Synthesizing Efficiently Monitorable Formulas in Metric Temporal Logic

Ritam Raha[1,2], Rajarshi Roy[3], Nathanaël Fijalkow[2], Daniel Neider[4,5], and Guillermo A. Pérez[1]

[1] University of Antwerp, Flanders Make, Belgium
[2] LaBRI, University of Bordeaux, Bordeaux, France
[3] Max Planck Institute for Software Systems, Kaiserslautern, Germany
[4] Technical University of Dortmund, Germany
[5] Center for Data Science and Security, University Alliance Ruhr, Germany

Abstract. In runtime verification, manually formalizing a specification for monitoring system executions is a tedious and error-prone process. To address this issue, we consider the problem of automatically synthesizing formal specifications from system executions. To demonstrate our approach, we consider the popular specification language Metric Temporal Logic (MTL) which is particularly tailored towards specifying temporal properties for cyber-physical systems (CPS). Most of the classical approaches for synthesizing temporal logic formulas aim at minimizing the size of the formula. However, for efficiency in monitoring, along with the size, the amount of "lookahead" required for the specification becomes relevant, especially for safety-critical applications. We formalize this notion and devise a learning algorithm that synthesizes concise formulas having bounded lookahead. To do so, our algorithm reduces the synthesis task to a series of satisfiability problems in Linear Real Arithmetic (LRA) and generates MTL formulas from their satisfying assignments. The reduction uses a novel encoding of a popular MTL monitoring procedure using LRA. Finally, we implement our algorithm in a tool called `TEAL` and demonstrate its ability to synthesize efficiently monitorable MTL formulas in a CPS application.

1 Introduction

Runtime verification is a well-established method for ensuring the correctness of cyber-physical systems during runtime. Techniques in runtime verification are known to be more rigorous than conventional testing while not being as resource-intensive as exhaustive formal verification [20]. In the field of runtime verification, among other techniques, monitoring system executions against formal specifications during runtime is a widely used one. Over the years, numerous monitoring techniques have been proposed for a variety of specification languages [28,23,21,7].

In this work, we focus on Metric Temporal Logic (MTL) [37]—a specification language popularly employed for monitoring cyber-physical systems [29,41].

MTL is a real-time extension of Linear Temporal Logic (LTL) [49] augmented with timing constraints for temporal operators. MTL specifications are often easy to interpret due to their resemblance to natural language and, thus, also find applications in Artificial Intelligence [56]. While there are many possible semantics of MTL (e.g., discrete, dense-time pointwise, etc. [48]), we employ the dense-time continuous semantics as it is more natural and general than the counterparts [9,4]. We expand on MTL and other prerequisites in Section 2.

Virtually all verification techniques for MTL rely on the availability of a formal specification. However, manually writing specifications is a tedious and error-prone task [1]. Synthesizing functional, correct, and interpretable specifications that precisely express the design requirements has been one of the major challenges in the adoption of formal techniques for verification [12,53].

To tackle the lack of formal specifications, there have been efforts to automatically synthesize specifications from system executions. Most of the existing works have targeted specification languages such as Linear Temporal Logic (LTL) [16,46,50] and Signal Temporal Logic (STL) [3,39,43,54], with few works for MTL [30,56]. Many of the works tend to synthesize specifications that are *concise* in size. Concise specifications are preferred over large ones because, based on the principle of Occam's razor, they are easier for humans to understand [52].

However, conciseness is not the only measure of interest for specifications, especially in the context of online monitoring. In online monitoring, specifically in *stream-based* runtime monitoring, a monitor reads an execution as a stream of data and verifies if a given specification is invariant (i.e., holds at all time points) in the execution. Many stream-based monitors [27,32,38] support MTL formulas. Typically, such monitors produce a stream of (Boolean) verdicts with some "latency", which depends on the lookahead of the formula. The lookahead required for an MTL formula is often formalized as its *future-reach* [29,31], which is the amount of time required to determine its satisfaction at any time point.

With the aim of reducing the latency for efficient online monitoring, we focus on automatically synthesizing MTL specifications based on two regularizers, size and future-reach. As input data, we rely on a sample $\mathcal{S}$ consisting of executions of a system that are observed for a finite duration. We consider the sample to be partitioned into a set P of positive (or desirable) executions and a set N of negative (or undesirable) executions.

We now formulate the central problem of synthesizing MTL formulas as follows: given a sample $\mathcal{S} = (P, N)$ and a future-reach bound K, synthesize a minimal size MTL formula φ that (i) is *globally-separating* for $\mathcal{S}$, in that φ holds at all time points in the positive executions and does not hold at some time point in the negative executions, and (ii) the future-reach of φ is smaller than K. The property of being globally-separating for $\mathcal{S}$ ensures that prospective formula φ is invariant in the desirable executions and not in the undesirable executions, as is typically preferred in specifications for online monitoring [11]. We expand on the problem formulation in Section 3.

Also, interestingly, without a future-reach bound, the most concise MTL formula that can be synthesized can have a large future-reach value, increasing the latency required for online monitoring. To illustrate this, assume that we observe some simulations of an autonomous vehicle. During the simulations, we sample executions (shown below) of the vehicle every second for six seconds. We classify them as positive (denoted using u_i's) or negative (denoted using v_i's) based on whether the vehicle encountered a collision or not.

	0	1	2	3	4	5
u_1:	$\{p,q\}$	$\{p\}$	$\{q\}$	$\{p,q\}$	$\{p\}$	$\{p\}$
u_2:	$\{q\}$	$\{\}$	$\{q\}$	$\{p\}$	$\{p\}$	$\{p,q\}$
v_1:	$\{p\}$	$\{q\}$	$\{\}$	$\{\}$	$\{\}$	$\{\}$
v_2:	$\{p\}$	$\{p,q\}$	$\{p\}$	$\{\}$	$\{p\}$	$\{\}$

In the executions, we use p to denote that there is no obstacle within a particular unsafe distance ahead of the vehicle and q to denote that the vehicle's brake is triggered. Our setting considers executions to be *continuous*. Thus, to ensure continuity of execution, in the above example, if p occurs at time point t, we interpret it as p holding during the entire interval $[t, t+1)$. We also assume that the executions last up to a final time point T which is 6 for this example. Thus, for the execution u_1, p holds in the intervals $[0, 2)$ and $[3, 6)$.

In the sample, a minimal globally separating formula is $\varphi_1 = \mathbf{F}_{[0,3]}\, q$. The formula φ_1 being globally separating indicates that in all positive executions, the brake is triggered every three seconds (i.e., within the interval $[t, t+3]$ for every time point t), irrespective of whether there is an obstacle within the unsafe distance. The formula φ_1 has size two and a future-reach of three seconds, meaning that any online monitor requires a three second lookahead window to check the satisfaction of φ_1. There is another formula $\varphi_2 = \neg p \rightarrow \mathbf{F}_{[0,1]}\, q$ that is globally separating for the sample. The formula φ_2 being globally-separating indicates that in all positive executions, for every time point t, if an obstacle is within the unsafe distance, then the brake is triggered within one second (i.e., within the interval $[t, t+1]$). Although of size five, φ_2 has future-reach of one second and will be typically preferred over φ_1 for online monitoring in a safety-critical scenario.

For the problem of synthesizing MTL formulas, we first study whether a solution exists. It turns out that there are samples $\mathcal{S}$ and future-reach bound K for which there might not exist any formula that is globally-separating for $\mathcal{S}$ and has future-reach within K. To aid in checking whether a prospective formula exists, we identify a simple characterization of $\mathcal{S}$ based on the future-reach K. Such a characterization enables us to design an NP algorithm that can decide whether a prospective algorithm exists. Also, it provides an upper-bound, which is polynomial in the inputs $\mathcal{S}$ and K, on the size of the prospective formula if one exists. We mention the details of the existence check in Section 4.

To synthesize a prospective formula, we rely on a reduction to constraint satisfaction problems. In particular, following other works in synthesis of formulas [46,52], our algorithm encodes the problem in a series of satisfiability modulo theory (SMT) problems in Linear Real Arithmetic (LRA). To our knowledge, we design the first SMT-based algorithm that can synthesize MTL formulas of arbitrary syntactic structure. Such an SMT-based algorithm allows us to extend our algorithm to work for other settings that are common in the synthesis of formulas [26,40].

Further, we analyze the complexity of the decision version of the problem of synthesizing MTL formulas. While the exact complexity lower bounds are open, we show that the corresponding decision problem is in NP. The central SMT-based algorithm with all the theoretical results is in Section 5.

We also implement our algorithm using a popular SMT solver in a prototype named `TEAL`. We evaluate the ability of `TEAL` to synthesize MTL formulas typically employed for monitoring cyber-physical systems. We also empirically study the interplay between the size and future-reach of a formula. We present all the experimental results in Section 6.

Related works. To our knowledge, there are only a limited number of works for synthesizing MTL formulas. One of them [56] infers MTL formulas as decision trees for representing task knowledge in Reinforcement Learning. Some other works [30,57] consider the parameter search problem for MTL where, given a parametric MTL formula (i.e., an MTL formula with missing temporal bounds), they infer the ranges of parameters where the formula holds/does not hold on a given system. Unlike our work, none of these works aims at synthesizing concise MTL specifications for monitoring tasks.

There are, nevertheless, numerous runtime monitoring procedures for MTL [55,4,22,29,10,17,33,38], clearly indicating the need for efficiently monitorable MTL specifications. Many of them also rely on the future-reach of a specification [29,10] or other similar measures (e.g., horizon [22], worst-case propagation delay [33], etc.) to quantify the efficiency of their monitoring procedure.

Interestingly, several works focus on synthesizing formulas in STL, an extension of MTL to reason about real-valued signals. Bartocci et al. [8] provide a comprehensive survey of the existing works on inferring STL. Many of them [3,35,34] solve the parameter search for STL, while others [14,13] learn decision trees over STL formulas, which typically do not result in concise formulas. There are few works [43,47] that do prioritize the conciseness of formulas during inference. These works cannot be directly applied to solve our problem for two main reasons. First, these works assume inputs to be *piecewise-affine continuous* signals. While the above assumption is natural for synthesizing STL formulas inference from real-valued signals, in our setting, we must rely on the assumption that our inputs are *piecewise-constant* signals, which is natural for Boolean-valued signals. Second, these works do not employ any measure, apart from conciseness, that directly influences the efficiency of runtime monitoring.

