logics

MDPI

Article

Multi-Fidelity Temporal Reasoning: A Stratified Logic for Cross-Scale System Specifications

Ali Baheri [1,*] and Peng Wei [2]

[1] Rochester Institute of Technology, Rochester, NY 14623, USA
[2] Mechanical and Aerospace Engineering Department, George Washington University, Washington, DC 20052, USA; pwei@gwu.edu
* Correspondence: akbeme@rit.edu

Abstract: We present **S**tratified **M**etric **T**emporal **L**ogic (SMTL), a novel formalism for specifying and verifying the properties of complex cyber–physical systems that exhibit behaviors across multiple temporal and abstraction scales. SMTL extends existing temporal logics by incorporating a stratification operator, enabling the association of temporal properties with specific abstraction levels. This allows for the natural expression of multi-scale requirements while maintaining formal reasoning about inter-level relationships. We formalize the syntax and semantics of SMTL, proving that it strictly subsumes metric temporal logic (MTL) and offers enhanced expressiveness by capturing properties unattainable in existing logics. Numerical simulations comparing agents operating under MTL and SMTL specifications show that SMTL enhances agent coordination and safety, reducing collision rates without substantial computational overhead or compromising path efficiency. These findings highlight SMTL's potential as a valuable tool for designing and verifying complex multi-agent systems operating across diverse temporal and abstraction scales.

Keywords: stratified-metric temporal logic; multi-scale systems; formal verification; multi-agent coordination; temporal logic

check for updates

Academic Editor: Valentin Goranko

Received: 18 February 2025
Revised: 19 April 2025
Accepted: 20 May 2025
Published: 3 June 2025

Citation: Baheri, A.; Wei, P. Multi-Fidelity Temporal Reasoning: A Stratified Logic for Cross-Scale System Specifications. *Logics* **2025**, *3*, 5. https://doi.org/10.3390/logics3020005

1. Introduction

Cyber–physical systems (CPS) exhibit behaviors across multiple temporal and abstraction scales, from millisecond-level control loops to hour-long mission objectives. While temporal logics have proven invaluable for specifying and verifying system properties [1], existing formalisms such as metric temporal logic (MTL) and signal temporal logic (STL) struggle to efficiently capture and reason about such multi-scale behaviors. This limitation becomes particularly acute in systems such as autonomous vehicles, where safety properties must be simultaneously maintained at the level of individual actuators, trajectory planning, and mission execution.

Existing temporal logics enforce a uniform treatment of time, requiring specifications to operate at the finest required temporal granularity [2]. Consider an autonomous drone delivery system: at the lowest level, motor control requires millisecond-precision timing; at the intermediate level, trajectory following operates on a timescale of seconds; and at the highest level, delivery scheduling involves minutes or hours. Expressing all of these requirements in MTL or STL requires reasoning on a millisecond scale throughout, leading to unnecessary complexity and computational overhead. The key challenge lies in the lack of formal mechanisms for stratifying temporal properties across different abstraction levels while maintaining sound logical relationships between these levels. Existing approaches typically handle multi-scale properties through informal decomposition or

through separate specifications at different levels, losing the ability to reason formally about inter-level dependencies.

1.1. Paper Contributions

This paper introduces SMTL, an extension of traditional temporal logics incorporating a stratification operator to enable multi-scale temporal specification. The key contributions are as follows:

- We formalize the syntax and semantics of SMTL, proving that it strictly subsumes MTL and offers enhanced expressiveness by capturing properties unattainable in existing logics.
- We analyze the decidability and computational complexity of SMTL model checking, identifying decidable fragments with complexities comparable to MTL for bounded-time properties
- Through case studies and numerical simulations, we demonstrate SMTL's practical utility in specifying multi-scale requirements and improving verification efficiency, enhancing agent coordination and safety.

The practical utility of SMTL is demonstrated through case studies, where we show how SMTL specifications naturally capture requirements that span multiple time scales while allowing more efficient verification compared to traditional approaches.

1.2. Related Work

Various temporal logics have been developed to specify and reason about temporal properties in CPS and control systems. STL and MTL are prominent examples that extend LTL to handle real-valued signals and timing constraints [3]. Probabilistic STL (PrSTL) extends STL with probabilistic operators to reason about stochastic CPS [4]. Truncated linear temporal logic (LTL) is a variant of LTL interpreted over finite traces and has been used to specify complex rules for RL agents. Robustness TL (RobTL) is a logic for specifying distances between CPS behaviors over a finite time horizon [5]. The use of temporal logics such as computation tree logic (CTL) and LTL provided a foundation for expressing desired behaviors in multi-agent systems [6–11].

SMTL builds upon the foundations of MTL to address the complexities of agent coordination and collision avoidance. MTL extends LTL by incorporating quantitative temporal constraints, enabling the specification of properties over real-time systems. While MTL has been used for specifying timing constraints and verifying system behaviors, it lacks mechanisms to *prioritize* different types of constraints, which is important in multi-agent coordination where safety and goal achievement may conflict. SMTL is a response to this limitation that introduces a stratification of temporal constraints, allowing agents to prioritize safety and coordination over individual objectives. The stratification concept aligns with the *hierarchical* organization of specifications, where higher-priority constraints must be satisfied before considering lower-priority ones. This approach is particularly relevant in multi-agent systems, where agents operate concurrently and interactions can lead to emergent behaviors.

Previous efforts have explored the use of hierarchical and prioritized logics in multi-agent planning and verification. For instance, ref. [12] developed TLA as a framework for reasoning about concurrent systems, with a focus on simplifying temporal logic reasoning. Separately, ref. [13] developed real-time logics to enable quantitative reasoning about timing delays in real-time applications. Research in multi-agent coordination employs various formal methods, including temporal logics, to specify and verify interaction protocols and ensure safe coordination between agents. These methods focus on specifying conventions of social interaction and rules that govern agent behavior in multi-agent systems [14].

Furthermore, research on robot control synthesis has focused on using LTL to specify both safety requirements (describing how the robot should always behave) and liveness specifications (describing goals that must eventually be achieved) [15].

