Learning Optimal Signal Temporal Logic Decision Trees for Classification: A Max-Flow MILP Formulation

Kaier Liang, Gustavo A. Cardona, Disha Kamale, and Cristian-Ioan Vasile

Abstract— **This paper presents a novel framework for inferring timed temporal logic properties from data. The dataset comprises pairs of finite-time system traces and corresponding labels, denoting whether the traces demonstrate specific desired behaviors, e.g. whether the ship follows a safe route or not. Our proposed approach leverages decision-tree-based methods to infer Signal Temporal Logic classifiers using primitive formulae. We formulate the inference process as a mixed integer linear programming optimization problem, recursively generating constraints to determine both data classification, i.e., decision criteria and the tree structure. Applying a max-flow algorithm on the resultant tree transforms the problem into a global optimization challenge, leading to improved classification rates compared to prior methodologies. Moreover, we introduce a technique to reduce the number of constraints by exploiting the symmetry inherent in STL primitives, which enhances the algorithm's time performance and interpretability. We conduct three case studies involving two-class, multi-class, and complex formula classification scenarios to assess our algorithm's effectiveness and classification performance.**

I. Introduction

The ever-increasing complexity of modern systems has resulted in the need for sophisticated techniques that can help understand and classify temporal behaviors from time-series data. Machine learning (ML) [1] approaches have emerged as effective tools for this task, particularly in two-class classification, where the objective is to differentiate between desired and undesired system behaviors. However, the opaque nature of traditional ML algorithms often hinders interpretability and insight into system dynamics [2], [3]. To address this limitation, formal methods such as Signal Temporal Logic (STL) [4], [5] have been gaining traction to specify temporal properties of real-valued signals. STL provides a structured language for expressing complex temporal and logical behaviors, offering both readability and interoperability. By formulating temporal properties as logical formulae, STL facilitates the inference of system behaviors from labeled time-series data [6].

While early methods in this domain focused on manual parameter synthesis from predefined template formulae [7]–[10], recent advancements have sought to automate this process by inferring both the structure and parameters of STL formulae directly from data [11]. The authors of [12] introduced a fragment of STL called inference parametric signal temporal logic (iPSTL), which enables the classification problem to be formulated as an optimization problem. However, this approach faces challenges such as high computational cost due to nonlinear parameter optimization routines and constructing a directed acyclic graph (DAG) based on the ordering of PSTL formulae, which may not necessarily improve classification performance. Other recent works, such as [13], [14], have addressed the two-class classification problem by building generative models for each class and deriving a discriminative formula that maximizes the probability of satisfaction for one model while minimizing it for the other. Despite their potential, these methods require building models of the system under analysis, which entails domain expertise and substantial data. On the other hand, decision tree-based frameworks have emerged as promising approaches for efficiently learning STL formulae, where each node encapsulates simple formulae optimized from a predefined set of primitives [15], [16]. Lastly, the authors in [17] use the different layers of neural networks to learn different aspects of STL formulae that allow a compact and efficient binary interpretable classification for level-one STL formulae, and multi-class inference [18], [19].

Despite recent advances, the formulae generated by existing algorithms often suffer from verbosity and complexity, limiting their interpretability and practicality in real-world scenarios. In response to these challenges, this paper introduces an innovative approach: optimal max-flow tree data classification using Signal Temporal Logic (STL) inference. Our method builds upon the concept of tree-based structures [15], [16], to encode STL primitives recursively and facilitate the classification of labeled datasets. However, unlike previous approaches, we leverage this tree structure to formulate a max-flow optimization problem [20], enabling us to determine the tree structure and the inferred STL specification simultaneously. At each node of the tree, we encapsulate STL primitives and make branching decisions based on the classification of the input data. Our approach offers several advantages over previous methods by formulating the classification task as a mixed integer linear programming (MILP) problem. Unlike [15], [16], ours enables global optimization and can be adapted for multi-class classification scenarios. Moreover, we reduce primitive redundancy, resulting in fewer constraints, which improves time performance and reduces complexity. In contrast to [17], our method considers a broader set of STL primitives and avoids the pitfalls associated with nonlinear optimization, such as getting trapped in local minima. Our approach ensures optimal performance with enhanced interpretability by maximizing classification accuracy and minimizing formula complexity. We validate the efficacy of our method using two-class and multi-class classification scenarios. The results demonstrate significantly improved classification performance and interpretability compared to existing approaches, thereby paving the way for more effective temporal behavior classification and analysis. The main contributions of this work are, 1) Proposing a novel decision tree-based STL inference

Kaier Liang, Gustavo A. Cardona, Disha Kamale, and Cristian-Ioan Vasile are with the Mechanical Engineering and Mechanics Department at Lehigh University, PA, USA: {kal221, gcardona, ddk320, cvasile}@lehigh.edu

algorithm that runs a global optimization and results in an improved classification rate compared to the existing related approaches. 2) Formulating a MILP encoding that classifies data and generates the rules for growing the tree structure based on STL primitives, followed by a global max-flow optimization approach that guarantees high-performance classification. 3) Constraint reduction based on the symmetry of STL primitives, resulting in more efficient and improved time performance. 4) Demonstrating three case studies to show the algorithm's capability to handle two-class, multi-class, and complex formulae classification and compare its performance with other approaches.

II. Preliminaries and Notation

Let $\mathbb{R}$ denote the set of all real numbers, $\mathbb{Z}$ the set of integers, $\mathbb{B}$ the binary set, and $\mathbb{Z}_{\geq 0}$ the set of non-negative integers. For a set $\mathcal{S}$, $2^{\mathcal{S}}$ and $|\mathcal{S}|$ represent its power set and cardinality. We have $\alpha + S = \{\alpha + x \mid x \in S\}$. The integer interval (range) from a to b is $[a..b]$. We use $\underline{I} = a$ and $\overline{I} = b$. A discrete-time signal s, with time horizon $H \in \mathbb{Z}_{\geq 0}$, is defined as a function $s : [0..H] \to \mathbb{R}^d$ mapping each time-step to an d-dimensional vector of real values. The j-th component of x is given by x_j, $j \in [1..d]$.