Finally, there are works on synthesizing formulas in other temporal logics such as Linear Temporal Logic (LTL) [46,51,16,50], Property Specification Language (PSL) [52], etc., which are not easily extensible to our setting.

2 Preliminaries

In this section, we introduce the basic notations used throughout the paper.

Signals and Prefixes. We represent continuous system executions as signals. A *signal* $\boldsymbol{x}\colon \mathbb{R}_{\geq 0} \to 2^{\mathcal{P}}$ over a set of propositions $\mathcal{P}$ is an infinite time series that describes relevant system events over time. A prefix of a signal $\boldsymbol{x}$ restricted to domain $\mathbb{T} = [0, T), T \in \mathbb{R}_{\geq 0}$ is a function $\boldsymbol{x}_{\mathbb{T}}\colon \mathbb{T} \to 2^{\mathcal{P}}$ where $\boldsymbol{x}_{\mathbb{T}}(t) = \boldsymbol{x}(t)$ for all $t \in \mathbb{T}$.

To synthesize MTL formulas, we rely on finite observations that are sequences of the form $\Omega = \langle (t_i, \delta_i) \rangle_{i \leq n}$, $n \in \mathbb{N}$ such that (i) $t_0 = 0$, (ii) $t_n < T$, and (ii) for all $i \leq n$, $\delta_i \subseteq \mathcal{P}$ is the set of propositions that hold at time point t_i. To construct well-defined signal prefixes, we approximate each observation Ω as a *piecewise-constant* signal prefix $\boldsymbol{x}^{\Omega}_{\mathbb{T}}$ using interpolation as: (i) for all $i < n$, for all $t \in [t_i, t_{i+1})$, $\boldsymbol{x}_{\mathbb{T}}(t) = \delta_i$; and (ii) for all $t \in [t_n, T)$, $\boldsymbol{x}_{\mathbb{T}}(t) = \delta_n$. For brevity, we refer to signal prefixes simply as 'prefixes' when clear from the context.

Metric Temporal Logic. MTL is a logic formalism for specifying real-time properties of a system. We consider the following syntax of MTL:

$$\varphi := p \in \mathcal{P} \;\mid\; \neg p \;\mid\; \varphi_1 \wedge \varphi_2 \;\mid\; \varphi_1 \vee \varphi_2 \;\mid\; \varphi_1 \, \mathbf{U}_I \, \varphi_2 \;\mid\; \mathbf{F}_I \, \varphi \;\mid\; \mathbf{G}_I \, \varphi$$

where $p \in \mathcal{P}$ is a proposition, $\neg$ is the negation operator, $\wedge$ and $\vee$ are the conjunction and disjunction operators respectively, and $\mathbf{U}_I, \mathbf{F}_I$ and $\mathbf{G}_I$ are the timed-Until, timed-Finally and timed-Globally operators respectively. Here, I is a closed interval of non-negative real numbers of the form $[a, b]$ where $0 \leq a \leq b$[6]. Note that the syntax is presented in *negation normal form*, meaning that the $\neg$ operator can only appear before a proposition.

As a syntactic representation of an MTL formula, we rely on *syntax-DAGs*. A syntax-DAG is similar to the parse tree of a formula but with shared common subformulas. We define the size $|\varphi|$ of an MTL formula φ as the number of nodes in its syntax-DAG, e.g., the size of $(p \wedge \mathbf{G}_I \, q) \vee (\mathbf{F}_I \, p)$ is six as its syntax-DAG has six nodes, as shown in Figure 1.

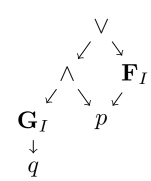


Fig. 1: Syntax DAG of $(p \wedge \mathbf{G}_I \, q) \vee (\mathbf{F}_I \, p)$

As mentioned already, we follow the continuous semantics of MTL. For the standard continuous semantics ($\models$) of MTL over infinite signals, we refer to the work of [48] and provide detailed descriptions in Appendix A. However, our setting demands a semantics of MTL over finite prefixes such that the synthesized formulas will be 'useful' while monitoring over infinite signals. Intuitively, we want an 'optimistic' semantics ($\models_{\mathrm{f}}$) of an MTL formula φ over a prefix $\boldsymbol{x}_{\mathbb{T}}$ such that $\boldsymbol{x}_{\mathbb{T}} \models_{\mathrm{f}} \varphi$ if there exists an infinite signal *extending* $\boldsymbol{x}_{\mathbb{T}}$ that satisfies φ. In other words, $\boldsymbol{x}_{\mathbb{T}}$ "carries no evidence against" the formula φ. Formally, we want the definition of $\models_{\mathrm{f}}$ to satisfy the following lemma.

Lemma 1. *Given a prefix $\boldsymbol{x}_{\mathbb{T}}$, let $ext(\boldsymbol{x}_{\mathbb{T}}) = \{\boldsymbol{x} \mid \boldsymbol{x}_{\mathbb{T}}$ is a prefix of $\boldsymbol{x}\}$ be the set of all infinite extensions of $\boldsymbol{x}_{\mathbb{T}}$. Then given an MTL formula φ, $\boldsymbol{x}_{\mathbb{T}} \models_{\mathrm{f}} \varphi$ if there exists $\boldsymbol{x} \in ext(\boldsymbol{x}_{\mathbb{T}})$ such that $\boldsymbol{x} \models \varphi$.*

[6] Since we infer MTL formulas with bounded lookahead, we restrict I to be bounded.

The proof of the above lemma proceeds via structural induction over φ, which is describe in Appendix B.

Towards this, we follow the idea of 'weak semantics' of MTL defined in [29][7] and interpret MTL over finite prefixes. Given a prefix $\boldsymbol{x}_{\mathbb{T}}$, we inductively define when an MTL formula φ holds at time point $t \in \mathbb{T}$, i.e., $(\boldsymbol{x}_{\mathbb{T}}, t) \models_{\mathrm{f}} \varphi$, as follows:

$(\boldsymbol{x}_{\mathbb{T}}, t) \models_{\mathrm{f}} p \iff p \in \boldsymbol{x}_{\mathbb{T}}(t)$;
$(\boldsymbol{x}_{\mathbb{T}}, t) \models_{\mathrm{f}} \neg p \iff p \notin \boldsymbol{x}_{\mathbb{T}}(t)$;
$(\boldsymbol{x}_{\mathbb{T}}, t) \models_{\mathrm{f}} \varphi_1 \wedge \varphi_2 \iff (\boldsymbol{x}_{\mathbb{T}}, t) \models_{\mathrm{f}} \varphi_1$ and $(\boldsymbol{x}_{\mathbb{T}}, t) \models_{\mathrm{f}} \varphi_2$;
$(\boldsymbol{x}_{\mathbb{T}}, t) \models_{\mathrm{f}} \varphi_1 \vee \varphi_2 \iff (\boldsymbol{x}_{\mathbb{T}}, t) \models_{\mathrm{f}} \varphi_1$ or $(\boldsymbol{x}_{\mathbb{T}}, t) \models_{\mathrm{f}} \varphi_2$;
$(\boldsymbol{x}_{\mathbb{T}}, t) \models_{\mathrm{f}} \varphi_1 \mathbf{U}_{[a,b]} \varphi_2 \iff$

- $\exists t' \in [t+a, t+b] \cap \mathbb{T}$ s.t. $(\boldsymbol{x}_{\mathbb{T}}, t') \models_{\mathrm{f}} \varphi_2$ and $\forall t'' \in [t, t'], (\boldsymbol{x}_{\mathbb{T}}, t'') \models_{\mathrm{f}} \varphi_1$, or
- $T \leq t+b$ and $\forall t'' \in [t, T), (\boldsymbol{x}_{\mathbb{T}}, t'') \models_{\mathrm{f}} \varphi_1$

$(\boldsymbol{x}_{\mathbb{T}}, t) \models_{\mathrm{f}} \mathbf{F}_{[a,b]} \varphi \iff t+b \geq T$ or $\exists t' \in [t+a, t+b] \cap \mathbb{T}$ s.t. $(\boldsymbol{x}_{\mathbb{T}}, t') \models_{\mathrm{f}} \varphi$;
$(\boldsymbol{x}_{\mathbb{T}}, t) \models_{\mathrm{f}} \mathbf{G}_{[a,b]} \varphi \iff t+a \geq T$ or $\forall t' \in [t+a, t+b] \cap \mathbb{T}, (\boldsymbol{x}_{\mathbb{T}}, t') \models_{\mathrm{f}} \varphi$

We say that $\boldsymbol{x}_{\mathbb{T}}$ satisfies φ if $(\boldsymbol{x}_{\mathbb{T}}, 0) \models_{\mathrm{f}} \varphi$. Also, for ensuring that our semantics complies with Lemma 1, we define $(\boldsymbol{x}_{\mathbb{T}}, t) \models_{\mathrm{f}} \varphi$ for all $t \geq T$ for any φ.

3 The Problem Formulation

Next, we formally introduce the various aspects of the central problem of the paper.

Sample. The input data consists of a set of labeled (piecewise-constant) prefixes. Formally, we rely on a sample $\mathcal{S} = (P, N)$ consisting of a set P of positive prefixes and a set N of negative prefixes such that $P \cap N = \emptyset$. We say an MTL formula φ is *globally-separating* ($\mathbf{G}$-sep, for short) for $\mathcal{S}$ if it satisfies all the positive prefixes at each time point and does not satisfy negative prefixes at some time point[8]. Formally, given a sample $\mathcal{S}$, we define an MTL formula φ to be $\mathbf{G}$-sep for $\mathcal{S}$ if (i) for all $\boldsymbol{x}_{\mathbb{T}} \in P$ and for all $t \in [0, T)$, $(\boldsymbol{x}_{\mathbb{T}}, t) \models_{\mathrm{f}} \varphi$; and (ii) for all $\boldsymbol{y}_{\mathbb{T}} \in N$, there exists $t \in [0, T)$ such that $(\boldsymbol{y}_{\mathbb{T}}, t) \not\models_{\mathrm{f}} \varphi$.