Stratification has been well established in database theory through Stratified Datalog, where it provides a way to handle negation and recursion by organizing rules into strata. This concept has been extended to temporal reasoning through formalisms like DatalogMTL, which incorporates metric temporal operators [16]. In recent years, the application of formal methods in multi-agent pathfinding has gained traction, with work using satisfiability modulo theories (SMTs) and other verification techniques to ensure safe navigation [17]. Sharon et al. developed conflict-based search (CBS) algorithms that decompose multi-agent pathfinding problems into individual agent plans while resolving conflicts iteratively [18]. These methods do not inherently incorporate stratified priorities in their formulations. The introduction of SMTL contributes to this landscape by providing a logical framework that *inherently* supports priority stratification. By structuring temporal constraints into layers, SMTL facilitates the design of agents that can dynamically adjust their actions based on the priority of constraints, leading to improved coordination and safety in complex environments. Moreover, the integration of SMTL into multi-agent systems aligns with the broader trend of incorporating formal verification into agent-based modeling. The works of Belta et al. on formal methods for control synthesis have emphasized the importance of temporal logic in specifying desired behaviors, although the stratification aspect has remained less explored [19]. An additional dimension relevant to stratified logic is the use of multi-fidelity modeling and verification techniques [20–24]. Multi-fidelity approaches involve employing models of varying levels of detail and accuracy to balance the trade-off between computational efficiency and the precision of system analysis.

The closest work to our approach is multi-type LTL (MLTLM), which focuses on heterogeneous data types (e.g., Boolean, integer, real signals) within a single-level temporal logic framework, enabling us to write and verify properties involving different kinds of signals at once [25]. By contrast, SMTL emphasizes hierarchical or multi-scale abstractions. While MLTLM is about handling various data domains in one logic, SMTL is about cleanly separating and coordinating multiple levels of temporal or system detail in a single formalism.

1.3. Paper Organization

The remainder of this paper is organized as follows. Section 2 provides the necessary background on temporal logics and introduces the mathematical framework for stratified systems. Section 3 presents the formal syntax and semantics of SMTL, including examples that demonstrate its expressiveness. Section 4 develops the theoretical results, proving SMTL's properties and complexity bounds. Section 5 presents experimental results comparing SMTL and MTL in multi-agent coordination scenarios. Finally, Section 6 concludes with a discussion of our findings and future research directions.

2. Preliminaries

Before introducing SMTL, we review the essential concepts of metric temporal logic and establish the mathematical framework for reasoning about stratified systems. Throughout this paper, we use the notation presented in Table 1.

Table 1. Symbols and complexity measures used in the formalism.

Symbol	Description
Σ_k	Signal space at abstraction level k
$\mathcal{L}(\varphi)$	Language of formula φ
$s \sqsubseteq_k s'$	State s refines state s' at level k
$\models_k$	Satisfaction relation at abstraction level k
$[a,b]$, (a,b), $[a,b)$, $(a,b]$	Intervals with $a, b \in \mathbb{Q}_{\geq 0} \cup \{\infty\}$
$\Diamond\varphi$	Shorthand for $\Diamond_{[0,\infty)}\varphi$
Complexity Measures	
$\lvert\varphi\rvert$	Size of formula φ
$\lVert\mathcal{M}\rVert$	Size of transition system $\mathcal{M}$
K	Maximum abstraction level
$c_{\max}$	Maximum constant in timing constraints

2.1. Metric Temporal Logic

MTL extends propositional temporal logic with timing constraints on temporal operators. Let AP be a set of atomic propositions and $\mathbb{I}$ be the set of non-empty intervals in $\mathbb{R}_{\geq 0}$ with endpoints in $\mathbb{Q}_{\geq 0} \cup \{\infty\}$.

Definition 1 (MTL Syntax)**.** *An MTL formula φ is built according to the grammar*

$$\varphi ::= p \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \mathcal{U}_I \varphi_2$$

where $p \in AP$ and $I \in \mathbb{I}$.

Additional temporal operators are defined as the following abbreviations:

$$\Diamond_I \varphi \triangleq \text{true}\mathcal{U}_I\varphi \qquad \text{(eventually)}$$
$$\Box_I \varphi \triangleq \neg\Diamond_I\neg\varphi \qquad \text{(always)}$$
$$\varphi_1 \mathcal{R}_I \varphi_2 \triangleq \neg(\neg\varphi_1 \mathcal{U}_I \neg\varphi_2) \qquad \text{(release)}$$

Definition 2 (Timed State Sequence)**.** *A timed state sequence is a pair $\rho = (\sigma, \tau)$ where*

- *$\sigma = \sigma_0\sigma_1\sigma_2\ldots$ is an infinite sequence of states $\sigma_i \in 2^{AP}$;*
- *$\tau = \tau_0\tau_1\tau_2\ldots$ is an infinite sequence of time stamps $\tau_i \in \mathbb{R}_{\geq 0}$, with $\tau_0 = 0$ and $\tau_i < \tau_{i+1}$ for all $i \geq 0$.*

Definition 3 (MTL Semantics)**.** *For a timed state sequence $\rho = (\sigma, \tau)$ and position $i \geq 0$, the satisfaction relation $\models$ is defined inductively:*

$$\begin{aligned}
(\rho, i) \models p &\iff p \in \sigma_i \\
(\rho, i) \models \neg\varphi &\iff (\rho, i) \not\models \varphi \\
(\rho, i) \models \varphi_1 \wedge \varphi_2 &\iff (\rho, i) \models \varphi_1 \text{ and } (\rho, i) \models \varphi_2 \\
(\rho, i) \models \varphi_1 \mathcal{U}_I \varphi_2 &\iff \exists j \geq i : (\rho, j) \models \varphi_2, \tau_j - \tau_i \in I, \\
&\qquad \text{and } \forall k, i \leq k < j : (\rho, k) \models \varphi_1
\end{aligned}$$

2.2. Stratified Systems and Abstraction Hierarchies

We now introduce the mathematical framework for reasoning about systems at multiple abstraction levels.