Signal Temporal Logic: Consider a discrete-time signal $s : [0..H] \to \mathbb{M}$ with values in the compact space $\mathbb{M} \subseteq \mathbb{R}^d$. Signal Temporal Logic (STL) [4] is a specification language that expresses real-time properties with the following syntax.

$$\phi ::= \top \mid h(s) \geq \pi \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \mid \Diamond_I \phi \mid \Box_I \phi, \quad (1)$$

where ϕ, ϕ_1, and ϕ_2 are STL formulae, $\top$ is the logical *True* value, $h(s) \geq \pi$ is a predicate with $h : \mathbb{R}^d \to \mathbb{R}$ and threshold value $\pi \in \mathbb{R}$, $\neg$, $\wedge$, and $\vee$ are the Boolean negation, conjunction, and disjunction operators. Discrete-time temporal operators *eventually* $\Diamond_I$, and *always* $\Box_I$ with $I = [\underline{I}..\overline{I}]$ a discrete-time interval, $\overline{I} \geq \underline{I} \geq 0$. Linear predicates of the form $s \sim \pi$, $\sim \in \{>, \leq, <\}$, follow via negation and sign change. The qualitative semantics are defined as in [4]. The quantitative semantics (robustness) $\rho(s, \phi, k)$ indicate how much a signal satisfies or violates a specification [21] it is recursively defined as

$$\begin{aligned} \rho(s, h(s) \geq \pi, k) &= h(s(k)) - \pi, \\ \rho(s, \phi_1 \wedge \phi_2, k) &= \min(\rho(s, \phi_1, k), \rho(s, \phi_2, k)), \\ \rho(s, \phi_1 \vee \phi_2, k) &= \max(\rho(s, \phi_1, k), \rho(s, \phi_2, k)), \\ \rho(s, \Box_I \phi, k) &= \min_{k' \in k+I} \rho(s, \phi, k'), \\ \rho(s, \Diamond_I \phi, k) &= \max_{k' \in k+I} \rho(s, \phi, k'), \end{aligned} \quad (2)$$

where $\rho_\top = \sup_{s,\pi}\{|h(s) - \pi|\}$ is the maximum robustness.

Theorem 1 (Soundness [22])**.** *Let s be a signal and ϕ an STL formula. It holds $\rho(s, \phi, k) > 0 \Rightarrow (s, k) \models \phi$ for satisfaction and $\rho(s, \phi, k) < 0 \Rightarrow (s, k) \not\models \phi$ for violation.*

The time horizon of an STL formula is captured as in [23].

III. Problem Formulation

This section introduces the data classification problem by inferring STL specifications. Consider a labeled dataset $\mathcal{S} := \{(s^i, \ell^i)\}_{i \in \mathcal{I}}$, where s^i and ℓ^i represent the i-th signal and its corresponding label. We denote the set of all possible classification classes as $\mathcal{C} = [1..|\mathcal{C}|]$, where $\ell^i = c$ signifies that the i-th sample is labeled with class $c \in \mathcal{C}$. Our aim is to infer STL formulae ϕ_c that encapsulate properties inherent to each data class $c \in \mathcal{C}$. Formally,

Problem 1 (Multi-class classification)**.** *Given a labeled data set $\mathcal{S}$, find a set of mutually-exclusive STL formulae $\Phi = \{\phi_c\}_{c \in \mathcal{C}}$ that maximizes the correct classification rate $CCR(\Phi)$ where*

$$CCR(\Phi) := \frac{\sum_{c \in \mathcal{C}} \left|\{s^i \mid s^i \models \phi_c \wedge \ell^i = c\}\right|}{|\mathcal{I}|}. \quad (3)$$

such that $\phi_c \wedge \phi_{c'} \equiv \bot$ for all $c, c' \in \mathcal{C}$ with $c \neq c'$.

IV. Solution

We propose an approach to learning STL formulae using decision trees. The tree branches from the root to the leaves correspond to the classification process, where leaves are associated with classes and intermediate nodes with decisions in primitive STL formulae. We infer the structure of the decision tree, the primitives to use for each decision node and their spatial and temporal parameters, and the classes associated with the leaf nodes such that the correct classification rate is maximized and the number of decision nodes is minimized. We cast the temporal inference problem as a MILP problem using a max flow encoding [20].

Our classification method employs a tree-based structure [20], akin to a binary decision tree but with a unique feature: it incorporates a source node $\mathbf{s}$ and a sink node $\mathbf{t}$, as illustrated in Fig. 1. Furthermore, each node is connected to the sink node. We refer to this configuration as a *classification tree*, defined as follows.