Future-Reach. To formalize the lookahead of an MTL formula φ, we rely on its future-reach $\mathit{fr}(\varphi)$, following [31,29], which indicates how much of the future is required to determine the satisfaction of φ. It is defined inductively as follows:

$$\begin{aligned}
&\mathit{fr}(p) = \mathit{fr}(\neg p) = 0\\
&\mathit{fr}(\varphi_1 \wedge \varphi_2) = \mathit{fr}(\varphi_1 \vee \varphi_2) = \max(\mathit{fr}(\varphi_1), \mathit{fr}(\varphi_2))\\
&\mathit{fr}(\varphi_1 \mathbf{U}_{[a,b]} \varphi_2) = b + \max(\mathit{fr}(\varphi_1), \mathit{fr}(\varphi_2))\\
&\mathit{fr}(\mathbf{F}_{[a,b]} \varphi) = \mathit{fr}(\mathbf{G}_{[a,b]} \varphi) = b + \mathit{fr}(\varphi)
\end{aligned}$$

[7] Following Eisner et al. [24], Ho et al. [29] defined the weak semantics of MTL for the pointwise setting, which we adapt here for the continuous setting.
[8] Most stream-based monitors check if the specification holds at every time point [11].

To highlight that $fr(\varphi)$ quantifies the lookahead of φ, we observe the following lemma:

Lemma 2. *Let φ be an MTL formula such that $fr(\varphi) \leq K$ for some $K \in \mathbb{R}^{\geq 0}$. Let $\boldsymbol{x}$ and $\boldsymbol{y}$ be two signals such that $\boldsymbol{x}_{[0,K]} = \boldsymbol{y}_{[0,K]}$. Then, for all $T \in \mathbb{R}^{\geq 0}$, $\boldsymbol{x}_{\mathbb{T}} \models_{\mathrm{f}} \varphi$ if and only if $\boldsymbol{y}_{\mathbb{T}} \models_{\mathrm{f}} \varphi$.*

Intuitively, the above lemma states that a formula with future-reach $\leq K$ cannot distinguish between two signals that are identical up to time K. The lemma can be proved using structural induction over φ. For space constraints, we include the whole proof in Appendix C.

The Problem. We now formally introduce the problem of synthesizing an MTL formula. In the problem, we ensure that the MTL formula is efficient for monitoring by allowing the system designer to specify a future-reach bound.

*Problem 1 (*SynTL*).* Given a sample $\mathcal{S} = (P, N)$ and a future-reach bound K, find an MTL formula φ such that (i) φ is **G**-sep for $\mathcal{S}$; (ii) $fr(\varphi) \leq K$; (iii) for every MTL formula φ' such that φ' is **G**-sep for $\mathcal{S}$ and $fr(\varphi') \leq K$, $|\varphi| \leq |\varphi'|$.

Intuitively, the above optimization problem asks to synthesize a minimal size MTL formula that is **G**-sep for the input sample and has a future-reach within the input bound. Before we dive into the procedure for finding such an MTL formula, we first study if such an MTL formula even exists.

4 Existence of a solution

As alluded to in the introduction, for any given sample $\mathcal{S}$ and future-reach bound K, the existence of a suitable **G**-separating formula is not always guaranteed. For an illustration, consider the sample $\mathcal{S}$ with one positive prefix $\boldsymbol{x}_{\mathbb{T}} = \langle (0, \{q\}), (2, \{\}) \rangle$ and one negative prefix $\boldsymbol{y}_{\mathbb{T}} = \langle (0, \{q\}) \rangle$, and domain $\mathbb{T} = [0, 4)$. For $\mathcal{S}$, there is no formula φ with $fr(\varphi) \leq 1$ that is **G**-sep. To see this, assume there exists a prospective formula φ. Consequently, φ being **G**-sep, $(\boldsymbol{x}_{\mathbb{T}}, 0) \models \varphi$. Observe that, for all time-points $t \in \mathbb{T}$, $\boldsymbol{y}_{\mathbb{T}}$ when restricted to time interval $[t, t+1] \cap \mathbb{T}$ appears identical to $\boldsymbol{x}_{\mathbb{T}}$ when restricted to time interval $[0, 1]$ to φ since its future-reach is 1 (using Lemma 2). Thus, for all time-points $t \in \mathbb{T}$, $(\boldsymbol{y}_{\mathbb{T}}, t) \models \varphi$ violating that φ is **G**-sep.

What we show now is that one can check whether a prospective formula exists by relying on a simple characterization of the inputs $\mathcal{S}$ and K. Towards this, we introduce introduce some terminology.

We introduce the infix of a prefix $\boldsymbol{x}_{\mathbb{T}}$ that is a restriction of $\boldsymbol{x}_{\mathbb{T}}$ to a specific time interval. Formally, given two time-points $t_1 \leq t_2 < T$ and a prefix $\boldsymbol{x}_{\mathbb{T}}$, infix $\boldsymbol{x}_{\mathbb{T}}^{[t_1,t_2]}$ is the function $\boldsymbol{x}_{\mathbb{T}}^{[t_1,t_2]} \colon [0, t_2 - t_1] \to 2^{\mathcal{P}}$ such that $\boldsymbol{x}_{\mathbb{T}}^{[t_1,t_2]}(t) = \boldsymbol{x}_{\mathbb{T}}(t + t_1)$ for all $t \in [0, t_2 - t_1]$.

Next, we define a characterization of a sample $\mathcal{S}$ based on the future-reach K, which we term as *K-infix-separable*. Intuitively, we say $\mathcal{S}$ to be *K-infix-separable* if there is a K-length infix $\boldsymbol{y}_{\mathbb{T}}^{[t_1,t_2]}$ for every negative prefix $\boldsymbol{y}_{\mathbb{T}}$ in $\mathcal{S}$ that is not an infix of any positive prefix in $\mathcal{S}$. Formally, $\mathcal{S}=(P,N)$ is *K-infix-separable* if for every negative prefix $\boldsymbol{y}_{\mathbb{T}}\in N$, there exists an infix $\boldsymbol{y}_{\mathbb{T}}^{[t_1,t_2]}$ with $t_2-t_1\leq K$ such that $\boldsymbol{y}_{\mathbb{T}}^{[t_1,t_2]}\neq \boldsymbol{x}_{\mathbb{T}}^{[t_1',t_2']}$ for any infix $\boldsymbol{x}_{\mathbb{T}}^{[t_1',t_2']}$ of any positive prefix $\boldsymbol{x}_{\mathbb{T}}\in P$.

We now state the result that enables checking the existence of a solution to Problem 1.

Lemma 3. *For a given sample $\mathcal{S}$ and future-reach bound K, there exists an MTL formula φ with $fr(\varphi)\leq K$ that is* **G**-sep *for $\mathcal{S}$ if and only if $\mathcal{S}$ is K-infix-separable.*

Proof. ($\Rightarrow$) For the forward direction, consider φ be an MTL formula with $fr(\varphi)\leq K$ that is **G**-sep for $\mathcal{S}$. Since φ is **G**-sep, for any arbitrary negative prefix, say $\bar{\boldsymbol{y}}_{\mathbb{T}}$, there must be a time-point, say $\bar{t}<T$, such that $(\bar{\boldsymbol{y}}_{\mathbb{T}},\bar{t})\not\models\varphi$. If $\bar{t}+K<T$, we show by contradiction that the infix $\bar{\boldsymbol{y}}_{\mathbb{T}}^{[\bar{t},\bar{t}+K]}$ is not an infix in any positive prefix. In particular, if $\bar{\boldsymbol{y}}_{\mathbb{T}}^{[\bar{t},\bar{t}+K]}=\boldsymbol{x}_{\mathbb{T}}^{[t,t+K]}$, then $(\boldsymbol{x}_{\mathbb{T}},t)\not\models\varphi$ as φ cannot distinguish between signals that are identical up to time K (using Lemma 2). If $\bar{t}+K\geq T$, the semantics of MTL being weak, there is an $L<K$ with $\bar{t}+L<T$ such that for any $\boldsymbol{y}\in ext(\boldsymbol{y}_{\mathbb{T}}^{[0,\bar{t}+L]})$, $(\boldsymbol{y},t)\not\models\varphi$ (using Lemma 1). Once again, we show by contradiction that the infix $\bar{\boldsymbol{y}}_{\mathbb{T}}^{[\bar{t},\bar{t}+L]}$ is not an infix in any positive prefix. In particular, if $\bar{\boldsymbol{y}}_{\mathbb{T}}^{[\bar{t},\bar{t}+L]}=\boldsymbol{x}_{\mathbb{T}}^{[t,t+L]}$, then for all $\boldsymbol{x}\in ext(\boldsymbol{x}_{\mathbb{T}}^{[0,t+L]})$ $(\boldsymbol{x},t)\not\models\varphi$. Also, for any $\boldsymbol{x}\in ext(\boldsymbol{x}_{\mathbb{T}})$ $(\boldsymbol{x},t)\not\models\varphi$, meaning $(\boldsymbol{x}_{\mathbb{T}},t)\not\models\varphi$ (again, using Lemma 1).

($\Leftarrow$) For the other direction, consider $\mathcal{S}$ to be *K-infix-separable*. Using the definition of *K-infix-separable*, for any arbitrary negative prefix, say $\bar{\boldsymbol{y}}_{\mathbb{T}}$, we have an infix $\bar{\boldsymbol{y}}_{\mathbb{T}}^{[t_1,t_2]}$ with $t_2-t_1\leq K$ that is not an infix in any positive prefix. We construct a formula $\varphi_{\bar{\boldsymbol{y}}_{\mathbb{T}}}$ that explicitly specifies the propositions appearing in each interval of the infix $\bar{\boldsymbol{y}}_{\mathbb{T}}^{[t_1,t_2]}$ using **G** and $\wedge$ operators. Observe that $fr(\varphi_{\bar{\boldsymbol{y}}_{\mathbb{T}}})\leq K$ since $t_2-t_1\leq K$ in $\bar{\boldsymbol{y}}_{\mathbb{T}}^{[t_1,t_2]}$. Now, the formula $\neg\varphi_{\boldsymbol{x}_{\mathbb{T}}}$ holds at all time-points in all positive prefixes, while it does not hold at time-point t_1 in $\bar{\boldsymbol{y}}_{\mathbb{T}}$. We finally construct the prospective formula as $\varphi=\bigwedge_{\boldsymbol{y}_{\mathbb{T}}\in N}\neg\varphi_{\boldsymbol{y}_{\mathbb{T}}}$ which is **G**-sep for $\mathcal{S}$ and also, $fr(\varphi)\leq K$. □

We now describe an NP algorithm to check whether a sample $\mathcal{S}$ is *K-infix-separable*. The crux of the algorithm is to guess, for each negative prefix $\boldsymbol{y}_{\mathbb{T}}$, an infix $\boldsymbol{y}_{\mathbb{T}}^{[t_1,t_2]}$ with $t_2-t_1\leq K$ and then check whether it is an infix of any positive prefix. The procedure of checking involves comparing the various intervals of $\boldsymbol{y}_{\mathbb{T}}^{[t_1,t_2]}$ against the intervals of infixes of positive prefixes.