Definition 4 (Abstraction Function)**.** *An abstraction function $\alpha : S \to S'$ maps concrete states to abstract states, where S and S' are state spaces. An abstraction hierarchy is a sequence of state spaces and abstraction functions $\{(S_k, \alpha_k)\}_{k=1}^{K}$ where*

- *S_k is the state space at level k;*
- *$\alpha_k : S_{k-1} \to S_k$ maps states from level $k-1$ to level k.*

Definition 5 (Abstraction Properties)**.** *An abstraction hierarchy satisfies the following:*

1. *Monotonicity: For $i < j$, $\alpha_j \circ \alpha_i = \alpha_j$.*
2. *Preservation: For any property φ preserved by α_k, if $s \models \varphi$, then $\alpha_k(s) \models \varphi$.*
3. *Refinement: For states s_1, s_2, if $\alpha_k(s_1) = \alpha_k(s_2)$, then s_1 and s_2 are equivalent at level k.*

Definition 6 (Temporal Resolution)**.** *Each abstraction level k has an associated temporal resolution $\rho_k \in \mathbb{R}_{>0}$, representing the minimum granularity of time distinguishable at that level. The temporal resolution hierarchy satisfies the following:*

$$\forall k < l : \rho_k < \rho_l$$

3. Syntax and Semantics of SMTL

3.1. Syntax

Definition 7 (SMTL Syntax)**.** *Let AP be a set of atomic propositions and $\mathbb{I}$ be the set of non-empty intervals in $\mathbb{R}_{\geq 0}$ with endpoints in $\mathbb{Q}_{\geq 0} \cup \{\infty\}$. An SMTL formula φ is built according to the grammar*

$$\varphi ::= p \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \mathcal{U}_I \varphi_2 \mid L_k \varphi$$

where $p \in AP$ is an atomic proposition, $I \in \mathbb{I}$ is a time interval, $k \in \mathbb{N}$ is an abstraction level, and L_k is the stratification operator for level k.

As in MTL, we define additional temporal operators using the following abbreviations:

$$\Diamond_I \varphi \triangleq \text{true} \mathcal{U}_I \varphi$$
$$\Box_I \varphi \triangleq \neg \Diamond_I \neg \varphi$$
$$\varphi_1 \mathcal{R}_I \varphi_2 \triangleq \neg(\neg\varphi_1 \mathcal{U}_I \neg\varphi_2)$$

The stratification operator L_k can be nested, allowing for relationships between abstraction levels:

$$L_i(L_j \varphi) \text{ where } i, j \in \mathbb{N}$$

Definition 8 (Well-Formed SMTL Formula)**.** *An SMTL formula φ is well formed if for any nested stratification operators (L_i and L_j) appearing in φ, where L_i is within the scope of L_j, we have $i \leq j$.*

3.2. Semantics

The semantics of SMTL is defined over timed state sequences with multiple abstraction levels.

Definition 9 (Stratified Timed State Sequence)**.** *A stratified timed state sequence is a tuple $\rho = (\{\sigma_k\}_{k=1}^{K}, \tau)$ where*

- $\sigma_k = \sigma_{k,0}\sigma_{k,1}\sigma_{k,2}\ldots$ *is an infinite sequence of states at level k, with* $\sigma_{k,i} \in 2^{AP}$;
- $\tau = \tau_0\tau_1\tau_2\ldots$ *is an infinite sequence of time stamps with* $\tau_0 = 0$ *and* $\tau_i < \tau_{i+1}$;
- *For each k, consecutive states in* σ_k *are separated by at least* ρ_k *time units.*

Definition 10 (Abstraction Consistency)**.** *A stratified timed state sequence* ρ *is abstraction consistent if for all levels* $i < j$ *and all positions n,*

$$\alpha_j(\sigma_{i,n}) = \sigma_{j,n}$$

where α_j *is the abstraction function from level i to level j.*

Definition 11 (SMTL Semantics)**.** *For a stratified timed state sequence* $\rho = (\{\sigma_k\}_{k=1}^{K}, \tau)$*, position* $i \geq 0$*, and abstraction level m, the satisfaction relation* $\models_m$ *is defined inductively:*

$$\begin{aligned}
(\rho, i) \models_m p &\iff p \in \sigma_{m,i} \\
(\rho, i) \models_m \neg\varphi &\iff (\rho, i) \not\models_m \varphi \\
(\rho, i) \models_m \varphi_1 \wedge \varphi_2 &\iff (\rho, i) \models_m \varphi_1 \text{ and } (\rho, i) \models_m \varphi_2 \\
(\rho, i) \models_m \varphi_1 \mathcal{U}_I \varphi_2 &\iff \exists j \geq i : (\rho, j) \models_m \varphi_2, \tau_j - \tau_i \in I, \\
&\qquad \text{and } \forall k, i \leq k < j : (\rho, k) \models_m \varphi_1 \\
(\rho, i) \models_m L_k\varphi &\iff (\rho, i) \models_k \varphi \text{ and } k \geq m
\end{aligned}$$

The semantics of the stratification operator L_k ensures the following:

1. Properties at level k are evaluated using the state sequence σ_k;
2. The temporal resolution at level k is respected (ρ_k);
3. Abstraction consistency is maintained between levels.

3.3. Practical Implications of SMTL Expressiveness

Here, we provide two examples that demonstrate the enhanced expressiveness of SMTL over MTL.

Example 1 (Multi-Scale System Specification)**.** *Consider an autonomous vehicle navigation system operating on multiple time scales. Existing MTL specifications struggle to capture requirements that span different temporal granularities. SMTL naturally expresses such multi-scale requirements through a single coherent formula:*

$$\begin{aligned}
\phi_{nav} = & L_1\Box_{[0,0.01]}(\|a_{actual} - a_{commanded}\| \leq \epsilon) \wedge \\
& L_2\Box_{[0,1]}(speed > v_{min} \rightarrow \Diamond_{[0,0.5]} lane_centered) \wedge \\
& L_3\Box_{[0,60]}(destination_reached \rightarrow \Box_{[0,5]} safely_parked)
\end{aligned}$$

This formula simultaneously captures millisecond-level actuator control (L_1), second-level trajectory tracking (L_2), and minute-level mission objectives (L_3). The stratification operator cleanly separates these concerns while maintaining their logical relationships.

Example 2 (Abstraction in Software Architecture)**.** *The expressiveness of SMTL aligns with layered software architectures. Consider a robotic system with a three-tier architecture:*

$$\begin{aligned}
\phi_{arch} = & L_1(functional_layer_spec) \wedge \\
& L_2(executive_layer_spec) \wedge \\
& L_3(planning_layer_spec)
\end{aligned}$$

where each layer's specification might involve temporal properties. The crucial advantage of SMTL lies in its ability to express inter-layer dependencies:

$$\begin{aligned}\phi_{deps} = L_2\Box_{[0,\infty)}(&plan\_update \rightarrow \\ &L_1\Diamond_{[0,\delta]}(exec\_acknowledge \wedge \\ &\Diamond_{[0,\tau]}functional\_update))\end{aligned}$$

This hierarchical specification captures both the independence of different architectural layers and their temporal coupling, a feat impossible in standard MTL.

