Survey of Signal Temporal Logic Specification Mining: Techniques, Applications, and Future Directions

Jialiang Fan and Fanxin Kong

Department of Computer Science and Engineering, University of Notre Dame

ABSTRACT

Formal methods play a crucial role in system design and verification, providing an explicit framework for specifying system behaviors and functional requirements. Signal Temporal Logic (STL), a widely adopted formal specification method, captures dynamic system behavior from a temporal perspective and is extensively used in embedded systems, robotics, and cyber-physical systems. STL mining is a technique that automatically extracts STL specifications from large sets of trajectories, aiding in system understanding, real-time monitoring, and control. Over the past decade, significant advancements in STL mining have emerged, utilizing approaches such as optimization-based, search-based, template-based, and template-free methods. These techniques have been applied across various fields, including cyber-physical systems, robotics, aerospace, and autonomous driving. This paper provides a comprehensive survey of STL specification mining, reviewing recent developments, summarizing key applications, and discussing future research directions.

Keywords: Cyber-physical systems, signal temporal logic, formal specification mining, optimization algorithms, machine learning

1. INTRODUCTION

Formal methods provide a rigorous framework for specifying and verifying system behaviors,[1] ensuring reliability and correctness in various applications such as cyber-physical systems (CPS), robotics, and autonomous systems. Among these methods, Signal Temporal Logic (STL) has gained significant attention due to its ability to express temporal and spatial properties of continuous signals. STL specifications serve as interpretable constraints that capture system requirements, making them valuable for system analysis, real-time monitoring, and controller synthesis. Sometimes, it is challenging to derive the STL specification of a system due to the complexity of its dynamics, the lack of prior knowledge, or the presence of noise and uncertainties in the observed trajectories. To address this, the STL mining has emerged as a promising research area that seeks to automatically extract STL formulas from large sets of system trajectories. By leveraging data-driven approaches, STL mining enables efficient system characterization, anomaly detection, and control policy evaluation without requiring predefined specifications.

In this paper, we provide a comprehensive review of STL specification mining by

- Summarizing advances in the past five years in STL mining methodologies, categorizing them based on their underlying principles and techniques.

Further author information: (Send correspondence to Fanxin Kong)
Fanxin Kong: E-mail: fkong@nd.edu
Jialiang Fan: E-mail: jfan5@nd.edu

- Reviewing key applications of STL mining in CPS, robotics, autonomous systems, and safety-critical domains.
- Identifying open challenges, future research directions, and potential applications of STL mining, including enhancing robustness, defending against adversarial attacks, and integrating with deep learning and reinforcement learning frameworks for improved specification learning and control.

The rest of this paper is organized as follows. Section 2 reviews recent advances in STL mining methodologies, categorizing them from several perspectives. Section 3 explores key applications of STL mining across cyber-physical systems (CPS), robotics, autonomous systems, and safety-critical domains. Section 4 identifies open challenges and future research directions, including robustness, adversarial defenses, and integration with deep learning and reinforcement learning frameworks. Finally, Section VI concludes the paper with a summary of findings and potential research opportunities.

2. RECENT ADVANCES IN STL MINING METHODOLOGIES

2.1 Overview of STL Specification Mining

STL mining techniques can be categorized from multiple perspectives, considering the level of prior knowledge and the underlying methodologies. Specifically, they can be classified as template-based or template-free, passive or active, supervised or unsupervised, and offline or online.

Template-based or template-free: Template-based STL mining relies on predefined formula structures with tunable parameters, reducing the search space and improving interpretability. In contrast, template-free approaches explore the space of possible STL formulas without predefined constraints, offering greater flexibility but often at the cost of increased computational complexity and reduced interpretability.

Passive or active: Passive STL mining extracts specifications from pre-collected trajectory data without influencing the system's behavior. In contrast, active STL mining interacts with the system by generating specific inputs or modifying conditions to explore the state space more effectively, often improving the quality and robustness of the mined specifications.

Supervised or unsupervised: Supervised STL mining requires labeled trajectory data, where each trajectory is annotated with whether it satisfies or violates certain system properties. This enables learning STL formulas that best separate positive and negative samples. In contrast, unsupervised STL mining discovers specifications without explicit labels, often relying on clustering, statistical analysis, or optimization techniques to identify patterns and constraints within the data.

Offline or online: Offline STL mining processes a fixed set of trajectory data to extract specifications, making it suitable for applications where complete datasets are available beforehand. In contrast, online STL mining incrementally updates specifications as new data becomes available, enabling real-time monitoring, adaptive control, and dynamic system verification.

Different applications and scenarios may require different STL mining techniques. In the following sections, we will explore recent research from these perspectives.

2.2 Recent advances of STL Mining

In this section, we provide an overview of advances in STL mining over the past five years. Our review covers 14 papers published since 2020, analyzing them from five key perspectives: template-free vs. template-based, passive vs. active, supervised vs. unsupervised, offline vs. online, and whether they are open-source. Additionally, we summarize their applications and methodologies. The results are presented in Table 1, and we will discuss these papers in detail in the following sections.