To describe the checking procedure in detail, let $\bar{\boldsymbol{y}}_{\mathbb{T}}^{[t_1,t_2]}$ be an infix of the negative prefix $\bar{\boldsymbol{y}}_{\mathbb{T}}$. We like to check whether $\bar{\boldsymbol{y}}_{\mathbb{T}}^{[t_1,t_2]}$ is an infix of the positive prefix $\bar{\boldsymbol{x}}_{\mathbb{T}}$. To do so, we check $\bar{\boldsymbol{y}}_{\mathbb{T}}^{[t_1,t_2]}=\bar{\boldsymbol{x}}_{\mathbb{T}}^{[t,t+(t_2-t_1)]}$ with only those infixes in which the time-points where $\boldsymbol{x}_{\mathbb{T}}$ and $\boldsymbol{y}_{\mathbb{T}}$ have been observed coincide. Precisely, we check $\bar{\boldsymbol{y}}_{\mathbb{T}}^{[t_1,t_2]}=\bar{\boldsymbol{x}}_{\mathbb{T}}^{[t,t+(t_2-t_1)]}$ for all those infixes of $\bar{\boldsymbol{x}}_{\mathbb{T}}$ where $t''-t=t'-t_1$, t'' and t' being timepoints where $\boldsymbol{x}_{\mathbb{T}}$ and $\boldsymbol{y}_{\mathbb{T}}$ have been observed, respectively. This process is based on the fact that the changes in an infix occur only at the observation time points. Also, this process takes time polynomial in the number of observation time-points of $\bar{\boldsymbol{x}}_{\mathbb{T}}$ and $\bar{\boldsymbol{y}}_{\mathbb{T}}$. We can perform the procedure for each positive and negative prefix. Overall, we have the following result.

Lemma 4. *Given a sample $\mathcal{S}$ and future-reach bound K, checking whether $\mathcal{S}$ is K-infix-separable can be done in* NP.

5 An SMT-based Algorithm

Our algorithm relies on an SMT-based approach inspired by the numerous constraint satisfaction-based approaches for synthesizing temporal logic formulas [46,15,52,2]. Roughly speaking, our algorithm constructs a series of formulas in Linear Real Arithmetic (LRA) and uses an optimized SMT solver to search for the desired solution. To expand on the specifics of our algorithm, we first familiarize the readers with LRA.

Linear Real Arithmetic (LRA). In LRA [6], given a set of real variables $\mathcal{Y}$, a *term* is defined recursively as either constant $c \in \mathbb{R}$, a real variable $y \in \mathcal{Y}$, a product $c \cdot y$ of a constant $c \in \mathbb{R}$ and a real variable $y \in \mathcal{Y}$, or a sum $t_1 + t_2$ of two terms t_1 and t_2. An *atomic formula* is of the form $t_1 \diamond t_2$ where $\diamond \in \{<, \leq, =, \geq, >\}$. An *LRA formula*, defined recursively, is either an atomic formula, the negation $\neg\Phi$ of an LRA formula Φ, or the disjunction $\Phi_1 \vee \Phi_2$ of two formulas Φ_1, Φ_2. We additionally include standard Boolean constants *true*, and *false* and Boolean operators $\wedge$, $\rightarrow$ and $\leftrightarrow$.

To assign meaning to an LRA formula, we rely on a so-called *interpretation* function $\iota \colon \mathcal{Y} \rightarrow \mathbb{R}$ that maps real variables to constants in $\mathbb{R}$. An interpretation ι can easily be lifted to a term t in the usual way, and is denoted by $\iota(t)$. We now define when ι *satisfies* a formula φ, denoted by $\iota \models \varphi$, recursively as follows: $\iota \models t_1 \diamond t_2$ for $\diamond \in \{<, \leq, =, \geq, >\}$ if and only if $\iota(t_1) \diamond \iota(t_2)$ is *true*, $\iota \models \neg\Phi$ if $\iota \not\models \Phi$, and $\iota \models \Phi_1 \vee \Phi_2$ if and only if $\iota \models \Phi_1$ or $\iota \models \Phi_2$. We say that an LRA formula Φ is *satisfiable* if there exists an interpretation ι with $\iota \models \Phi$.

Despite being NP-complete, with the rise of the SAT/SMT revolution [42], checking the satisfiability of LRA formulas can be handled effectively by several highly-optimized SMT solvers [45,18,5].

Algorithm Overview. Our algorithm constructs a series of LRA formulas $\langle \Phi^n_{\mathcal{S},K} \rangle_{n=1,2,\ldots}$ to facilitate the search for a suitable MTL formula. The formula $\Phi^n_{\mathcal{S},K}$ has the following properties:

1. $\Phi^n_{\mathcal{S},K}$ is satisfiable if and only if there exists an MTL formula φ of size n such that φ is $\mathbf{G}$-sep for $\mathcal{S}$ and $fr(\varphi) \leq K$.
2. from any satisfying interpretation ι of $\Phi^n_{\mathcal{S},K}$, one can construct an appropriate MTL formula φ^ι.

In our algorithm, sketched in Algorithm 1, we first check whether $\mathcal{S}$ is K-*infix-separable* (as described in Section 4) which informs us whether a prospective formula exists. We now check the satisfiability of $\Phi^n_{\mathcal{S},K}$ for increasing values

Algorithm 1 Overview of our algorithm

Input: Sample $\mathcal{S}$, *fr*-bound K
if $\mathcal{S}$ is not *K-infix-separable* **then return** No prospective formula
$n \leftarrow 0$
while True **do**
  $n \leftarrow n + 1$
  Construct $\Phi^n_{\mathcal{S},K} \coloneqq \Phi^{str}_{n,\mathcal{S},K} \wedge \Phi^{fr}_{n,\mathcal{S},K} \wedge \Phi^{sem}_{n,\mathcal{S},K}$
  if $\Phi^n_{\mathcal{S},K}$ is SAT **then**
    Construct φ^ι from a satisfying interpretation ι **return** φ^ι

of size n starting from 1. If $\Phi^n_{\mathcal{S},K}$ is satisfiable for some n, then our algorithm constructs a prospective MTL formula φ^ι from a satisfying interpretation ι returned by the SMT solver. This algorithm terminates because of checking whether a solution exists apriori and it returns a minimal formula because of the iterative search through MTL formulas of increasing sizes.

The crux of our algorithm lies in the construction of the formula $\Phi^n_{\mathcal{S},K}$. Internally, $\Phi^n_{\mathcal{S},K} \coloneqq \Phi^{str}_{n,\mathcal{S},K} \wedge \Phi^{fr}_{n,\mathcal{S},K} \wedge \Phi^{sem}_{n,\mathcal{S},K}$ is a conjunction of three subformulas, each with a distinct role. The subformula $\Phi^{str}_{n,\mathcal{S},K}$ encodes the structure of the prospective MTL formula. The subformula $\Phi^{fr}_{n,\mathcal{S},K}$ ensures that the future-reach of the prospective formula is less than or equal to K. Finally, the subformula $\Phi^{sem}_{n,\mathcal{S},K}$ ensures that the prospective formula is $\mathbf{G}$-sep for $\mathcal{S}$. In what follows, we expand on the construction of each of the introduced subformulas. We drop the subscripts n, $\mathcal{S}$, and K from the subformulas when clear from the context.

Structural Constraints. Following Neider and Gavran [46], we symbolically encode the syntax-DAG of the prospective MTL formula using the formula Φ^{str}. For this, we first fix a naming convention for the nodes of the syntax-DAG of an MTL formula. For a formula of size n, we assign to each of its nodes an identifier from $\{1, \dots, n\}$ such that the identifier of each node is larger than that of its children if it has any. Note that such a naming convention may not be unique. Based on these identifiers, we denote the subformula of φ rooted at Node i as $\varphi[i]$. In that case, $\varphi[n]$ is precisely the formula φ.

Next, to encode a syntax-DAG symbolically, we introduce the following variables[9]: (i) Boolean variables $x_{i,\lambda}$ for $i \in \{1, \dots, n\}$ and $\lambda \in \mathcal{P} \cup \{\neg, \vee, \wedge, \mathbf{U}_I, \mathbf{F}_I, \mathbf{G}_I\}$; (ii) Boolean variables $l_{i,j}$ and $r_{i,j}$ for $i \in \{1, \dots, n\}$ and $j \in \{1, \dots, i\}$; (iii) real variables a_i and b_i for $i \in \{1, \dots, n\}$. The variable $x_{i,\lambda}$ tracks the operator labeled in Node i, meaning, $x_{i,\lambda}$ is set to true if and only if Node i is labeled with λ. The variable $l_{i,j}$ (resp., $r_{i,j}$) tracks the left (resp., right) child of Node i, meaning, $l_{i,j}$ (resp., $r_{i,j}$) is set to true if and only if the left (resp., right) child of Node i is Node j. Finally, the variable a_i (resp., b_i) tracks the lower (resp., upper) bound of the interval I of a temporal operator (i.e., operators $\mathbf{U}_I$, $\mathbf{F}_I$ and $\mathbf{G}_I$), meaning that, if a_i (resp. b_i) is set to $a \in \mathbb{R}$ (resp.,

[9] We include Boolean variables in our LRA formulas since Boolean variables can always be simulated using real variables that are constrained to be either 0 or 1.

$b \in \mathbb{R}$), then the lower (resp., upper) bound of the interval of the operator in Node i is a (resp., b). While we introduce variables a_i and b_i for each node, they become relevant only for the nodes that are labeled with a temporal operator.

