\{\varphi_1, \varphi_2\}))$. To prove the strict inequality, we must prove that either or both sums inside the log are non-zero. For the case where $|\mathcal{V}(\{\varphi_1, \varphi_2\})| = 2$, since $\varphi_1 \neq \varphi_2$, the only option for $\mathcal{V}(\{\varphi_1, \varphi_2\})$ is $\{01, 10\}$ as per the proof of Lemma 1, resulting in at least two verdicts in either $\mathcal{V}(\{\varphi_1, \varphi_2\})^{\models}$ or $\mathcal{V}(\{\varphi_1, \varphi_2\})^{\not\models}$. Since $\forall v \in \mathcal{V}(\{\varphi_1, \varphi_2\})\ P(V = v) > 0$, the sum of the last two terms on the right-hand side are negative, thus upholding the inequality in this case.

For the case where $|\mathcal{V}(\{\varphi_1, \varphi_2\})| > 2$, we see that at least two cases must map to one of the two binary verdicts, and since $\forall v \in \mathcal{V}(\{\varphi_1, \varphi_2\})\ P(V = v) > 0$, at least one of the sums inside the log are > 0 thus upholding the inequality. □

B Conditional entropy expressions for temporal operators

B.1 Case 2

$$\begin{aligned} & H(V_k(\mathbf{O}_{[a,b]}\varphi) | V_{k-1}(\mathbf{O}_{[a,b]}\varphi)) = \\ & \qquad \frac{n-(l+m)}{n} H_b\left(\frac{1}{n-(l+m)}\right) + \frac{l+m}{n} H_b\left(\frac{1}{l+m}\right), \end{aligned} \tag{16}$$

$$H(V_k(\varphi) | V_{k-1}(\varphi)) = \frac{n-m}{n} H_b\left(\frac{1}{n-m}\right) + \frac{m}{n} H_b\left(\frac{1}{m}\right) \tag{17}$$

where $l > 2, n \geq l+1, l+m < n$. When $l+m \geq n$, we note that $P(V(\mathbf{O}_{[a,b]}\varphi) = 1) = 1$.

B.2 Case 3

$$H(V_k(\mathbf{O}_{[a,b]}\varphi)|V_{k-1}(\mathbf{O}_{[a,b]}\varphi)) = \frac{n-(l+m\Delta)}{n} H_b\left(\frac{1}{n-(l+m\Delta)}\right) + \frac{l+m\Delta}{n} H_b\left(\frac{1}{l+m\Delta}\right), \quad (18)$$

$$H(V_k(\varphi)|V_{k-1}(\varphi)) = \frac{n-m}{n} H_b\left(\frac{m}{n-m}\right). \quad (19)$$

where $l > 2, n \geq l + \Delta + 1, m < n/2$ and $\Delta \geq 2$ is the time difference between occurrences of $V(\varphi) = 1$ which map to the interval where $V(\mathbf{O}_{[a,b]}\varphi) = 1$.