Table 1: Recent Advances of STL mining

Ref.	Year	Template-free	Passive/Active	Supervised/Unsupervised	Offline/Online	Open-source	Applications	Method
Ref.[2]	2020	✗	Passive	Sup.	Offline	N/A	CPS	Decision trees
Ref.[3]	2021	✓	Passive	Unsup.	Offline	Available*	Traffic monitoring	Context-free grammar genetic programming
Ref.[4]	2024	✗	Passive	Sup.	Offline	N/A	Design with verification	Decision-tree algorithm with data clustering
Ref.[5]	2024	✓	Passive	Sup.	Offline	N/A	CPS	Bayesian optimization and information retrieval
Ref.[6]	2024	✗	Passive	Sup.	Offline	Available†	General-purpose	Search & Optimization
Ref.[7]	2024	✗	Active	Sup.	Online	Available‡	Safe reinforcement learning	Bayesian optimization
Ref.[8]	2023	✓	Passive	Unsup.	Offline	N/A	Hyperproperties system	Syntax-guided synthesis

Table 1: Recent Advances of STL mining

Ref.	Year	Template-free	Passive/Active	Supervised/Unsupervised	Offline/Online	Open-source	Applications	Method
Ref.[9]	2021	✓	Passive	Sup.	Offline	Available[§]	Sequential logic reasoning and system analysis	Decision trees
Ref.[10]	2024	✓	Passive	Sup.	Online	N/A	Time-incremental learning	Neural networks and decision trees
Ref.[11]	2023	✓	Passive	Sup.	Offline	Available[¶]	Natural language to STL	Transformer-based generative models
Ref.[12]	2021	✓	Passive	Sup.	Online	N/A	Monitoring & Classification	Decision trees
Ref.[13]	2023	✓	Passive	Sup.	Offline	Available[‖]	Time-Series classification	Symbolic neural networks
Ref.[14]	2020	✗	Active	Unsup.	Online	Available[**]	Robotic control and planning	Decision trees
Ref.[15]	2024	✓	Passive	Sup.	Offline	N/A	Robotic control & Trajectory classification	Genetic algorithms

3. APPLICATIONS OF STL MINING

The applications of STL mining are diverse. In this section, we discuss the applications of literatures in Table 1 from two perspectives: general functions and specific use cases. The general functions include tasks such as monitoring, classification, and verification. Meanwhile, specific applications focus on real-world scenarios where STL mining is employed, such as robotic control, cyber-physical systems (CPS), and naval surveillance.

3.1 General Functions

STL mining serves various purposes in analyzing and ensuring the correctness of system behaviors. Its general functions can be categorized as follows:

- **Monitoring:** Evaluates whether the output signals of system components adhere to predefined temporal logic requirements.[2–4, 8]
- **Classification:** Frames STL mining as a classification problem, where input signals are categorized as good (leading to valid output signals) or bad (otherwise).[2, 5, 8–10, 12, 13]
- **Verification:** Formally analyzes execution traces to ensure that a system's behavior conforms to predefined STL specifications.[11, 15]
- **Scalability:** Efficiently processes long execution traces, making STL mining applicable to large-scale industrial systems.[4]
- **Human-Readable Rule Extraction:** Employs grammar-based evolutionary computation to generate interpretable STL specifications, facilitating human understanding and validation.[3, 4]
- **Constraint Identification:** Utilizes parametric Signal Temporal Logic (pSTL) to infer unknown safety constraints dynamically, rather than relying on predefined rules. This enhances safety in applications such as robotics, autonomous vehicles, and drones by learning and enforcing STL-based constraints in real-time.[7]

3.2 Specific Applications

The studies listed in Table 1 explore a diverse range of real-world applications of STL mining, including:

---

*https://github.com/pigozzif/STLRulesEvolutionaryInferenceNoClass

†https://gricad-gitlab.univ-grenoble-alpes.fr/verimag/tempo/multidimensional_search

‡https://github.com/SAILRIT/Concurrent-Learning-of-Control-Policy-and-Unknown-Constraints-in-Reinforcement-Learning

§https://github.com/maryambagheri1989/POR/

¶https://github.com/yongchao98/NL2TL

‖https://github.com/danyangl6/NN-TLI

**https://github.com/allinard/active-learn-stl

- **Transportation Systems:** Identifies throttle conditions that prevent excessive engine and vehicle speed[2] and determines operational limits for throttle angles to maintain an efficient air-to-fuel ratio.[2] Extracts STL-based traffic rules to detect violations and unsafe driving patterns,[3] while also providing interpretable traffic behavior models to guide autonomous driving policies.[3] Assists city planners in analyzing real-world driving patterns to develop safety strategies[3] and leverages STL-based predictions to classify driving behaviors and detect deviations from expected temporal patterns.[10]

- **Cyber-Physical Systems (CPS):** Automatically extracts formal STL specifications for CPS and mixed discrete-continuous systems.[4] Mines STL rules to detect system behaviors based on trajectory data[5] and provides a structured framework for analyzing time-series behaviors in CPS, industrial automation, and safety-critical applications.[13]

- **Medical Applications:** Utilizes parametric STL to detect heart diseases such as arrhythmia by analyzing the shape of ECG pulses.[7]

- **Robotics:** Infers STL formulas to describe the behavior of interval trajectories in a Pusher-robot system.[9] Identifies patterns in vehicle system traces to detect fault conditions using STL classification.[12] Extracts social requirements for autonomous systems, enabling robots to learn STL-based social behavior rules from human interactions.[14] Uses STL-based trajectory predictions to constrain motion planning tasks, preventing collisions and ensuring safe navigation.[15]

4. OPEN CHALLENGES

In this section, we summarize and outline key research challenges identified in the papers listed in Table 1. Despite significant advancements made in STL mining over the past five years, we summary six challenges from these papers that remain open for future exploration.