We now impose structural constraints on the introduced variables to ensure they encode valid MTL formulas. Exemplarily, we have the following constraint:

$$\Big[\bigwedge_{1\leq i\leq n}\bigvee_{\lambda\in\Lambda} x_{i,\lambda}\Big] \wedge \Big[\bigwedge_{1\leq i\leq n}\bigwedge_{\lambda\neq\lambda'\in\Lambda} \neg x_{i,\lambda} \vee \neg x_{i,\lambda'}\Big],$$

where $\Lambda = \mathcal{P} \cup \{\neg, \vee, \wedge, \mathbf{U}_I, \mathbf{F}_I, \mathbf{G}_I\}$. The above constraint ensures that each node is labeled by exactly one operator or one proposition. We also impose other structural constraints, such as each node can have at most two children, Node 1 must be a proposition, etc. Such structural constraints are similar to the ones proposed by Neider and Gavran [46]. We here additionally ensure that the $\neg$ operator appears only in front of propositions. Also, we ensure that the intervals of the temporal operators are proper using the constraint $\bigwedge_{1\leq i\leq n} 0 \leq a_i \leq b_i < K$. We refer interested readers to Appendix E.1 for all the constraints.

The subformula Φ^{str} is a conjunction of all the structural constraints we described. Using a satisfying interpretation ι of Φ^{str}, one can construct the syntax DAG of a unique MTL formula φ^ι.

Future-reach Constraints. To symbolically compute the future-reach of the prospective formula φ, we encode the inductive definition of the future-reach, as described in Section 3 in an LRA formula. To this end, we introduce real variables f_i for $i \in \{1, \ldots, n\}$ to encode the future-reach of the subformula $\varphi[i]$. Precisely, f_i is set to $f \in \mathbb{R}$ if and only if $fr(\varphi[i]) = f$.

To ensure the desired meaning of the f_i variables, we impose constraints such as

$$\bigwedge_{1\leq i\leq n, 1\leq j<i} [x_{i,\mathbf{F}_I} \wedge l_{i,j}] \rightarrow [f_i = f_j + b_i]. \tag{1}$$

This constraint expresses that if Node i contains the $\mathbf{F}_I$ operator where I is encoded using a_i and b_i, then the future-reach of $\varphi[i]$, i.e., $fr(\varphi[i])$), must be the future-reach of $\varphi[j]$ plus b, i.e., $fr(\varphi[j])+b$. For other operators, we impose similar constraints based on the definition of future-reach for that operator, described in Section 3. We refer the readers to Appendix E.2 for the remaining future-reach constraints.

Finally, to enforce that the future-reach of the prospective MTL formula is within K, along with the constraints mentioned above, we have $f_n \leq K$ in Φ^{fr}.

Semantic Constraints. To symbolically check whether the prospective formula is **G**-sep, we must encode the procedure of checking the satisfaction of an MTL formula into an LRA formula. To this end, we rely on the monitoring procedure devised by Maler and Nickovic [41] for efficiently checking when a signal satisfies an MTL formula. Since our setting is slightly different, we take a brief detour via the description of our adaptation of the monitoring algorithm.

Given an MTL formula φ and a signal prefix $\boldsymbol{x}_{\mathbb{T}}$, our monitoring algorithm computes the (lexicographically) ordered set $\mathcal{I}_{\varphi}(\boldsymbol{x}_{\mathbb{T}}) = \{I_1, \cdots, I_\eta\}$ of *maximal disjoint time intervals* $I_1, \cdots, I_\eta$ where φ holds on $\boldsymbol{x}_{\mathbb{T}}$. Mathematically speaking, the following property holds for the set $\mathcal{I}_{\varphi}(\boldsymbol{x}_{\mathbb{T}})$ we construct:

Lemma 5. *Given an MTL formula φ and a prefix $\boldsymbol{x}_{\mathbb{T}}$, for all $t \in \mathbb{T}$, $(\boldsymbol{x}_{\mathbb{T}}, t) \models_{\mathrm{f}} \varphi$ if and only if $t \in I$ for some $I \in \mathcal{I}_{\varphi}(\boldsymbol{x}_{\mathbb{T}})$.*

In our monitoring algorithm, we compute the set $\mathcal{I}_{\varphi}(\boldsymbol{x}_{\mathbb{T}})$ inductively on the structure of the formula φ. To describe the induction, we use the notation $\mathcal{I}^{\cup}_{\varphi}(\boldsymbol{x}_{\mathbb{T}}) = \bigcup_{I \in \mathcal{I}_{\varphi}(\boldsymbol{x}_{\mathbb{T}})} I$ to denote the union of the intervals in $\mathcal{I}_{\varphi}(\boldsymbol{x}_{\mathbb{T}})$. For the base case, we compute $\mathcal{I}_{p}(\boldsymbol{x}_{\mathbb{T}})$ for every $p \in \mathcal{P}$ by accumulating the time points $t \in [0, T)$ where $(\boldsymbol{x}_{\mathbb{T}}, t) \models_{\mathrm{f}} p$ into maximal disjoint time intervals. In the inductive step, we exploit the relations presented in Table 1 for the different MTL operators. In the table, $[t_1, t_2) \ominus [a, b] = [t_1 - b, t_2 - a) \cap \mathbb{T}$ and $\mathcal{I}^c = \mathbb{T} - \mathcal{I}$. While the table presents the computation of $\mathcal{I}^{\cup}_{\varphi}(\boldsymbol{x}_{\mathbb{T}})$, we can obtain $\mathcal{I}_{\varphi}(\boldsymbol{x}_{\mathbb{T}})$ by

Table 1: The relations for inductive computation of $\mathcal{I}^{\cup}_{\varphi}(\boldsymbol{x}_{\mathbb{T}})$.

$$\mathcal{I}^{\cup}_{\neg p}(\boldsymbol{x}_{\mathbb{T}}) = \big(\mathcal{I}^{\cup}_{p}(\boldsymbol{x}_{\mathbb{T}})\big)^c$$

$$\mathcal{I}^{\cup}_{\varphi_1 \vee \varphi_2}(\boldsymbol{x}_{\mathbb{T}}) = \mathcal{I}^{\cup}_{\varphi_1}(\boldsymbol{x}_{\mathbb{T}}) \cup \mathcal{I}^{\cup}_{\varphi_2}(\boldsymbol{x}_{\mathbb{T}})$$

$$\mathcal{I}^{\cup}_{\varphi_1 \wedge \varphi_2}(\boldsymbol{x}_{\mathbb{T}}) = \mathcal{I}^{\cup}_{\varphi_1}(\boldsymbol{x}_{\mathbb{T}}) \cap \mathcal{I}^{\cup}_{\varphi_2}(\boldsymbol{x}_{\mathbb{T}})$$

$$\mathcal{I}^{\cup}_{\mathbf{F}_{[a,b]}\,\varphi}(\boldsymbol{x}_{\mathbb{T}}) = \big(\textstyle\bigcup_{I \in \mathcal{I}_{\varphi}(\boldsymbol{x}_{\mathbb{T}})} I \ominus [a, b]\big) \cup [T - b, T)$$

$$\mathcal{I}^{\cup}_{\mathbf{G}_{[a,b]}\,\varphi}(\boldsymbol{x}_{\mathbb{T}}) = \big(\textstyle\bigcup_{I \in (\mathcal{I}_{\varphi}(\boldsymbol{x}_{\mathbb{T}}))^c} I \ominus [a, b]\big)^c \cup [T - a, T)$$

$$\mathcal{I}^{\cup}_{\varphi\,\mathbf{U}_{[a,b]}\,\psi}(\boldsymbol{x}_{\mathbb{T}}) = \textstyle\bigcup_{I_\varphi \in \mathcal{I}_{\varphi}(\boldsymbol{x}_{\mathbb{T}})} \bigcup_{I_\psi \in \mathcal{I}_{\psi}(\boldsymbol{x}_{\mathbb{T}})} \Big(\big((I_\varphi \cap I_\psi) \ominus [a, b]\big) \cap I_\varphi\Big) \cup I_{\mathbb{T}},$$

$$\text{where } I_{\mathbb{T}} = \begin{cases} [\max(T - b, t), T), & \text{if } \exists t \text{ s.t. } [t, T) \in \mathcal{I}_{\varphi}(\boldsymbol{x}_{\mathbb{T}}) \\ \emptyset, & \text{otherwise} \end{cases}$$

simply partitioning $\mathcal{I}^{\cup}_{\varphi}(\boldsymbol{x}_{\mathbb{T}})$ into maximal disjoint intervals.

For an illustration, we consider the example from the introduction and compute $\mathcal{I}_{\varphi_2}(u_1)$ where u_1 is the first positive prefix, $\varphi_2 = p \vee \mathbf{F}_{[0,1]}\, q$, and $\mathbb{T} = [0, 6)$. First, we have $\mathcal{I}_{p}(u_1) = \{[0, 2), [3, 6)\}$ and $\mathcal{I}_{q}(u_1) = \{[0, 1), [2, 4)\}$. Now, we can compute $\mathcal{I}_{\mathbf{F}_{[0,1]}\, q}(u_1) = \{[0, 4), [5, 6)\}$ and then $\mathcal{I}_{p \vee \mathbf{F}_{[0,1]}\, q}(u_1) = \{[0, 6)\}$.

In the monitoring algorithm, the number of maximal intervals required in $\mathcal{I}_{\varphi}(\boldsymbol{x}_{\mathbb{T}})$ is upper-bounded by $\mathcal{M} = \mu|\varphi|$, where $\mu = \max(\{|\mathcal{I}_{p}(\boldsymbol{x}_{\mathbb{T}})| \mid p \in \mathcal{P}\})$, as also observed by Maler and Nickovic [41]. The computation of this bound can also be done inductively on the structure of φ.

Now, in the subformula Φ^{sem}, we symbolically encode the set $\mathcal{I}_{\varphi}(\boldsymbol{x}_{\mathbb{T}})$ of our prospective MTL formula φ. To this end, we introduce variables $t^{l}_{i,m,s}$ and $t^{r}_{i,m,s}$ where $i \in \{1, \ldots, n\}$, $m \in \{1, \ldots, \mathcal{M}\}$, and $s \in \{1, \ldots, |\mathcal{S}|\}$, s being an identifier for the s^{th} prefix $\boldsymbol{x}^{s}_{\mathbb{T}}$ in $\mathcal{S}$. The variables $t^{l}_{i,m,s}$ and $t^{r}_{i,m,s}$ encode the m^{th} interval of $\mathcal{I}_{\varphi[i]}(\boldsymbol{x}^{s}_{\mathbb{T}})$ for the subformula $\varphi[i]$. In other words, $t^{l}_{i,m,s} = t_1$ and $t^{r}_{i,m,s} = t_2$ if and only if $[t_1, t_2)$ is the m^{th} interval of $\mathcal{I}_{\varphi[i]}(\boldsymbol{x}^{s}_{\mathbb{T}})$.