4. Theoretical Analysis

Theorem 1 (Stratification Soundness)**.** *For any SMTL formula φ and abstraction levels $i < j$,*

$$(\rho, n) \models_i L_j\varphi \implies (\rho, n) \models_j \varphi$$

Proof. This follows directly from the semantics of L_k and abstraction consistency. When $(\rho, n) \models_i L_j\varphi$, the formula φ is evaluated at level j using σ_j, which, due to abstraction consistency, is a valid abstraction of states at level i. □

Theorem 2 (Temporal Resolution Hierarchy)**.** *For any SMTL formula φ and levels $i < j$,*

$$\text{If } (\rho, n) \models_i \Diamond_{[0,t]}\varphi \text{ then } t \geq \rho_i$$

Proof. According to the definition of stratified timed state sequences, the states at level i must be separated by at least ρ_i time units. Therefore, any temporal property at level i must span at least this duration. □

4.1. Expressiveness of SMTL

We begin by establishing the foundational properties of abstraction functions that underpin the stratified nature of our logic. An abstraction function $\alpha_k : \Sigma \to \Sigma_k$ maps a concrete signal $\sigma \in \Sigma$ to an abstract signal at abstraction level k, where Σ_k represents the signal space at level k. These functions form a hierarchy satisfying crucial properties: monotonicity ensures that for $i < j$, $\alpha_i \circ \alpha_j = \alpha_i$; consistency guarantees that if ϕ holds for $\alpha_i(\sigma)$, then there exists some higher level $j > i$ where ϕ holds for $\alpha_j(\sigma)$; and the refinement property ensures that distinct signals remain distinguishable at some abstraction level.

Theorem 3 (Strict Expressiveness)**.** *SMTL is strictly more expressive than MTL. This relationship manifests both in SMTL's ability to express all MTL properties and in its capacity to express properties that are inexpressible in MTL.*

Proof. The proof proceeds in two parts, first establishing that SMTL subsumes MTL, then demonstrating that SMTL can express properties beyond MTL's capabilities. To show that SMTL subsumes MTL, we construct a translation function τ : MTL $\to$ SMTL defined inductively on the structure of MTL formulas:

$$\begin{aligned}\tau(p) &= p \text{ for } p \in AP \\ \tau(\neg\phi) &= \neg\tau(\phi) \\ \tau(\phi_1 \wedge \phi_2) &= \tau(\phi_1) \wedge \tau(\phi_2) \\ \tau(\phi_1 \mathcal{U}_{[a,b]} \phi_2) &= \tau(\phi_1) \mathcal{U}_{[a,b]} \tau(\phi_2)\end{aligned}$$

Through structural induction, we prove that for any signal σ and time point t, the satisfaction relation preserves equivalence: $(\sigma, t) \models_{\text{MTL}} \phi \iff (\sigma, t) \models_{\text{SMTL}} \tau(\phi)$. The base case for atomic propositions follows directly from the definition. For the inductive step, we consider each operator, with particular attention to the metric unit operator:

$$\begin{aligned}
&(\sigma, t) \models_{\text{MTL}} \phi_1 \mathcal{U}_{[a,b]} \phi_2 \\
&\quad \iff \exists t' \in [t+a, t+b] : (\sigma, t') \models_{\text{MTL}} \phi_2 \wedge \forall t'' \in [t, t') : (\sigma, t'') \models_{\text{MTL}} \phi_1 \\
&\quad \iff \exists t' \in [t+a, t+b] : (\sigma, t') \models_{\text{SMTL}} \tau(\phi_2) \wedge \forall t'' \in [t, t') : (\sigma, t'') \models_{\text{SMTL}} \tau(\phi_1) \\
&\quad \iff (\sigma, t) \models_{\text{SMTL}} \tau(\phi_1) \mathcal{U}_{[a,b]} \tau(\phi_2)
\end{aligned}$$

To establish SMTL's strictly greater expressiveness, we construct an SMTL formula that cannot be expressed in MTL. Consider the formula

$$\psi = L_1 \Box_{[0,1]}(p) \wedge L_2 \Diamond_{[0,2]}(\neg p)$$

We prove its MTL-inexpressibility through a contradiction argument. Let $\Sigma = \{0,1\}^{\mathbb{R}_{\geq 0}}$ be the set of Boolean signals. Define two signals, σ_1 and σ_2, that differ only at a single point:

$$\sigma_1(t) = \begin{cases} 1 & t \in [0,1] \\ 0 & \text{otherwise} \end{cases}$$

$$\sigma_2(t) = \begin{cases} 1 & t \in [0,1] \setminus \{0.5\} \\ 0 & \text{otherwise} \end{cases}$$

Define abstraction functions where α_1 smooths out isolated points while α_2 maintains the original signal:

$$\alpha_1(\sigma)(t) = \begin{cases} 1 & \text{if } \exists \delta > 0 : \forall t' \in (t-\delta, t+\delta), \sigma(t') = 1 \\ 0 & \text{otherwise} \end{cases}$$

$$\alpha_2 = \text{id}$$

In this context, $\alpha_2 = \text{id}$ means (α_2) is the identity function. The identity function is being used at abstraction level 2 to represent "no abstraction" or "keep all details". This contrasts with α_1 which does perform abstraction. Under these definitions, $(\sigma_1, 0) \models_{\text{SMTL}} \psi$, while $(\sigma_2, 0) \not\models_{\text{SMTL}} \psi$. However, any MTL formula θ must be evaluated equivalently on both signals: $(\sigma_1, 0) \models_{\text{MTL}} \theta \iff (\sigma_2, 0) \models_{\text{MTL}} \theta$, as MTL cannot distinguish signals that differ at isolated points. This contradiction establishes that no MTL formula can express the property captured by ψ. □

Corollary 1 (Stratification Power)**.** *Let $\mathcal{L}_{\text{SMTL}}$ be the language of SMTL, and $\mathcal{L}_{\text{MTL}}$ be the language of MTL. From Theorem 3, we know that $\mathcal{L}_{MTL} \subsetneq \mathcal{L}_{SMTL}$, which means the following:*

1. *Every MTL formula φ has an equivalent SMTL formula $\tau(\varphi)$.*
2. *There exists at least one formula $\psi \in \mathcal{L}_{SMTL}$ such that $\psi \notin \mathcal{L}_{\text{MTL}}$.*

Because ψ leverages the stratification operator L_k in a way that cannot be emulated by standard MTL syntax or semantics, the stratification mechanism is strictly more powerful. Hence, the operator L_k effectively extends the expressive frontier of the logic, allowing one to specify behaviors or constraints at multiple abstraction levels that cannot be collapsed into a single, uniform time scale.