Definition 1 (Classification Tree)**.** *A classification tree $\mathcal{T} = (\mathcal{N}, \mathcal{L}, \mathcal{E}, \mathbf{s}, \mathbf{t})$ is a Directed Acyclic Graph (DAG) with single source $\mathbf{s}$ and single sink $\mathbf{t}$. Nodes in the final layer of the tree are referred to as leaf nodes, denoted by $\mathcal{L}$, while nodes situated between the source and the leaf nodes are internal nodes, denoted by $\mathcal{N}$ and $\mathcal{E} \subseteq (\mathcal{N} \cup \{\mathbf{s}\}) \times (\mathcal{N} \cup \mathcal{L} \cup \{\mathbf{t}\})$ captures relation between nodes. We define the sets $\mathrm{pa}(n) = \{n' \mid (n', n) \in \mathcal{E}\}$ and $\mathrm{ch}(n) = \{n' \mid (n, n') \in \mathcal{E}\}$ to represent the parent and children nodes of a given node n, respectively.*

Note that in Fig. 1, $\mathrm{pa}(\mathbf{s}) = \varnothing$, since sink has no parent node and $\mathrm{pa}(1) = \mathbf{s}$[1]. On the other hand, $\mathrm{ch}(\mathbf{t}) = \varnothing$ and for all $n \in \mathcal{L}$ we have $\mathrm{ch}(n) = \{\mathbf{t}\}$. An internal node $n \in \mathcal{N}$ has three children, a left child, a right child, and the sink $\mathbf{t}$. In the classification process using $\mathcal{T}$, data samples that are correctly classified traverse from the source node through the tree to reach the sink. Conversely, misclassified samples are blocked from progressing beyond the source node. The binary decision tree (BDT) with STL primitive nodes is extracted from the classification tree $\mathcal{T}$. The root of the STL BDT is the node connected to the source $\mathbf{s}$, i.e., node 1 in Fig. 1. Not all nodes $\mathcal{N} \cup \mathcal{L}$ of $\mathcal{T}$ are part of the inferred STL BDT. The final structure follows from the optimization result. Note that a pre-determined depth value limits the size of the classification tree.

To partition the data at each node $n \in \mathcal{N}$, a finite list of possible splitting rules is considered [24]. We use simple

[1]We abuse notation and write $\mathrm{pa}(n) = n'$ instead of $\mathrm{pa}(n) = \{n'\}$ for readability since a node has at most one parent.

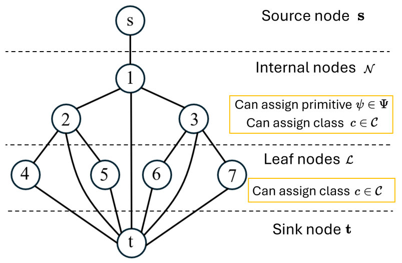

s
Source node $\mathbf{s}$
Internal nodes $\mathcal{N}$
1
2
3
Can assign primitive $\psi \in \Psi$
Can assign class $c \in \mathcal{C}$
Leaf nodes $\mathcal{L}$
4
5
6
7
Can assign class $c \in \mathcal{C}$
Sink node $\mathbf{t}$
t

Fig. 1: Max flow classification tree.

Parametric Signal Temporal Logic (PSTL) formulae [7], called primitives [16], defined as follows.

Definition 2 (PSTL primitives)**.** *The primitive in the set Ψ consists of a temporal operator, a relational operator, and a predicate function. Formally, $\Psi = \{\Gamma(h(s) \sim \pi) \mid \Gamma \in \{\Box_{I_1}, \Diamond_{I_1}, \Box_{I_1}\Diamond_{I_2}, \cdots\}, \sim \in \{\geq, <\}, h \in \mathcal{H}\}$, where $\pi \in \mathbb{R}$ is the threshold, I_j are integer ranges of temporal operators, and $\mathcal{H}$ is a finite set of predicate functions $h : [0..H] \to \mathbb{M}$. The set of all possible time intervals of temporal operators is $\Theta = \{\theta = (I_1, \ldots, I_q) \mid I_p = [\underline{I}_p..\bar{I}_p] \mid \underline{I}_p \leq \bar{I}_p$, with $\underline{I}, \bar{I} \in [0..H], \forall p \in [1..q], q \in \mathbb{Z}_{\geq 1}\}$, where q is the number of temporal operators in a primitive.*

In Def. 2, π is the *spatial parameter*, while the time bounds of temporal operators I_p are the *time parameters*.

When classifying using tree $\mathcal{T}$, we assign a primitive $\psi \in \Psi$ or a class $c \in \mathcal{C}$ to each node n in $\mathcal{N}$ and $\mathcal{L}$. Note that a class $c \in \mathcal{C}$ can be assigned at internal or leaf nodes, whereas primitives can only be assigned to internal nodes. For internal nodes $n \in \mathcal{N}$, we also compute the spatial and time parameters of primitive ψ, denoted as π_n and $\theta_n = (I_1, \ldots, I_{q_\psi})$, where q_ψ is the number of temporal operators in ψ.

For each data sample s^i in the dataset $\mathcal{S}$, the following encoding captures the classification of s^i as a flow from the source $\mathbf{s}$ to the sink $\mathbf{t}$. At the starting node $\mathbf{s}$, s^i flows down to the top node 1, see Fig. 1. If the node n is a *decision node* associated with the STL formula $\phi_n = \psi_n(\pi_n, \theta_{\psi_n})$, then the check $s^i \models \phi_n$ is performed. If signal s^i satisfies ϕ_n, the signal flows to the left child $l(n) \in \text{ch}(n)$. Otherwise, it flows to the right child $r(n) \in \text{ch}(n)$. The link to the sink node $\mathbf{t}$ is not used for decision nodes. The procedure continues from the node reached by s^i. If the node n is a *classification node* associated with class c_n, the check $c_n = \ell^i$ is performed. If the label ℓ^i matches the class c_n, then the signal s^i flows to the sink $\mathbf{t}$ and is correctly classified and counted towards the objective. If a signal cannot be correctly classified, it does not enter the classification tree and, thus, does not count toward the objective as explained later in Sec. IV. In either case, the classification of s^i ends when it reaches the sink $\mathbf{t}$. For classification nodes, the links to the right and left nodes are unused. Moreover, leaves $\mathcal{L}$ can only be classification nodes since they have edges only to the sink $\mathbf{t}$.