Now, to ensure that the variables $t^l_{i,m,s}$ and $t^r_{i,m,s}$ have their desired meaning, we introduce constraints for each operator based on the relations defined in Table 1. We now present these constraints for the different MTL operators.

For the $\neg$ operator, we have the following constraints:

$$\bigwedge_{\substack{1\leq i\leq n\\ 1\leq j<i}} x_{i,\neg} \land l_{i,j} \rightarrow \big[\bigwedge_{1\leq s\leq |\mathcal{S}|} comp_s(i,j)\big],$$

where, for every $\boldsymbol{x}^s_{\mathbb{T}}$ in $\mathcal{S}$, $comp_s(i,j)$ encodes that $\mathcal{I}^{\cup}_{\varphi[i]}(\boldsymbol{x}^s_{\mathbb{T}})$ is the complement of $\mathcal{I}^{\cup}_{\varphi[j]}(\boldsymbol{x}^s_{\mathbb{T}})$. We construct $comp_s(i,j)$ as follows:

$$\texttt{ite}(t^l_{j,1,s} = 0, \tag{2}$$

$$\bigwedge_{1\leq m\leq \mathcal{M}-1} t^l_{i,m,s} = t^r_{j,m,s} \land t^r_{i,m,s} = t^l_{j,m+1,s}, \tag{3}$$

$$t^l_{i,1,s} = 0 \land t^r_{i,1,s} = t^l_{j,1,s} \land \tag{4}$$

$$\bigwedge_{1\leq m\leq \mathcal{M}-1} t^l_{i,m+1,s} = t^r_{j,m,s} \land t^r_{i,m+1,s} = t^l_{j,m+1,s}),$$

where `ite` is a syntactic sugar for the "if-then-else" construct over LRA formulas, which is standard in many SMT solvers. Here, Condition 2 checks whether the left bound of the first interval of $\mathcal{I}_{\varphi[j]}(\boldsymbol{x}^s_{\mathbb{T}})$, encoded by $t^l_{j,1,s}$, is 0. If that holds, as specified by Constraint 3, the left bound of the first interval of $\mathcal{I}_{\varphi[i]}(\boldsymbol{x}^s_{\mathbb{T}})$, encoded by $t^l_{1,i,s}$, will be the right bound of the first interval of $\mathcal{I}_{\varphi[j]}(\boldsymbol{x}^s_{\mathbb{T}})$, encoded $t^r_{1,j,s}$ and so on. If Condition 2 does not hold, as specified by Constraint 4, the left bound of the first interval of $\mathcal{I}_{\varphi[i]}(\boldsymbol{x}^s_{\mathbb{T}})$ will start with 0, and so on.

As an example, for a prefix $\boldsymbol{x}^s_{\mathbb{T}}$ and $\mathbb{T} = [0,7)$, let $\mathcal{I}_{\varphi[j]}(\boldsymbol{x}^s_{\mathbb{T}}) = \{[0,4),[6,7)\}$. Then, Constraint 3 ensures that $\mathcal{I}_{\varphi[i]}(\boldsymbol{x}^s_{\mathbb{T}}) = \{[4,6)\}$[10]. Conversely, if $\mathcal{I}_{\varphi[j]}(\boldsymbol{x}^s_{\mathbb{T}}) = \{[1,4),[6,7)\}$, then Constraints 4 ensures that $\mathcal{I}_{\varphi[i]}(\boldsymbol{x}^s_{\mathbb{T}}) = \{[0,1),[4,6)\}$.

For the $\lor$ operator, we have the following constraint:

$$\bigwedge_{\substack{1\leq i\leq n\\ 1\leq j,j'<i}} x_{i,\lor} \land l_{i,j} \land r_{i,j'} \rightarrow \big[\bigwedge_{1\leq s\leq |\mathcal{S}|} union_s(i,j,j')\big],$$

where, for every $\boldsymbol{x}^s_{\mathbb{T}}$ in $\mathcal{S}$, $union_s(i,j,j')$ encodes that $\mathcal{I}_{\varphi[i]}(\boldsymbol{x}^s_{\mathbb{T}})$ consists of the maximal disjoint intervals obtained from the union of the intervals in $\mathcal{I}_{\varphi[j]}(\boldsymbol{x}^s_{\mathbb{T}})$

[10] $|\mathcal{I}_{\varphi[i]}(\boldsymbol{x}^s_{\mathbb{T}})|$ may differ for different subformulas $\varphi[i]$; we address this at the end of this section.

and $\mathcal{I}_{\varphi[j']}(\boldsymbol{x}^s_{\mathbb{T}})$. We construct $union_s(i,j,j')$ as follows:

$$\bigwedge_{\sigma\in[l,r]}\bigwedge_{1\leq m\leq\mathcal{M}}\left(\bigvee_{1\leq m'\leq\mathcal{M}}(t^\sigma_{i,m,s}=t^\sigma_{j,m',s})\vee\bigvee_{1\leq m'\leq\mathcal{M}}(t^\sigma_{i,m,s}=t^\sigma_{j',m',s})\right)\wedge \tag{5}$$

$$\bigwedge_{\sigma\in[l,r]}\bigwedge_{1\leq m\leq\mathcal{M}}\left(\bigvee_{1\leq m'\leq\mathcal{M}}(t^\sigma_{i,m,s}=t^\sigma_{j,m',s})\iff\bigwedge_{1\leq m''\leq\mathcal{M}}(t^\sigma_{j,m',s}\notin I_{j',m'',s})\right)\wedge \tag{6}$$

$$\bigwedge_{\sigma\in[l,r]}\bigwedge_{1\leq m\leq\mathcal{M}}\left(\bigvee_{1\leq m'\leq\mathcal{M}}(t^\sigma_{i,m,s}=t^\sigma_{j',m',s})\iff\bigwedge_{1\leq m''\leq\mathcal{M}}(t^\sigma_{j',m',s}\notin I_{j,m'',s})\right), \tag{7}$$

where $I_{k,m,s}$ denotes the interval encoded by bounds $t^l_{k,m,s}$ and $t^r_{k,m,s}$ [11]. Here, Constraint 5 states that the left (resp., right) bound of each interval of $\mathcal{I}_{\varphi[i]}(\boldsymbol{x}^s_{\mathbb{T}})$, encoded by $t^l_{i,m,s}$ (resp., $t^r_{i,m,s}$) corresponds to one of the left (resp., right) bounds of the intervals in $\mathcal{I}_{\varphi[j]}(\boldsymbol{x}^s_{\mathbb{T}})$ or in $\mathcal{I}_{\varphi[j']}(\boldsymbol{x}^s_{\mathbb{T}})$. Then, Constraint 6 states that for each interval I in $\mathcal{I}_{\varphi[j]}(\boldsymbol{x}^s_{\mathbb{T}})$, the left (resp., right) bound of I should appear as the left (resp., right) bound of some interval in $\mathcal{I}_{\varphi[i]}(\boldsymbol{x}^s_{\mathbb{T}})$ if and only if the left (resp., right) bound of I is not included in any of the intervals in $\mathcal{I}_{\varphi[j']}(\boldsymbol{x}^s_{\mathbb{T}})$. Constraint 7 mimics the statement made by Constraint 6 but for the bounds of the intervals in $\mathcal{I}_{\varphi[j']}(\boldsymbol{x}^s_{\mathbb{T}})$.

For an illustration, assume that $\mathcal{I}_{\varphi[j]}(\boldsymbol{x}^s_{\mathbb{T}})=\{[1,4),[6,7)\}$ and $\mathcal{I}_{\varphi[j']}(\boldsymbol{x}^s_{\mathbb{T}})=\{[3,5),[6,7)\}$ for a prefix $\boldsymbol{x}^s_{\mathbb{T}}$ and $T=7$. Now, if $\varphi[i]=\varphi[j]\vee\varphi[j']$, then $\mathcal{I}_{\varphi[i]}(\boldsymbol{x}^s_{\mathbb{T}})=\{[1,5),[6,7)\}$ based on the relation for $\vee$-operator in Table 1. Observe that all the bounds of the intervals in $\mathcal{I}_{\varphi[i]}(\boldsymbol{x}^s_{\mathbb{T}})$, i.e., 1, 5, 6, and 7, are present as the bounds of the intervals in either $\mathcal{I}_{\varphi[j]}(\boldsymbol{x}^s_{\mathbb{T}})$ or $\mathcal{I}_{\varphi[j']}(\boldsymbol{x}^s_{\mathbb{T}})$. This fact is in accordance with Constraint 5. Also, the right bound of $[1,4)$ in $\mathcal{I}_{\varphi[j]}(\boldsymbol{x}^s_{\mathbb{T}})$ does not appear as a bound of any intervals in $\mathcal{I}_{\varphi[i]}(\boldsymbol{x}^s_{\mathbb{T}})$, as it is included in an interval in $\mathcal{I}_{\varphi[j']}(\boldsymbol{x}^s_{\mathbb{T}})$, i.e., $4\in[3,5)$. This is in accordance with Constraint 6.

Next, for the $\mathbf{F}_I$-operator where I is encoded using a_i and b_i, we have the following constraint:

$$\bigwedge_{\substack{1\leq i\leq n\\1\leq j<i}} x_{i,\mathbf{F}_I}\wedge l_{i,j}\rightarrow\Big[\bigwedge_{1\leq s\leq|\mathcal{S}|} union'_s(i,k,k)\wedge\ominus_s^{[a_i,b_i]}(k,j)\Big].$$

based on the relation for the $\mathbf{F}_{[a,b]}$ operator in Table 1. We here rely on an intermediate set of intervals $\tilde{\mathcal{I}}_k$ encoded using some auxiliary variables $\tilde{t}^l_{k,m,s}$ and $\tilde{t}^r_{k,m,s}$ where $m\in\{1,\ldots,\mathcal{M}\}$ and $s\in\{1,\ldots,|\mathcal{S}|\}$. Also, we use the formula $\ominus_s^{[a_i,b_i]}(k,j)$ to encode that the intervals in $\tilde{\mathcal{I}}_k$ can be obtained by performing

[11] In LRA, $t\notin[t_1,t_2)$ can be encoded as $t<t_1\vee t\geq t_2$.