Corollary 2 (Multi-Scale Properties)**.** *Consider a CPS model with a hierarchy of time resolutions* $\{\rho_1, \rho_2, \ldots, \rho_K\}$ *and corresponding state abstractions* $\{\alpha_k\}_{k=1}^{K}$*. From Definition 5 (abstraction consistency) and the semantics of* L_k*, if a property* ϕ_k *is designated at level k, then it is evaluated (i) using time steps separated by at least* ρ_k*, and (ii) in a state space* σ_k *consistent with* α_k*. Thus, a single SMTL specification*

$$\phi = \bigwedge_{k=1}^{K} L_k(\phi_k)$$

can encode requirements ϕ_k *at multiple time granularities. No purely single-scale temporal logic (MTL/STL) can replicate this multifaceted time-resolution property in a single unified formula without artificially inflating the timescale or losing key abstraction details. As a result, SMTL natively captures multiscale temporal requirements under one logical framework.*

Definition 12 (Bounded-Time SMTL Formula)**.** *An SMTL formula* φ *is called* bounded-time *if, for every temporal operator in* φ *(e.g.,* $\mathcal{U}_I, \diamond_I, \Box_I$*), the corresponding time interval I is a subset of* $[0, T]$ *for some finite* $T \in \mathbb{Q}_{\geq 0}$*. Equivalently, no time interval may be of the form* $[a, \infty)$ *or* (a, ∞)*.*

This definition confines verification to finite time horizons for each temporal operator. By contrast, a lemph{unbounded-time} formula is one that includes at least one interval $[a, \infty)$ or (a, ∞).

Theorem 4 (SMTL Model-Checking Complexity)**.** *Let* ϕ *be an SMTL formula and* $\mathcal{M}$ *be a finite-state transition system. Denote as K the maximum abstraction level in* ϕ*, and as* $c_{\max}$ *the largest integer constant appearing in its temporal intervals. The complexity of checking* $\mathcal{M} \models \phi$ *satisfies the following:*

- (*Bounded-Time Case) For formulas with only bounded intervals, the model-checking problem is EXPTIME-complete.*
- *(Unbounded-Time Case) If* ϕ *includes at least one unbounded interval, the model-checking problem is 2EXPTIME-complete.*

Moreover, the complexity grows polynomially in K.

4.2. Proof Sketch

The bounded-time result (EXPTIME-complete) follows from constructing a region automaton where each clock is bounded by the largest integer constant $c_{\max}$ in ϕ. Because the transitions and clock valuations remain finite under bounded intervals, the automaton state space can be explored in exponential time with respect to $\|\mathcal{M}\|$ and $|\phi|$. For each abstraction level k, we build an automaton of comparable size, yielding an overall EXPTIME procedure multiplied by a factor polynomial in K. For the unbounded-time case, we note that SMTL strictly generalizes MTL. It is known that model checking unbounded MTL is complete in 2EXPTIME, due to the necessity of handling infinite paths and the resulting exponential blow-ups when automating temporal logic over infinite horizons. We can reduce the unbounded MTL problem to SMTL by embedding an MTL formula ϕ_{MTL} into SMTL (see Theorem 3). Therefore, no asymptotically lower complexity can be applied to an unbounded SMTL, establishing a 2EXPTIME-hard lower bound. Since our upper bound construction also leads to a 2EXPTIME procedure, this completes the 2EXPTIME-completeness result for unbounded-time SMTL formulas.

Corollary 3. *The 2EXPTIME-hardness for unbounded-time SMTL directly follows from the fact that there exists a linear-time many-to-one reduction from unbounded MTL model checking (itself 2EXPTIME-complete) to SMTL model checking.*

In summary, if all intervals are finite, the complexity is contained in EXPTIME, whereas unbounded intervals invoke the 2EXPTIME-complete complexity class. This aligns with classical results on MTL, thus confirming both the bounded- and unbounded-time complexity claims for SMTL.

Corollary 4 (Incremental Verification)**.** *Let ϕ be an SMTL formula with maximum abstraction level K, and let $\mathcal{M}$ be a finite-state transition system. Suppose $\mathcal{M} \models \phi$ has already been verified. Now, consider a new SMTL formula ϕ' obtained from ϕ by adding or modifying constraints solely at a new level $K+1$. Formally,*

$$\phi' = \phi \wedge \bigwedge_{i \in I} L_{K+1}(\psi_i)$$

where each ψ_i is a fresh subformula (or a refinement of existing constraints) that did not previously appear in ϕ, and no constraints at levels $\leq K$ are altered. Then, to check whether $\mathcal{M} \models \phi'$, it suffices to carry out the following: 1. Verify the newly introduced subformulas $\{\psi_i\}$ at level $K+1$ within the abstract transition system $\alpha_{K+1}(\mathcal{M})$. 2. Check the consistency between levels $K+1$ and K, ensuring that $\sigma_{K+1,n}$ correctly abstracts to $\sigma_{K,n}$ for each state index n, in accordance with Definition 5 (abstraction consistency). Because the properties at the existing levels $1, 2, \ldots, K$ remain unchanged—and have already been verified—no further verification is required for those levels. As a result, the incremental-verification cost is polynomial in the size of ϕ' and $\|\mathcal{M}\|$ (as opposed to repeating a full model check from scratch). Hence, adding each new level only imposes overhead for the newly introduced constraints and their immediate interactions with the immediately lower level, confirming an efficient incremental-verification process.

5. Numerical Results

To evaluate the effectiveness of SMTL in enhancing multi-agent coordination and safety, we conducted a series of simulations comparing the performance of agents operating under SMTL and MTL specifications. The experiments were designed to assess how the introduction of stratification in temporal logic influences agent behavior in terms of collision avoidance and path efficiency. In our experimental setup, agents navigated grid worlds of dimensions ranging from 5×5 to 150×150. Each grid represented a discrete environment in which agents moved from randomly assigned starting positions to specified target locations. Obstacles were randomly placed within the grids. Agents were programmed to move one unit per time step in any of the four cardinal directions, provided the destination cell was within the grid boundaries and not occupied by an obstacle or another agent. The primary objective for each agent was to reach its goal while avoiding collisions with other agents and obstacles.