Hence, to achieve accurate data classification, we aim to construct a classification tree $\mathcal{T}$ that maximizes the Correct Classification Rate ($CCR(\Phi)$). Consequently, the flow-based optimization problem solving Pb. 1 is:

$$\begin{aligned} \max \quad & CCR(\Phi) \\ \text{s.t.} \quad & \text{Flow constraints, Node function constraints,} \\ & \text{STL satisfaction constraints.} \end{aligned} \tag{4}$$

The objective is to maximize the Correct Classification Rate (CCR) while adhering to *flow constraints* that ensure the conservation of data flow within the tree. Constraints related to *node functions* define primitive predicates, child allocation, and classification. The final soundness constraint ensures that the predicted class satisfies the signal. The following sections provide a detailed description of these constraints.

***Flow constraints**:* Here, we establish the conservation of data flow within the classification tree $\mathcal{T}$. Let $z_n^i \in \mathbb{B}$ denote a binary variable representing whether data point $(s^i, \ell^i) \in \mathcal{S}$ traverses node $n \in \mathcal{N} \cup \mathcal{L}$. It takes a value of one if data point (s^i, ℓ^i) flows through node n and zero otherwise. We designate the entry of a data point into the source node as $z_{\mathbf{s}}^i \in \mathbb{B}$. Since all internal nodes are connected to a common sink node $\mathbf{t}$, we introduce $z_{\mathbf{t},n}^i \in \mathbb{B}$ which also identifies the node through which data point (s^i, ℓ^i) enters the sink node $\mathbf{t}$. Specifically, $z_{\mathbf{t},n}^i = 1$ indicates that data (s^i, ℓ^i) reaches the sink node via node n, and zero otherwise. Then, the flow constraints are defined as follows

$$z_{\mathbf{s}}^i = z_1^i, \quad \forall i \in \mathcal{I}, \tag{5a}$$

$$z_n^i = z_{\mathbf{t},n}^i + z_{l(n)}^i + z_{r(n)}^i, \quad \forall n \in \mathcal{N}, \forall i \in \mathcal{I}, \tag{5b}$$

$$z_n^i = z_{\mathbf{t},n}^i, \quad \forall n \in \mathcal{L}, \forall i \in \mathcal{I}. \tag{5c}$$

The constraint (5a) enforces a flow from the source node $\mathbf{s}$ to enter node $n = 1$ corresponding to the root of the inferred BDT that eventually reaches sink $\mathbf{t}$ and is correctly classified. In this case, $z_{\mathbf{s}}^i = z_1^i = 1$. Otherwise, we have $z_{\mathbf{s}}^i = z_1^i = 0$, and there is no flow for s^i, and it is thus misclassified. Next, (5b) ensures that data s^i leaving internal node $n \in \mathcal{N} \cup \mathcal{L}$ must enter one of its children nodes or the sink node $\mathbf{t}$. The last equation (5c) enforces the flow for s^i from leaves $\mathcal{L}$ to the sink $\mathbf{t}$, which is their only child. These constraints imply that data is either correctly classified, i.e., $z_{\mathbf{t},n}^i = 1$ for a node $n \in \mathcal{N} \cup \mathcal{L}$ via a deterministic tree flow, or it does not enter the tree, i.e., $z_{\mathbf{s}}^i = 0$.

***Node function constraints**:* In this section, we capture the functionality of the nodes in $\mathcal{T}$. Internal nodes $\mathcal{N}$ are either decision or classification nodes. Each internal node $n \in \mathcal{N}$ in $\mathcal{T}$ either checks a primitive STL formula and creates child nodes or classifies data by directing the flow to the sink. Leaf nodes $n \in \mathcal{L}$ exclusively make classifications by enforcing the flow to the sink $\mathbf{t}$. For each node $n \in \mathcal{N}$, we introduce the binary decision variables $b_n^\psi \in \mathbb{B}$ to capture whether n is a decision node associated with primitive STL function $\psi \in \Psi$. Conversely, binary decision variables $w_n^c \in \mathbb{B}$ indicate that node n is a classification node for class $c \in \mathcal{C}$.

The constraints governing the node functionality are:

$$\sum_{\psi \in \Psi} b_n^\psi + \sum_{c \in \mathcal{C}} w_n^c = 1, \quad \forall n \in \mathcal{N}, \tag{6a}$$

$$\sum_{c \in \mathcal{C}} w_n^c = 1, \quad \forall n \in \mathcal{L}. \tag{6b}$$

For internal nodes $n \in \mathcal{N}$, (6a) guarantees that n either performs a decision using a primitive ψ or classifies the data into a class c. For leaves $n \in \mathcal{L}$, (6b) enforces classification, i.e., assignment of a class c.

When imposing a decision using a primitive, time intervals and thresholds are required to complete the STL formula. The time parameters variable $\xi_\theta^n \in \mathbb{B}$, $\forall \theta \in \Theta$, encompasses all possible valuations of the time parameters within horizon H. The constraint capturing the primitive variable is

$$\sum_{\psi\in\Psi} b_n^\psi = \sum_{\theta\in\Theta} \xi_\theta^n, \quad \forall n \in \mathcal{N}. \tag{7}$$

If a primitive is selected, i.e., $\sum_{\psi\in\Psi} b_n^\psi = 1$, then (7) ensures that exactly one set of time parameters θ is selected for the primitive ψ. Otherwise, no θ is selected.