$I \ominus [a, b]$ to each interval I in $\mathcal{I}_{\varphi[j]}(\boldsymbol{x}^s_{\mathbb{T}})$, where $a_i = a$ and $b_i = b$. Finally, the formula $union'(i, k, k)$ encodes that $\mathcal{I}_{\varphi[i]}(\boldsymbol{x}^s_{\mathbb{T}})$ consists of the maximal disjoint intervals obtained from the union of the intervals in $\tilde{\mathcal{I}}_k$ and $\{[T - b, T)\}$.

The construction of $union'(i, k, k)$ is similar to that of $union(i, j, j')$ in that the constraints involved are similar to Constraints 5 to 7. For $\ominus_s^{[a_i,b_i]}(k, j)$, we have the following constraint:

$$\bigwedge_{1 \le m \le \mathcal{M}-1} \left[\tilde{t}^l_{k,m,s} = \max\{0, \left(t^l_{j,m,s} - b_i\right)\} \ \wedge \tilde{t}^r_{k,m,s} = \max\{0, \left(t^r_{j,m,s} - a_i\right)\}\right] \quad (8)$$

As an example, consider $\mathcal{I}_{\varphi[j]}(\boldsymbol{x}^s_{\mathbb{T}}) = \{[1, 4), [6, 7)\}$ for a prefix $\boldsymbol{x}^s_{\mathbb{T}}$ and $T = 7$. Now, if $\varphi[i] = \mathbf{F}_{[1,4]}\, \varphi[j]$, then first we have $\tilde{\mathcal{I}}_k = \{[0, 3), [2, 6)\}$ based on Constraint 8 [12]. Next, we have $\mathcal{I}_{\varphi[i]}(\boldsymbol{x}^s_{\mathbb{T}}) = \{[0, 7)\}$ which consists of the maximal disjoint intervals from $\tilde{\mathcal{I}}_k \cup \{[T-4, T)\} = \{[0, 3), [2, 6), [3, 7)\}$ using $union'(i, k, k)$.

For the $\mathbf{U}_I$ operator, we have the following constraint:

$$\bigwedge_{\substack{1 \le i \le n \\ 1 \le j, j' < i}} x_{i,\mathbf{U}_I} \wedge l_{i,j} \wedge r_{i,j'} \rightarrow \Big[\bigwedge_{1 \le s \le |\mathcal{S}|} intersection_s(k_1, j, j') \wedge \ominus_s^{[a_i,b_i]}(k_2, k_1) \wedge cond{-}int_s(k_3, k_2, j) \wedge union_s(i, k_3, k_3)\Big]$$

Here, we introduce three intermediate set of intervals $\tilde{\mathcal{I}}_{k_1}$, $\tilde{\mathcal{I}}_{k_2}$ and $\tilde{\mathcal{I}}_{k_3}$ encoded using auxiliary variables $\tilde{t}^l_{k_i,m,s}$ and $\tilde{t}^r_{k_i,m,s}$ where $i \in \{1, 2, 3\}$, $m \in \{1, \dots, \mathcal{M}\}$ and $s \in \{1, \dots, |\mathcal{S}|\}$. Similar to the constraints for the $\vee$ operator, we denote an interval in $\tilde{\mathcal{I}}_{k_i}$ as $I_{k_i,m,s}$ where, $I_{k_i,m,s} = [\tilde{t}^l_{k_i,m,s}, \tilde{t}^r_{k_i,m,s})$. Now, $intersection_s(k_1, j, j')$ encodes that $\tilde{\mathcal{I}}_{k_1}$ consists of the maximal disjoint intervals obtained from the intersection of the intervals in $\mathcal{I}_{\varphi[j]}(\boldsymbol{x}^s_{\mathbb{T}})$ and $\mathcal{I}_{\varphi[j']}(\boldsymbol{x}^s_{\mathbb{T}})$. Note that the intersection can be achieved using the $union_s$ and the $comp_s$ operators using De Morgan's law, i.e., $A \cap B = (A^c \cup B^c)^c$. Then, $\ominus_s^{[a_i,b_i]}(k_2, k_1)$ denotes that the intervals in $\tilde{\mathcal{I}}(k_2)$ can be obtained by performing $I \ominus [a, b]$ to each interval in $\tilde{\mathcal{I}}_{k_1}$ using constraint 8. Next, the operator $cond - int_s(k_3, k_2, j)$ denotes that the m^{th} interval in $\tilde{\mathcal{I}}(k_3)$ $(I_{k_3,m,s})$ is obtained by taking the intersection of the m^{th} interval in $\tilde{\mathcal{I}}(k_2)$ $(I_{k_2,m,s})$ and the m'^{th} interval in $\mathcal{I}_{\varphi[j]}(\boldsymbol{x}^s_{\mathbb{T}})$ $(I_{j,m',s})$ such that, $I_{k_1,m,s}$ $(I_{k_2,m,s} = I_{k_1,m,s} \ominus [a, b]$, by construction) is a subset of $I_{j,m',s}$. This can be achieved by encoding $cond - int_s(k_3, k_2, j)$ as the following constraint:

$$\bigwedge_{1 \le m \le \mathcal{M}} \bigwedge_{1 \le m' \le \mathcal{M}} (I_{k_1,m,s} \subseteq I_{j,m',s}) \rightarrow I_{k_3,m,s} = I_{k_2,m,s} \cap I_{j,m',s}$$

Note that the subset check and the intersection of two intervals both allow simple encodings in LRA. Finally, the formula $union_s(i, k_3, k_3)$ encodes that

[12] While the intervals in $\tilde{\mathcal{I}}_k$ may not be disjoint, $union'(i, k, k)$ ensures that $\mathcal{I}_{\varphi[i]}(\boldsymbol{x}^s_{\mathbb{T}})$ consists of only maximal disjoint intervals.

$\mathcal{I}_{\varphi[i]}(\boldsymbol{x}^s_{\mathbb{T}})$ consists of the maximal disjoint intervals obtained from the union of the intervals in $\tilde{\mathcal{I}}_{k_3}$.

For an illustration, assume that $\mathcal{I}_{\varphi[j]}(\boldsymbol{x}^s_{\mathbb{T}}) = \{[1,3),[5,8)\}$ and $\mathcal{I}_{\varphi[j']}(\boldsymbol{x}^s_{\mathbb{T}}) = \{[4,6),[7,9)\}$ for a prefix $\boldsymbol{x}^s_{\mathbb{T}}$ and $T=9$. Now, let $\varphi[i] = \varphi[j]\,\mathbf{U}_{[0,3]}\,\varphi[j']$. Then, $\mathcal{I}_{\varphi[i]}(\boldsymbol{x}^s_{\mathbb{T}}) = \{[5,8)\}$ using the computation in Table 1.

Note that, following the constraint, $\tilde{\mathcal{I}}_{k_1} = \{[5,6),[7,8)\}$ after taking the intersection of $\mathcal{I}_{\varphi[j]}(\boldsymbol{x}^s_{\mathbb{T}})$ and $\mathcal{I}_{\varphi[j']}(\boldsymbol{x}^s_{\mathbb{T}})$. Then, the Minkowski minus results into the set of intervals $\tilde{\mathcal{I}}_{k_2} = \{[2,6),[4,8)\}$ with $a=0$ and $b=3$. The conditional intersection of $\tilde{\mathcal{I}}_{k_2}$ and $\mathcal{I}_{\varphi[j}(\boldsymbol{x}^s_{\mathbb{T}})$ produces the set of intervals $\tilde{\mathcal{I}}_{k_3} = \{[5,6),[5,8)\}$. Note that this is because both the intervals in $\tilde{\mathcal{I}}_{k_1}$ are subsets of the interval $[5,8)$ in $\mathcal{I}_{\varphi[j]}(\boldsymbol{x}^s_{\mathbb{T}})$ and not of $[1,3)$ and we intersect the intervals in $\tilde{\mathcal{I}}_{k_2}$ with only $[5,8)$. Finally the operator $union_s$ on $\tilde{\mathcal{I}}_{k_3}$ results in $\mathcal{I}_{\varphi[i]}(\boldsymbol{x}^s_{\mathbb{T}})$ to be $\{[5,8)\}$ that complies with the actual semantics of the $\mathbf{U}_I$ operator. It can be also checked that taking a normal intersection instead of the conditional one would have wrongly resulted in $\mathcal{I}_{[}(\boldsymbol{x}^s_{\mathbb{T}})\varphi[i]]$ to be $\{[2,3),[5,8)\}$ that depicts the intricacy in computing the satisfaction intervals for $\mathbf{U}_I$ as shown in Figure 3(a) in [41].

For the $\mathbf{G}_I$ and the $\wedge$ operator, we encode the relations described in Table 1 by reusing the constraints from the formulas $comp_s(i,j)$, $union_s(i,j,j')$, and $\ominus_s^{[a_i,b_i]}(k,j)$. We present the exact constraints in Appendix E.3. We now assert the correctness of the formulas encoding the set operations as follows:

Lemma 6. *The formulas $comp_s(i,j)$, $union_s(i,j,j')$, $\ominus_s^{[a_i,b_i]}(k,j)$, $intersection_s(i,j,j')$ and $cond{-}int_s(i,j,j')$ correctly encode the complement, union, $\ominus$, intersection and conditional intersection operations on a set of intervals, resp.*

The proof of the lemma is presented in Appendix F.

It is worth noting that although the number of intervals in $\mathcal{I}_{\varphi[i]}(\boldsymbol{x}^s_{\mathbb{T}})$ for each subformula $\varphi[i]$ is bounded by $\mathcal{M}$, it may not contain the same number of intervals. For instance, $\mathcal{I}_p(\boldsymbol{x}^s_{\mathbb{T}}) = \{[0,1),[6,7)\}$ has two intervals, while, assuming $T=7$, $\mathcal{I}_{\neg p}(\boldsymbol{x}^s_{\mathbb{T}}) = \{[1,6)\}$ has only one interval.