Under the MTL framework, agents were governed by temporal logic specifications that dictated their movement towards goals without consideration of other agents. The MTL specification used can be mathematically expressed as

$$\phi_{\text{MTL}} = \Box_{[0,T]} \left(\text{reachGoal} \rightarrow \diamond_{[0,T]} \text{atGoal}\right)$$

This formula specifies that globally, from time 0 to T, if an agent intends to reach its goal (reachGoal), then eventually, within time T, it must be at the goal position (atGoal). In contrast, agents operating under SMTL incorporated stratified reasoning to anticipate and avoid potential conflicts with other agents. The SMTL specification extended the MTL formula to include collision avoidance at a higher level of reasoning:

$$\phi_{\text{SMTL}} = L_1\Big(\phi_{\text{MTL}} \wedge \Box_{[0,T]}\Big(\forall j \neq i, \neg\Big(\text{pos}_i = \text{pos}_j\Big)\Big)\Big)$$

Here, L_1 represents the stratification level where agents consider not only their goal-reaching objectives but also the positions of other agents to prevent collisions. The term $\Box_{[0,T]}\left(\forall j \neq i, \neg\left(\text{pos}_i = \text{pos}_j\right)\right)$ ensures that at all times, agents avoid occupying the same position as any other agent. In this case study, the stratified nature of SMTL matches its temporal stratification through the separation of agent behaviors and constraints according to their temporal characteristics:

- **Short-Term Temporal Constraints (Fine-Grained Temporal Level):** Agents using SMTL enforce collision avoidance at every time step. This is a temporal property that requires agents to ensure that they do not occupy the same position as any other agent. In SMTL, this corresponds to a higher-priority constraint specified at a lower stratification level (e.g., L_1), which operates at a finer temporal granularity. The logic enforces that safety properties hold at all times or within very short time intervals.
- **Long-Term Temporal Objectives (Coarse-Grained Temporal Level):** Agents aim to reach their designated goals, which is a temporal property considered over a longer time horizon. In SMTL, this is captured at a higher stratification level (e.g., L_2 or L_3), where the temporal properties involve longer intervals or eventualities.

The experimental results presented in Figure 1 provide a comparison of agent performance under MTL and SMTL. For each grid size, the number of agents corresponds to the grid dimension, ensuring a consistent agent density across different environment scales. The key performance metrics evaluated include (i) collision rate, (ii) average path length, (iii) path efficiency, (iv) and average waits due to others. A key observation from the data is the contrast in collision rates between the MTL and SMTL agents. MTL agents show a non-zero collision rate across all grid sizes, with the collision rate generally increasing as the grid size and the number of agents grow. Specifically, MTL agents experience an average collision rate of 5.20 collisions per agent on the 5×5 grid, which escalates to over 81 collisions per agent in the 150×150 grid.

In contrast, SMTL agents maintain a collision rate of zero across all grid sizes, demonstrating their effectiveness in collision avoidance regardless of the environment's scale. This result highlights the efficacy of the stratification approach in SMTL, which allows agents to prioritize safety and coordination over individual objectives. The ability of SMTL agents to completely avoid collisions, even in densely populated and extensive grids, signifies a substantial improvement over traditional MTL agents and validates the practical utility of SMTL in multi-agent systems. Analyzing the average path length reveals that SMTL agents often have shorter or comparable path lengths compared to MTL agents, especially in smaller grids. For example, on the 5×5 grid, SMTL agents achieve an average path length of 8.40 units, whereas MTL agents average 10.47 units. This suggests that SMTL agents are capable of reaching their goals more efficiently in less congested environments. However, as the grid size increases, the difference in average path length between SMTL and MTL agents diminishes. In the 150×150 grid, SMTL agents have an average path length of approximately 558.57 units, compared to 561.24 units for MTL agents. This convergence indicates that in larger environments, both agent types require similar path lengths to reach their goals, likely due to the proportional increase in distances that need to be traversed. The path efficiency metric, calculated as the ratio of the shortest possible path length to the actual path length, provides insight into the agents' navigational effectiveness relative to the optimal path. SMTL agents consistently show higher path efficiency percentages than MTL agents across all grid sizes. In the 5×5 grid, SMTL agents attain a path efficiency of approximately 40.37%, surpassing the 34.68% efficiency of MTL agents. Although both agent types experience a decline in path efficiency as the grid size increases—reflecting the greater complexity and potential detours in larger environments—SMTL agents maintain a slight advantage. This efficiency suggests that SMTL agents effectively balance the need for

collision avoidance with the pursuit of efficient paths. A notable distinction between the two agent types is observed in the average waits due to others. SMTL agents exhibit a substantial number of waits, which increases with grid size—from an average of 5.93 waits in the 5×5 grid to over 517 waits in the 150×150 grid. This behavior reflects the SMTL agents' strategy of waiting to avoid potential collisions, emphasizing their prioritization of safety and adherence to collision avoidance constraints. In contrast, MTL agents have zero waits across all grid sizes, indicating that they proceed toward their goals without regard for other agents, even if it results in collisions. The willingness of SMTL agents to wait highlights their commitment to coordination and safe operation within shared environments.