- **Enhancing Learning Paradigms:** Expanding supervised, unsupervised, and active learning techniques to improve STL inference, especially in settings with limited labeled data or dynamically changing environments.

- **Improving Efficiency and Scalability:** Reducing computational overhead by optimizing formula search, fitness functions, and parameter learning, while ensuring scalability for large datasets and real-time applications.

- **Handling Uncertainty and Robustness:** Developing uncertainty-aware STL mining to improve reliability in noisy and stochastic environments, particularly for applications like reinforcement learning and cyber-physical systems.

- **Expanding Expressiveness of STL:** Enhancing STL grammar, operators, and feature selection to capture more complex system behaviors, making mined specifications more interpretable and generalizable.

- **Ensuring Safety and Interpretability:** Integrating formal verification, correction queries, and explainable AI to ensure that mined STL formulas are both human-interpretable and aligned with safety-critical constraints.

- **Bridging Human Interaction and Automation:** Leveraging natural language processing (NLP), crowdsourcing, and interactive learning to enable more intuitive and adaptive STL mining approaches for real-world deployment.

5. CONCLUSION

In this paper, we review advancements in STL mining over the past five years, summarizing key works from seven perspectives. We then explore the motivations behind STL mining and its real-world applications. Finally, we highlight several open challenges in STL mining. This survey provides readers with a clear and practical outlook on STL mining methodologies.

ACKNOWLEDGMENT

This work was supported in part by NSF CNS-2333980. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the National Science Foundation (NSF).

REFERENCES

[1] Rashid, A. and Hasan, O., "Formal analysis of the continuous dynamics of cyber–physical systems using theorem proving," *Journal of Systems Architecture* **112**, 101850 (2021).

[2] Mohammadinejad, S., Deshmukh, J. V., and Puranic, A. G., "Mining environment assumptions for cyber-physical system models," in [*2020 ACM/IEEE 11th International Conference on Cyber-Physical Systems (ICCPS)*], 87–97, IEEE (2020).

[3] Pigozzi, F., Medvet, E., and Nenzi, L., "Mining road traffic rules with signal temporal logic and grammar-based genetic programming," *Applied Sciences* **11**(22), 10573 (2021).

[4] Nicoletti, D., Germiniani, S., and Pravadelli, G., "Mining signal temporal logic specifications for hybrid systems," in [*2024 Forum on Specification & Design Languages (FDL)*], 1–8, IEEE (2024).

[5] Saveri, G. and Bortolussi, L., "Retrieval-augmented mining of temporal logic specifications from data," in [*Joint European Conference on Machine Learning and Knowledge Discovery in Databases*], 315–331, Springer (2024).

[6] Mambakam, A., Requeno Jarabo, J. I., Bakhirkin, A., Basset, N., and Dang, T., "Mining of extended signal temporal logic specifications with paretolib 2.0," *Formal Methods in System Design* **62**(1), 260–284 (2024).

[7] Yifru, L. and Baheri, A., "Concurrent learning of control policy and unknown safety specifications in reinforcement learning," *IEEE Open Journal of Control Systems* (2024).

[8] Bartocci, E., Mateis, C., Nesterini, E., and Ničković, D., "Mining hyperproperties using temporal logics," *ACM Transactions on Embedded Computing Systems* **22**(5s), 1–26 (2023).

[9] Baharisangari, N., Gaglione, J.-R., Neider, D., Topcu, U., and Xu, Z., "Uncertainty-aware signal temporal logic inference," in [*International Workshop on Numerical Software Verification*], 61–85, Springer (2021).

[10] Aasi, E., Cai, M., Vasile, C. I., and Belta, C., "Time-incremental learning of temporal logic classifiers using decision trees," in [*Learning for Dynamics and Control Conference*], 547–559, PMLR (2023).

[11] Chen, Y., Gandhi, R., Zhang, Y., and Fan, C., "Nl2tl: Transforming natural languages to temporal logics using large language models," *arXiv preprint arXiv:2305.07766* (2023).

[12] Bombara, G. and Belta, C., "Offline and online learning of signal temporal logic formulae using decision trees," *ACM Transactions on Cyber-Physical Systems* **5**(3), 1–23 (2021).

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

[14] Linard, A. and Tumova, J., "Active learning of signal temporal logic specifications," in [*2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)*], 779–785, IEEE (2020).

[15] Soroka, E., Sinha, R., and Lall, S., "Learning temporal logic predicates from data with statistical guarantees," *arXiv preprint arXiv:2406.10449* (2024).