For classification nodes responsible, data flows to the sink node $\mathbf{t}$ if the predicted class is correct. The following constraint enforces this behavior. The inequality constraint is enforced since multiple classification nodes can predict the same class and the data can only flow into one of the nodes.

$$z_{\mathbf{t},n}^i \le w_n^c, \quad \forall i : \ell^i = c \in \mathcal{C},\, n \in \mathcal{N} \cup \mathcal{L}. \tag{8}$$

STL satisfaction constraints: In this section, we connect the decision variables for decision nodes with the satisfaction of primitive formulas. Additionally, the constraints capture the robustness of signals with respect to the STL formula obtained from a PSTL primitive and computed spatial and time parameters.

1) Robustness calculation: To capture the robustness of signals with respect to primitives, we use the following result.

Proposition 1. *Let s be a signal and $\psi \in \Psi$ be a PSTL primitive. For any valuations π and θ of the spatial and time parameters of ψ, we have $\rho(s, \psi(\pi, \theta)) = \rho(s, \psi(0, \theta)) - \pi$*

Proof. The proof follows trivially from the structure of primitive formulae in Φ, and is omitted for brevity. □

Using Prop. 1, we can precompute the robustness values for all signals s^i, primitives ψ and time parameters θ, which we denote as $\tau_\theta^{i,\psi}$. For example, the robustness of signal s^1 with respect to $\psi_1 = \Diamond_{[1..5]} h(s) \ge 0$ is $\tau_{[1..5]}^{1,\psi_1} = \max_{k\in[1..5]} h(s^1(k))$.

The robustness $\rho_\psi^{i,n} \in \mathbb{R}$ of data s^i with respect to the formula ψ with an arbitrary $\pi_n \in \mathbb{R}$ is captured by

$$\sum_{\theta\in\Theta} \xi_\theta^n \tau_\theta^{i,\psi} + \pi_n = \rho_\psi^{i,n}, \quad \forall i \in \mathcal{I},\, \psi \in \Psi,\, n \in \mathcal{N}. \tag{9}$$

We enforce constraint (9) for all internal nodes $n \in \mathcal{N}$ to compute the robustness of every signal with respect to the predicted formula at node n.

2) Soundness constraints: At a decision node $n \in \mathcal{N}$, a signal satisfying (violating) the primitive formula is directed to the left child node $l(n)$ (right child node $r(n)$).

To represent the sign of robustness, we introduce the binary variables $y_\psi^{i,n} \in \mathbb{B}$. The variable $y_\psi^{i,n}$ takes the value of one if $\rho_\phi^i \ge 0$, and zero if $\rho_\phi^i \le 0$. We employ the *big-M* method to encode this relationship such that the robustness value is not over-constrained when $y_\phi^i = 0$ as follows

$$\begin{aligned} \rho_\psi^{i,n} &\le M y_\psi^{i,n}, \quad \forall i \in \mathcal{I}, \psi \in \Psi, n \in \mathcal{N}, \\ -\rho_\psi^{i,n} &\le M(1 - y_\psi^{i,n}), \quad \forall i \in \mathcal{I}, \psi \in \Psi, n \in \mathcal{N}. \end{aligned} \tag{10}$$

where $M = \rho_\top \in \mathbb{R}_{>0}$ is the value of the maximum robustness.

Next, we need to connect the satisfaction of primitives at a decision node n with the flow to the left or right child. Hence, for indicating data flowing to the left child node, we introduce variable $z_{l(n)}^i \in \mathbb{B}$ which takes the value of one if two conditions are satisfied simultaneously (1) n is a decision node with primitive ψ, i.e., $b_n^\psi = 1$, and (2) the robustness is positive with respect to the formula $y_\psi^{i,n} = 1$. The conditions are enforced by

$$z_{l(n)}^i \le \sum_{\psi\in\Psi} \kappa_\psi^{i,n}, \quad \forall i \in \mathcal{I}, \forall n \in \mathcal{N}, \tag{11}$$

where $\kappa_\psi^{i,n} = y_\psi^{i,n} \cdot b_n^\psi$ is an auxiliary variable that captures the condition $z_{l(n)}^i = 1$ if and only if $y_\psi^{i,n} = 1$ and $b_n^\psi = 1$. Note that since both variables are binary, the product is equivalent to $\kappa_\psi^{i,n} = \min\{y_\psi^{i,n}, b_n^\psi\}$ and is encoded as mixed integer linear constraints. Data flowing to the right $z_{r(n)}^i \in \mathbb{B}$ follows a similar process but with opposite conditions as follows:

$$\begin{aligned} z_{r(n)}^i &\le 1 - \sum_{\psi\in\Psi} \kappa_\psi^{i,n}, \quad \forall i \in \mathcal{I}, n \in \mathcal{N}. \\ z_{r(n)}^i &\le \sum_{\psi\in\Psi} b_n^\psi. \end{aligned} \tag{12}$$

MILP formulation: The $CCR(\Phi)$ of the data set, which is our main objective in the optimization problem, is given by the flow into sink $\mathbf{t}$. Thus, we obtain the objective by summing up $z_{\mathbf{t},n}^i$ as follows:

$$CCR(\phi) = \frac{\sum_{i\in\mathcal{I}} \sum_{n\in\mathcal{N}\cup\mathcal{L}} z_{\mathbf{t},n}^i}{|\mathcal{I}|}. \tag{13}$$

Moreover, the optimal flow tree structure enables specifying a penalty for the number of decision nodes, thereby reducing the complexity of the resulting formula. However, this approach may involve a trade-off in performance, as more decisions can lead to better classification, i.e., larger CCR. The trade-off is captured by a variable $\lambda \in [0, 1]$ that serves as a regularization parameter. This parameter allows users to adjust the emphasis between complexity reduction and maintaining high performance. Consequently, the objective function in (13) is modified as follows:

$$\mathcal{J} = (1 - \lambda) \sum_{i\in\mathcal{I}} \sum_{n\mathcal{N}\cup\mathcal{L}} z_{\mathbf{t},n}^i + \lambda \sum_{\psi\in\Psi} b_n^\psi. \tag{14}$$

The first in $\mathcal{J}$ evaluates the number of correctly classified data, while the second term quantifies the number of decision nodes with λ serving as the adjustment variable.