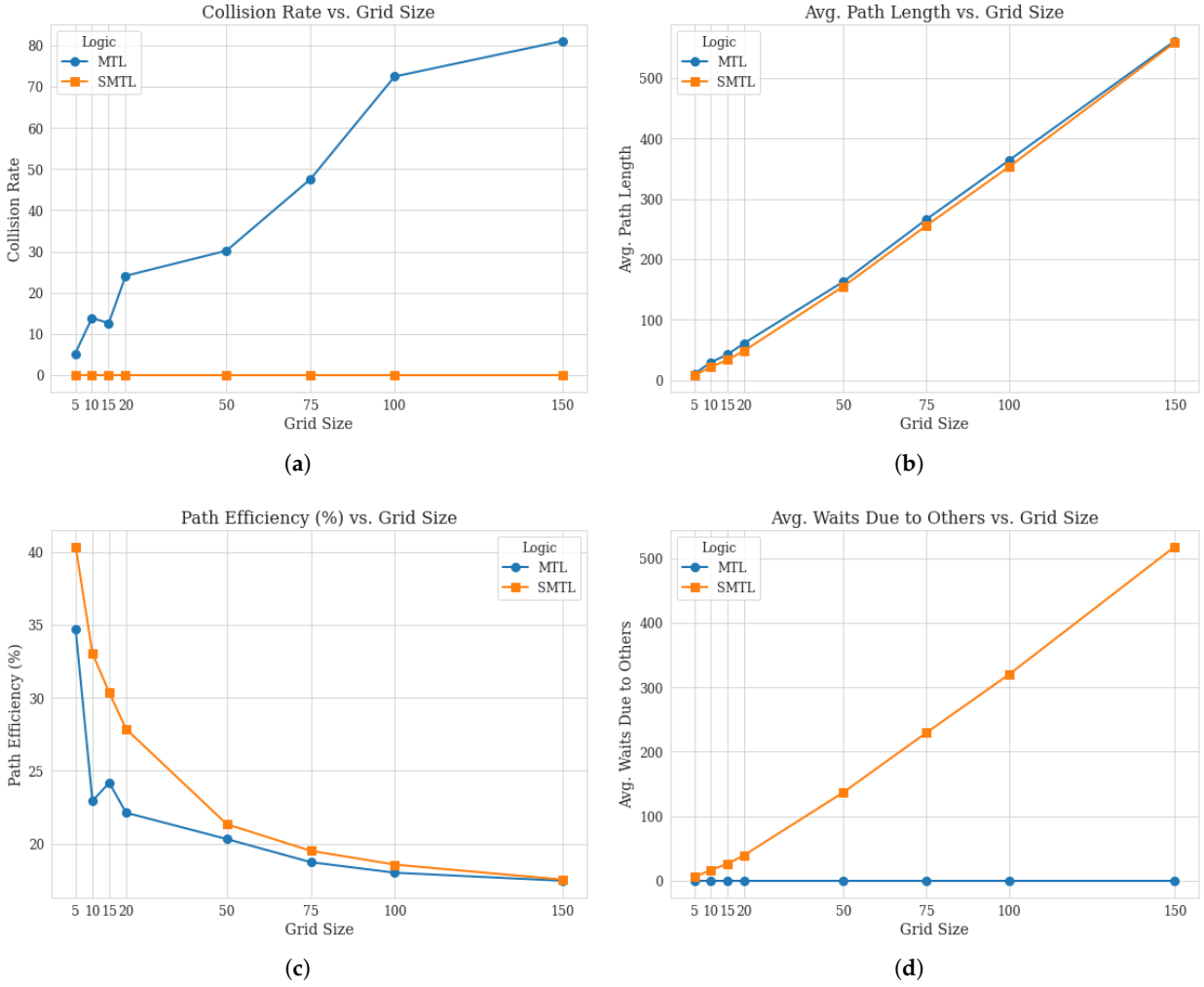

Figure 1. Comparison of agent performance metrics under MTL and SMTL across different grid sizes: (**a**) collision rate, (**b**) average path length, (**c**) path efficiency, and (**d**) average waits due to others.

Computation time per step per agent is another metric where SMTL agents demonstrate competitive performance. Figure 2 shows this comparison. While SMTL agents incur slightly higher computation times compared to MTL agents, the difference is marginal. For example, in the 150×150 grid, SMTL agents have an average computation time of approximately 0.935 milliseconds per step, compared to 0.489 milliseconds for MTL agents. This increase can be attributed to the additional processing required for collision avoidance and coordination with other agents. These results indicate that the coordination achieved by SMTL agents does not come at the expense of prohibitive computational costs.

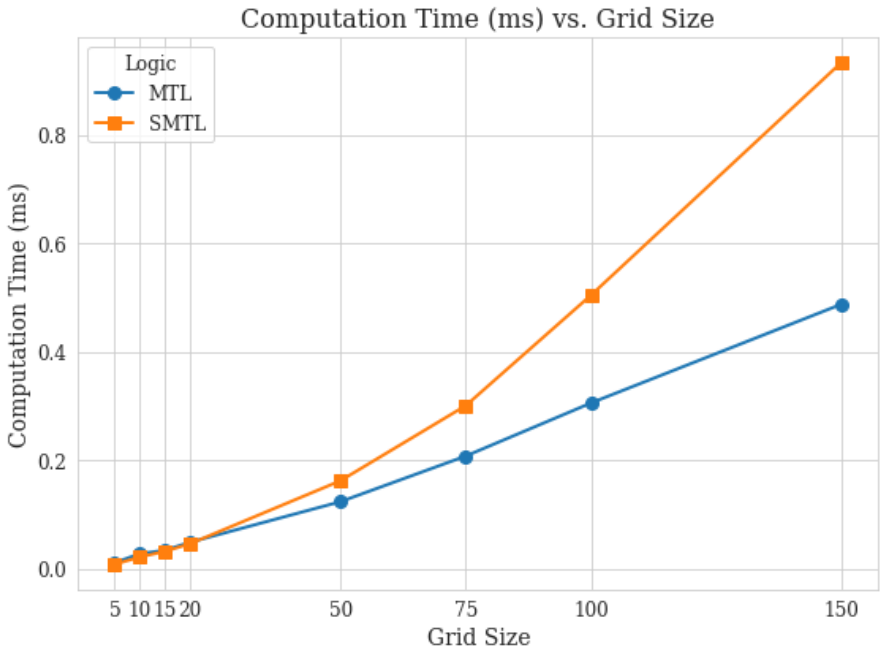

Figure 2. Comparison of computation time per step between two agents.

6. Conclusions

In this paper, we proposed SMTL as a formalism for specifying and verifying properties of complex cyber–physical systems exhibiting behaviors across multiple temporal and abstraction scales. By incorporating a stratification operator into traditional temporal logics, SMTL enables the association of temporal properties with specific abstraction levels, facilitating the natural expression of multi-scale requirements while preserving formal reasoning about inter-level relationships. We formalized the syntax and semantics of SMTL, proving that it strictly subsumes MTL and offers enhanced expressiveness by capturing properties unattainable in existing logics. Through numerical simulations, we demonstrated the utility of SMTL in improving agent coordination and safety. The simulations comparing agents operating under MTL and SMTL specifications revealed that SMTL significantly reduces collision rates without incurring substantial computational overhead or compromising path efficiency. Specifically, SMTL agents maintained a zero collision rate at various grid sizes, highlighting the effectiveness of stratified reasoning in avoiding conflicts and improving overall system reliability. Our findings emphasize the potential of SMTL as a valuable tool for designing and verifying complex multi-agent systems operating across diverse temporal and abstraction scales.

Future Work

In future work, we plan to explore integrations and comparisons between SMTL and other hierarchical methods (e.g., MLTLM and CBS), aiming to assess how these approaches complement or further enhance SMTL's multi-scale coordination capabilities. Furthermore, we plan to explore the integration of SMTL with automated synthesis tools to generate controllers that guarantee compliance with multi-scale specifications. Furthermore, investigating the application of SMTL in other domains, such as distributed sensor networks or human–robot interaction, could further validate its effectiveness. Extending SMTL to incorporate probabilistic reasoning or learning-based components may also enhance its applicability to systems with inherent uncertainties. Finally, the development of optimized algorithms for the checking of the SMTL model can facilitate its adoption in large-scale industrial applications.

Author Contributions: Conceptualization, A.B. and P.W.; methodology, A.B.; software, A.B.; validation, A.B.; formal analysis, A.B.; investigation, A.B.; resources, A.B.; data curation, A.B.; writing—original draft preparation, A.B.; writing—review and editing, A.B. and P.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Bartocci, E.; Deshmukh, J.; Donzé, A.; Fainekos, G.; Maler, O.; Ničković, D.; Sankaranarayanan, S. Specification-based monitoring of cyber-physical systems: A survey on theory, tools and applications. In *Lectures on Runtime Verification: Introductory and Advanced Topics*; Springer: Cham, Switzerland, 2018; pp. 135–175.
2. Eappen, J.; Xiong, Z.; Patel, D.; Bera, A.; Jagannathan, S. Scaling Safe Multi-Agent Control for Signal Temporal Logic Specifications. In Proceedings of the CoRL Workshop on Safe and Robust Robot Learning for Operation in the Real World, Munich, Germany, 6–9 November 2024.
3. 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), San Antonio, TX, USA, 9–12 May 2023; pp. 154–165.
4. Yoo, C.; Belta, C. Control with probabilistic signal temporal logic. *arXiv* **2015**, arXiv:1510.08474.
5. Fainekos, G.E.; Pappas, G.J. Robustness of temporal logic specifications. In *Formal Approaches to Software Testing and Runtime Verification, Proceedings of the First Combined International Workshops FATES 2006 and RV 2006, Seattle, WA, USA, 15–16 August 2006*; Revised Selected Papers; Springer: Berlin/Heidelberg, Germany, 2006; pp. 178–192.
6. Fisher, M. Temporal development methods for agent-based. *Auton. Agents Multi-Agent Syst.* **2005**, *10*, 41–66. [CrossRef]
7. Nilsson, P.; Ozay, N. Control synthesis for large collections of systems with mode-counting constraints. In Proceedings of the 19th International Conference on Hybrid Systems: Computation and Control, Vienna, Austria, 12–14 April 2016; pp. 205–214.
8. Sahin, Y.E.; Nilsson, P.; Ozay, N. Provably-correct coordination of large collections of agents with counting temporal logic constraints. In Proceedings of the 2017 ACM/IEEE 8th International Conference on Cyber-Physical Systems (ICCPS), Pittsburgh PA, USA, 18–20 April 2017.
9. Xu, Z.; Julius, A.A. Census signal temporal logic inference for multiagent group behavior analysis. *IEEE Trans. Autom. Sci. Eng.* **2016**, *15*, 264–277. [CrossRef]
10. Leahy, K.; Jones, A.; Vasile, C.I. Fast decomposition of temporal logic specifications for heterogeneous teams. *IEEE Robot. Autom. Lett.* **2022**, *7*, 2297–2304. [CrossRef]
11. Kantaros, Y.; Zavlanos, M.M. Stylus*: A temporal logic optimal control synthesis algorithm for large-scale multi-robot systems. *Int. J. Robot. Res.* **2020**, *39*, 812–836. [CrossRef]
12. Lamport, L. The temporal logic of actions. *ACM Trans. Program. Lang. Syst. (TOPLAS)* **1994**, *16*, 872–923. [CrossRef]
13. Alur, R.; Henzinger, T.A. Real-time logics: Complexity and expressiveness. *Inf. Comput.* **1993**, *104*, 35–77. [CrossRef]
14. Endriss, U. Temporal logics for representing agent communication protocols. In Proceedings of the International Workshops on Agent Communication, Utrecht, The Netherlands, 25 July 2005; Springer: Berlin/Heidelberg, Germany, 2005; pp. 15–29.
15. Kress-Gazit, H.; Lahijanian, M.; Raman, V. Synthesis for robots: Guarantees and feedback for robot behavior. *Annu. Rev. Control. Robot. Auton. Syst.* **2018**, *1*, 211–236. [CrossRef]
16. Cucala, D.J.T.; Wałęga, P.A.; Grau, B.C.; Kostylev, E. Stratified negation in Datalog with metric temporal operators. In Proceedings of the AAAI Conference on Artificial Intelligence, Online, 2–9 February 2021; Volume 35, pp. 6488–6495.
17. Felli, P.; Gianola, A.; Montali, M. SMT-based safety checking of parameterized multi-agent systems. In Proceedings of the AAAI Conference on Artificial Intelligence, Online, 2–9 February 2021; Volume 35, pp. 6321–6330.
18. Sharon, G.; Stern, R.; Felner, A.; Sturtevant, N.R. Conflict-based search for optimal multi-agent pathfinding. *Artif. Intell.* **2015**, *219*, 40–66. [CrossRef]
19. Belta, C.; Sadraddini, S. Formal methods for control synthesis: An optimization perspective. *Annu. Rev. Control. Robot. Auton. Syst.* **2019**, *2*, 115–140. [CrossRef]
20. Shahrooei, Z.; Kochenderfer, M.J.; Baheri, A. Optimizing Falsification for Learning-Based Control Systems: A Multi-Fidelity Bayesian Approach. *arXiv* **2024**, arXiv:2409.08097.
21. Baheri, A. Safety Validation of Learning-Based Autonomous Systems: A Multi-Fidelity Approach. In Proceedings of the AAAI Conference on Artificial Intelligence, Washington, DC, USA, 7–14 February 2023; Volume 37, p. 15432.

22. Baheri, A. Exploring the role of simulator fidelity in the safety validation of learning-enabled autonomous systems. *AI Mag.* **2023**, *44*, 453–459. [CrossRef]
23. Shahrooei, Z.; Kochenderfer, M.J.; Baheri, A. Falsification of Learning-Based Controllers through Multi-Fidelity Bayesian Optimization. In Proceedings of the European Control Conference (ECC), Bucharest, Romania, 13–16 June 2023.
24. Beard, J.J.; Baheri, A. Safety verification of autonomous systems: A multi-fidelity reinforcement learning approach. *arXiv* **2022**, arXiv:2203.03451.
25. Hariharan, G.; Kempa, B.; Wongpiromsarn, T.; Jones, P.H.; Rozier, K.Y. MLTL multi-type (MLTLM): A logic for reasoning about signals of different types. In Proceedings of the International Workshop on Numerical Software Verification, Haifa, Israel, 11 August 2022; Springer: Cham, Switzerland, 2022; pp. 187–204.