Finally, problem in (4) becomes the MILP problem:

$$\begin{aligned} &\max\ \mathcal{J} \\ &\text{s.t. } (5) \quad \text{(Flow constraints)}, \\ &\quad (6) - (8) \quad \text{(Node function constraints)}, \\ &\quad (9) - (12) \quad \text{(STL satisfaction constraints)}. \end{aligned} \tag{15}$$

Problem simplification: In this section, we present a simplified version of the problem in (15) when considering first-level STL primitive formulae, i.e., formulae without nested temporal operators [16]. Under these conditions, the primitives $\rho(s, \Diamond_\theta h(s) \ge \pi) = -\rho(s, \Box_\theta h(s) \le \pi)$ and $\rho(s, \Diamond_\theta h(s) \le \pi) = -\rho(s, \Box_\theta h(s) \ge \pi)$ are equivalent.

Proof. From the definition, we have $\rho(\Diamond_\theta h(s) \ge \pi) = \max_{t\in\theta} \{h(s(t)) - \pi\} = -(-\max_{t\in\theta} \{h(s(t)) - \pi\}) = -(\min_{t\in\theta} \{\pi - h(s(t))\}) = -\rho(\Box_\theta h(s) \le \pi)$. The proof for the second equivalence follows similarly. □

Inferring a primitive $\square_\theta h(s) \le \pi$ from data is the same as learning $\lozenge_\theta h(s) \ge \pi$. We remove the primitives containing the always operator and, thus, reduce Ψ by half. The simplification can be employed with any set Φ closed under negation. For example, second level primitives of the form $\square_{\theta_1}\lozenge_{\theta_2}$ and $\lozenge_{\theta_1}\square_{\theta_2}$.

STL formula summary: Once the optimization in (15) is completed, we can trace all STL formulae starting from the first layer for each data class. Beginning with the first layer, the traversal to a left child from any node signifies the satisfaction of the STL formula associated with that node, whereas progressing to a right child indicates the negation of the STL formula ($\neg$). As we proceed through successive layers, the STL formulae encountered along the path are unified using the logical conjunction ($\wedge$). Upon reaching a class prediction node, the aggregated conjunction formula essentially defines the class for that node's prediction. Lastly, in cases where a class is represented in multiple nodes across the tree, the distinct STL formulae corresponding to each node are combined using the logical disjunction ($\vee$).

V. Case Studies

We demonstrate the effectiveness of our method in three case studies. First, the two-class naval surveillance dataset is first compared to other binary STL classification methods. Second, it demonstrates applying the proposed approach to multi-class STL inference using a four-class trace dataset. Third, it is a designed dataset that uses second-level STL primitives for classification. The adjustment variable λ is set to 0 to maximize CCR.

Naval Surveillance: We use the naval surveillance dataset [11]. The different classes represent two behaviors of the vessel trajectories, see Fig. 2. All trajectories start from the open sea on the right side. The blue trajectories are normal behaviors that directly head toward the harbor. The green and red trajectories are abnormal. The red trajectories approach the island and return to the open sea, while the green trajectories veer to the island before heading to the harbor. All trajectories contain 61 time steps of x and y positions. We use the optimal flow tree with a depth of 2 for

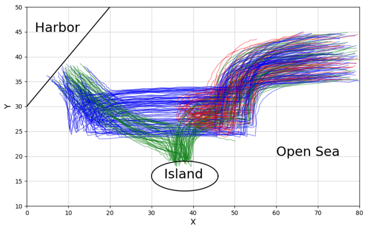
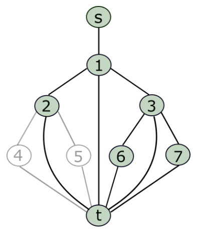
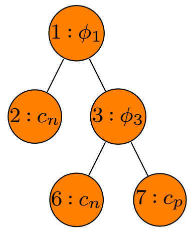


Fig. 2: Naval Surveillance Dataset. Blue trajectories: normal behavior, Green and Red trajectories: abnormal behavior

this dataset and compare other methods from [15]. All results are computed with an average of 10 runs. The classification tree and the resulting decision tree are shown in Fig. 3a and Fig. 3b, respectively. The colored nodes indicate nodes utilized for classification. A typical example of the learned formula from our method for the normal and abnormal trajectories is: node 1: $\phi_1 = \lozenge_{[13,59]} y \le 23.168$; node 2: abnormal; node 3: $\phi_3 = \lozenge_{[50,59]} x \ge 25.483$; node 6: abnormal; node 7: normal; STL formula for normal behavior $c_p = \neg\phi_1 \wedge \neg\phi_3$;

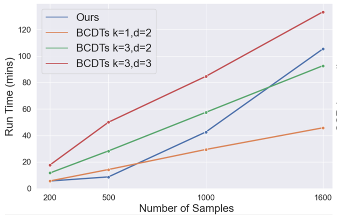


(a)

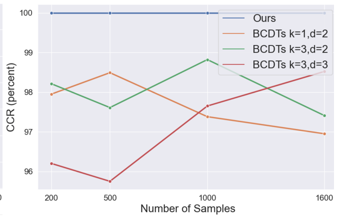


(b)

Fig. 3: 3a depicts the classification tree of depth 2 wherein only green-colored nodes are used for classification while grey nodes and edges are redundant. Fig. 3b shows the resulting STL BDT wherein c_p denotes "normal" and c_n indicates "abnormal" behavior.



(a)

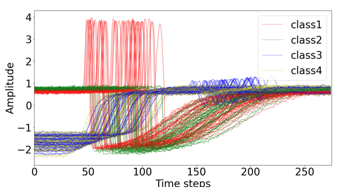


(b)

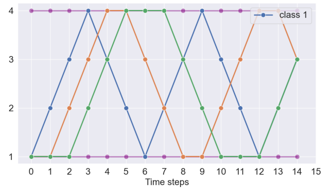
Fig. 4: Performance comparison (a) Run time (b) Correct classification rate.

STL formula for abnormal behavior $c_n = \phi_1 \vee \phi_3$. The interpretation of this formula is clear: the y coordinates of normal vessels do not reach the island between 13 and 59 seconds, and the x coordinates of normal vessels eventually enter the harbor between 50 and 59 seconds, while abnormal trajectories are the opposite. We compare the results of [15] using boosted concise decision trees (BCDTs) with respect to the average runtime and the classification accuracy in Fig. 4 by varying the number of samples. In the BCDTs approach, the variable k is the different number of decision trees used for classification, and d is the depth of the tree. The results demonstrate that our approach consistently outperforms the BCDTs approach in obtaining optimal solutions.

Trace Dataset: The trace dataset contains four classes of one-dimensional transient behavior from a simulated nuclear industry process [25]. The length of the data is 275, a total of 200 samples in the dataset, evenly distributed by classes as shown in Fig. 5a. We used a tree with a depth of 3 for this dataset, and the objective achieves 100 percent accuracy and the resulting formulae for the four classes are computed as follows: node 1: $\phi_1 = \lozenge_{[173,181]} s \le 0.74$; node 2: $\phi_2 = \lozenge_{[12,16]} s \ge 0$; node 3: $\phi_3 = \lozenge_{[115,167]} s \le 0.59$; node 4: $\phi_4 = \square_{[21,27]} s < 0.72$; node 5: $\phi_5 = \lozenge_{[157,216]} s \le 0.35$; node 6: $\phi_6 = \lozenge_{[230,234]} s \ge 0.88$; node 7: $\phi_7 = \square_{[210,210]} s < 0.69$; node 8: class 2; node 9: class 1; node 10 &13 &15: class 3; node 11 &12 &14: class 4. With class 1: $\phi_1 \wedge \phi_2 \wedge \phi_4$; class 2: $\phi_1 \wedge \phi_2 \wedge \neg\phi_4$; class3: $(\phi_1 \wedge \neg\phi_2 \wedge \phi_5) \vee (\neg\phi_1 \wedge \phi_3 \wedge$



(a)



(b)

Fig. 5: (a) Trace Dataset (b) Trajectories for high-level STL.

$\neg\phi_6) \vee (\neg\phi_1 \wedge \neg\phi_3 \wedge \phi_7)$; class4: $(\phi_1 \wedge \neg\phi_2 \wedge \neg\phi_5) \vee (\neg\phi_1 \wedge \phi_3 \wedge \phi_6) \vee (\neg\phi_1 \wedge \neg\phi_3 \wedge \neg\phi_7)$ The results demonstrate the capability for multi-class classification.

Classification using Second-level STL Primitives: Our framework accommodates complex STL formulae. Fig. 5b is an illustrative example of sample trajectories designed to highlight the expressivity of second-level STL formula. Current neural network-based approaches are limited to first-level formulae, and adapting them to other primitive sets is non-trivial [17]. Class 1 of the blue trajectory is a triangle wave, and we complement the data with variations of plateaued triangular waves, such as the yellow and green trajectories with plateaus of 2 and 3, respectively. To ensure that the inferred formula is not trivially defined by one tree level, we have trajectories of constant values at the lower and upper bounds drawn in purple. Next, we sample different trajectories from each type of wave by shifting the initiation point. We generate 300 samples with 15 time steps, 100 of which are class 1, and we denote the rest of the data as class 2. Lastly, we add a small random noise between -0.01 to 0.01 to all values. We use a tree with a depth of 2 for classification. The optimization returns 100 percent accuracy, and the results are the following: node 1: $\phi_1 = \Box_{[2,11]}\Diamond_{[0,4]}s \leq 1.0086$; node 2: class 2; node 3: $\phi_3 = \Box_{[0,10]}\Diamond_{[0,2]}s \leq 3.017$; node 6: class 1; node 7: class 2. ϕ_1 means within the time interval $[2, 11]$, there is always a time within any subinterval of length 4 where the signal value is less than or equal to 1.0086. Therefore, only the lower constant trajectory satisfies ϕ_1. Next, ϕ_3 means that within the time interval $[0, 10]$, there is always an event within any subinterval of length 2 where the signal value is less than or equal to 3.017. From the remaining data that does not satisfy ϕ_1, only the class 1 triangular wave satisfies ϕ_3. As a result, the STL formula for class 1 is $\neg\phi_1 \wedge \phi_3$.

VI. Conclusions

We propose a decision tree-based STL inference algorithm that works for two-class and multi-class classification problems. Our algorithm formulates the problem as a MILP by leveraging a max-flow approach over a synthesized tree using STL primitives, which allows for the computation of a globally optimal solution, i.e., a high Correct Classification Rate. Additionally, we exploit the symmetry of STL primitives to reduce the required constraints for classification, resulting in improved performance and reduced complexity. We evaluate the performance through three case studies to demonstrate the ability to interpret complex STL formulas and the precision in predictive accuracy. We show the capacity of our approach to handling two-class and multi-class classification problems with first-level and second-level STL primitives.

References

[1] A. A. Soofi and A. Awan, “Classification techniques in machine learning: applications and issues,” *J. Basic Appl. Sci*, vol. 13, no. 1, pp. 459–465, 2017.

[2] M. Krishnan, “Against interpretability: a critical examination of the interpretability problem in machine learning,” *Philosophy & Technology*, vol. 33, no. 3, pp. 487–502, 2020.

[3] L. H. Gilpin, D. Bau, B. Z. Yuan, A. Bajwa, M. Specter, and L. Kagal, “Explaining explanations: An overview of interpretability of machine learning,” in *2018 IEEE 5th International Conference on data science and advanced analytics (DSAA)*, pp. 80–89, IEEE, 2018.

[4] O. Maler and D. Nickovic, “Monitoring temporal properties of continuous signals,” in *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*, pp. 152–166, Springer, 2004.

[5] A. Donzé, “On signal temporal logic,” in *Runtime Verification: 4th International Conference, RV 2013, Rennes, France, September 24-27, 2013. Proceedings 4*, pp. 382–383, Springer, 2013.

[6] S. Mohammadinejad, J. V. Deshmukh, A. G. Puranic, M. Vazquez-Chanlatte, and A. Donzé, “Interpretable classification of time-series data using efficient enumerative techniques,” in *Proceedings of the 23rd International Conference on Hybrid Systems: Computation and Control*, pp. 1–10, 2020.

[7] E. Asarin, A. Donzé, O. Maler, and D. Nickovic, “Parametric identification of temporal properties,” in *Runtime Verification: Second International Conference, RV 2011, San Francisco, CA, USA, September 27-30, 2011, Revised Selected Papers 2*, pp. 147–160, Springer, 2012.

[8] X. Jin, A. Donzé, J. V. Deshmukh, and S. A. Seshia, “Mining requirements from closed-loop control models,” in *Proceedings of the 16th international conference on Hybrid systems: computation and control*, pp. 43–52, 2013.

[9] B. Hoxha, A. Dokhanchi, and G. Fainekos, “Mining parametric temporal logic properties in model-based design for cyber-physical systems,” *International Journal on Software Tools for Technology Transfer*, vol. 20, pp. 79–93, 2018.

[10] A. Bakhirkin, T. Ferrère, and O. Maler, “Efficient parametric identification for stl,” in *Proceedings of the 21st International Conference on Hybrid Systems: Computation and Control (part of CPS Week)*, pp. 177–186, 2018.

[11] Z. Kong, A. Jones, and C. Belta, “Temporal logics for learning and detection of anomalous behavior,” *IEEE Transactions on Automatic Control*, vol. 62, no. 3, pp. 1210–1222, 2016.

[12] A. Jones, Z. Kong, and C. Belta, “Anomaly detection in cyber-physical systems: A formal methods approach,” in *53rd IEEE Conference on Decision and Control*, pp. 848–853, IEEE, 2014.

[13] E. Bartocci, L. Bortolussi, and G. Sanguinetti, “Data-driven statistical learning of temporal logic properties,” in *International conference on formal modeling and analysis of timed systems*, pp. 23–37, Springer, 2014.

[14] S. Bufo, E. Bartocci, G. Sanguinetti, M. Borelli, U. Lucangelo, and L. Bortolussi, “Temporal logic based monitoring of assisted ventilation in intensive care patients,” in *Leveraging Applications of Formal Methods, Verification and Validation. Specialized Techniques and Applications: 6th International Symposium, ISoLA 2014, Imperial, Corfu, Greece, October 8-11, 2014, Proceedings, Part II 6*, pp. 391–403, Springer, 2014.

[15] E. Aasi, C. I. Vasile, M. Bahreinian, and C. Belta, “Classification of time-series data using boosted decision trees,” in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1263–1268, IEEE, 2022.

[16] G. Bombara, C.-I. Vasile, F. Penedo, H. Yasuoka, and C. Belta, “A decision tree approach to data classification using signal temporal logic,” in *Proceedings of the 19th International Conference on Hybrid Systems: Computation and Control*, pp. 1–10, 2016.

[17] D. Li, M. Cai, C.-I. Vasile, and R. Tron, “Learning signal temporal logic through neural network for interpretable classification,” in *2023 American Control Conference (ACC)*, pp. 1907–1914, IEEE, 2023.

[18] A. Linard, I. Torre, I. Leite, and J. Tumova, “Inference of multi-class stl specifications for multi-label human-robot encounters,” in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1305–1311, IEEE, 2022.

[19] D. Li and R. Tron, “Multi-class temporal logic neural networks,” *arXiv preprint arXiv:2402.12397*, 2024.

[20] S. Aghaei, A. Gómez, and P. Vayanos, “Strong optimal classification trees,” *arXiv preprint arXiv:2103.15965*, 2021.

[21] G. E. Fainekos and G. J. Pappas, “Robustness of temporal logic specifications for continuous-time signals,” *Theoretical Computer Science*, vol. 410, no. 42, pp. 4262–4291, 2009.

[22] A. Donzé and O. Maler, “Robust satisfaction of temporal logic over real-valued signals,” in *International Conference on Formal Modeling and Analysis of Timed Systems*, pp. 92–106, Springer, 2010.

[23] A. Dokhanchi, B. Hoxha, and G. Fainekos, *5th International Conference on Runtime Verification, Toronto, ON, Canada.*, ch. On-Line Monitoring for Temporal Logic Robustness, pp. 231–246. Springer, 2014.

[24] B. D. Ripley, *Pattern recognition and neural networks*. Cambridge university press, 2007.

[25] H. A. Dau, A. Bagnall, K. Kamgar, C.-C. M. Yeh, Y. Zhu, S. Gharghabi, C. A. Ratanamahatana, and E. Keogh, “The ucr time series archive,” *IEEE/CAA Journal of Automatica Sinica*, vol. 6, no. 6, pp. 1293–1305, 2019.