To circumvent this, we introduce some variables $num_{i,s}$ for $i \in \{1,\dots,n\}$ and $s \in \{1,\dots,|\mathcal{S}|\}$ to track of the number of intervals in $\mathcal{I}_{\varphi[i]}(\boldsymbol{x}^s_{\mathbb{T}})$ for each subformula $\varphi[i]$ for each prefix $\boldsymbol{x}^s_{\mathbb{T}}$. We now impose $\bigwedge_{1\le i\le n, 1\le m\le\mathcal{M}}[m > num_{i,s}] \rightarrow [t^l_{i,m,s} = T \wedge t^r_{i,m,s} = T]$. This ensures that all the unused variables $t^\sigma_{i,m,s}$ for each Node i and prefix $\boldsymbol{x}^s_{\mathbb{T}}$ in $\mathcal{S}$ are all set to T. We also use the $num_{i,s}$ variables in the constraints for easier computation of $\mathcal{I}_{\varphi[i]}(\boldsymbol{x}^s_{\mathbb{T}})$ for each operator. We include this in our implementation but omit it here for a simpler presentation.

Finally, to ensure that the prospective formula φ is $\mathbf{G}$-sep for $\mathcal{S}$, we add:

$$\bigwedge_{\boldsymbol{x}^s_{\mathbb{T}}\in P}\big[(t^l_{n,1,s}=0)\wedge(t^r_{n,1,s}=T)\big]\wedge\bigwedge_{\boldsymbol{x}^s_{\mathbb{T}}\in N}\big[(t^l_{n,1,s}\neq 0)\vee(t^r_{n,1,s}\neq T)\big].$$

This constraint says that $\mathcal{I}_{\varphi[n]}(\boldsymbol{x}^s_{\mathbb{T}}) = \{[0,T)\}$ for all the positive prefixes $\boldsymbol{x}^s_{\mathbb{T}}$, while $\mathcal{I}_{\varphi[n]}(\boldsymbol{x}^s_{\mathbb{T}}) \neq \{[0,T)\}$ for any negative prefixes $\boldsymbol{x}^s_{\mathbb{T}}$.

The correctness of our algorithm follows from the correctness of the inductive computation of $\mathcal{I}_{\varphi}(\boldsymbol{x}_{\mathbb{T}})$ in Lemma 5 and its encoding using the formulas described in Lemma 6. We state the correctness result formally as follows:

Theorem 1 (Correctness). *Given a sample $\mathcal{S}$ and a future-reach bound K, Algorithm 1 terminates and outputs a minimal MTL formula φ such that φ is globally separating for $\mathcal{S}$ and $fr(\varphi) \leq K$, if such a formula exists.*

Proof. The termination of Algorithm 1 is guaranteed by the decision procedure of checking whether $\mathcal{S}$ is K-*infix-separable* (Section 4). The minimality of the synthesized formula is due to the iterative search of formulas of increasing size and the correct encoding of $\Phi^n_{\mathcal{S},K}$. The correctness of $\Phi^n_{\mathcal{S},K}$ follows from the correctness of the encoding of set operations described in Lemma 6 and the correctness of computation of the sets $\mathcal{I}_{\varphi}(\boldsymbol{x}_{\mathbb{T}})$ using Lemma 5. □

Our synthesis algorithm solves the optimization problem SynTL by constructing formulas in LRA. We now analyze the computational hardness of SynTL and, thus, consider its corresponding decision problem SynTL$_d$: given a sample $\mathcal{S}$, a future-reach bound K and size bound B (in unary), does there exist an MTL formula φ such that φ is **G**-sep for $\mathcal{S}$, $fr(\varphi) \leq K$, and $|\varphi| \leq B$. Following our algorithm, we can encode the SynTL$_d$ problem in an LRA formula $\Phi = \bigvee_{n \leq B} \Phi^n_{\mathcal{S},K}$, where $\Phi^n_{\mathcal{S},K}$ is as described in Algorithm 1. One can check that the size of Φ is $\mathcal{O}(|\mathcal{S}||K|B^3\mathcal{M}^3)$. Now, the fact that the satisfiability of an LRA formula is NP-complete [19] proves the following:

Theorem 2. SynTL$_d$ *is in* NP.

Remark 1. While the exact complexity lower bound for SynTL$_d$ is unknown, we conjecture that SynTL$_d$ is NP-hard. Our hypothesis stems from the fact that one can show SynTL$_d$ is NP-hard for a simple fragment MTL($\mathbf{G}_I, \vee, \neg$), which consists of only $\mathbf{G}_I, \vee$ and $\neg$ operators, following the techniques of Fijalkow and Lagarde [25] (see Appendix G). Note that the hardness result does not directly extend to full MTL: the complexity might be either lower or higher since the logic is a priori more expressive. We leave the hardness result for full MTL as an open problem.

6 Experiments

In this section, we answer the following research questions to assess the performance of our algorithm for synthesizing MTL formulas.

RQ1: Can our algorithm synthesize concise formulas with small future-reach?
RQ2: How does lowering the future-reach bound affect the size of the formulas?
RQ3: How does our algorithm scale for different sample sizes?

To answer the research questions above, we have implemented a prototype of our algorithm in Python 3 using Z3 [44] as the SMT solver in a tool named

TEAL (synThesizing Efficiently monitorAble mtL). To our knowledge, TEAL is the only tool for synthesizing minimal MTL formulas for monitoring purposes (see related works). In TEAL, we implement a heuristic on top of Algorithm 1. We initially set the maximum number of intervals $\mathcal{M}$ in sets $\mathcal{I}_{\varphi[i]}(\boldsymbol{x}_{\mathbb{T}})$ to be $\mu+2$ where $\mu = \max(\{|\mathcal{I}_p(\boldsymbol{x}_{\mathbb{T}})| \mid p \in \mathcal{P}\})$. We iteratively increase the value of $\mathcal{M}$ until we find a solution. To ensure that the synthesized MTL formulas are correct, we implement a verifier based on the inductive computation of $\mathcal{I}_{\varphi}(\boldsymbol{x}_{\mathbb{T}})$ mentioned in Table 1. The heuristic improves the runtime of TEAL significantly since most **G**-sep formulas φ never require the worst-case upper bound[13] of $\mathcal{M} = \mu|\varphi|$.

As typically done in the literature of synthesizing formulas [46,2,50], we evaluate TEAL on benchmarks generated synthetically from MTL formulas. To obtain useful MTL formulas, we identify a number of MTL patterns, listed in Table 2, that are commonly used for monitoring cyber-physical systems. For instance, the time-sensitive requirement of an electronically controlled steering (ECS) system "operational checks like RAM verification must be done every 20 secs" can be monitored globally using the bounded recurrence formula $\mathbf{F}_{[0,20]}$ `operational_check` [36]; the requirement of an autonomous vehicle (from the introductory example) "brake should be triggered until within 2 secs the vehicle has no obstacle in an unsafe distance " can be monitored globally using the bounded until formula `brake` $\mathbf{U}_{[0,2]}$ `no_obstacle`.

Table 2: Typical MTL patterns used for monitoring cyber-physical systems

Bounded Recurrence:	$Globally(\mathbf{F}_{[t_1,t_2]}\, p)$
Bounded Response:	$Globally(p \rightarrow \mathbf{F}_{[t_1,t_2]}\, q)$
Bounded Invariance:	$Globally(p \rightarrow \mathbf{G}_{[t_1,t_2]}\, q)$
Bounded Until:	$Globally(p\, \mathbf{U}_{[t_1,t_2]}\, q)$

In our experiments, we construct MTL formulas from the patterns in Table 2 by replacing time interval $[t_1, t_2]$ with different values. Now, to generate a sample from an MTL formula φ, we generated a set of random prefixes and then classified them into positive or negative depending on whether φ holds at all time-points of the prefix or not. We conducted all the experiments on a single core of a AMD EPYC 7702 64-Core CPU (at 2GHz) using up to 10GB of RAM. The timeout was set to be 5400 secs for all the experiments.

To address **RQ1**, we ran TEAL on a benchmark suite generated from nine MTL formulas obtained from the three MTL patterns in Table 2 by replacing t_1 with 0 and t_2 with 1,2, and 3. The suite consists of 36 samples for each pattern (12 samples for each formula), with the number of prefixes ranging from 10 to 40 and the length of prefixes (i.e., the number of sampled time points) ranging

[13] The operators $\mathbf{F}_I$, $\mathbf{G}_I$, $\wedge$, and $\neg$ increase the number of required intervals by at most one. Only the $\vee$ operator can double it in the worst-case.

from 4 to 6. For each sample $\mathcal{S}$, we set the future-reach bound K to be $fr(\varphi)$, where φ is the formula from which $\mathcal{S}$ was generated.

Table 3: Summary of the synthesized formulas.

Formula pattern	Successful runs		Timed out	Avg Size	Avg Time
	Matched	Not Matched			(in sec)
Bounded Recurrence	36	0	0	2	17.5
Bounded Response	25	5	6	3.7	1860.3
Bounded Invariance	15	7	14	3.6	1397.2
Bounded Until	32	4	0	2.9	362.4

We depict the summary of the results for this experiment in Table 3. For each run, we noted the formula synthesized, its size and the total time taken. Further, we noted whether the synthesized formula matched the pattern of the original formula using which the sample was generated. We observed that the synthesized formulas matched the pattern of the original formula in 87.1% of the cases in which `TEAL` did not time out. This shows that the randomly generated samples captured the behaviour of the original formula rather well, enabling a fair evaluation of `TEAL`.

Furthermore, we observed that the size of the synthesized formula is always equal to or less than that of the original formula, demonstrating that `TEAL` always finds a concise formula for a given future-reach bound Thus, we answer RQ1 in positive.

To address **RQ2**, we investigate how the size of the synthesized formula changed over varying future-reach bounds. For this, we ran `TEAL` on the same benchmark suite from RQ1 but, this time, by varying the future-reach bound K from 1 to 4. We investigate the average size of the minimal formula we get over the generated 108 samples for each future-reach bound.

We observed that for future-reach bounds K of 1, 2, 3, and 4, the average size of the synthesized minimal formulas were 3.904, 3.734, 3.370, and 3.361, respectively. Thus, the trend is that with an increase in K, the average size of the minimal formula decreased. This is because an increase in K allows a bigger search space of formulas. One can, however, also notice that the decrease in the average size of the formulas with increasing future-reach bound is not vast. This highlights the advantage of using a future-reach bound for synthesizing formulas for online monitoring and confirms the efficacy of our algorithm.

To address **RQ3**, we ran `TEAL` on a benchmark suite generated from MTL formulas which originate from the MTL patterns in Table 2, setting $t_1 = 0$ and $t_1 = 2$. The suite consists of 36 samples for each formula, with the number of prefixes varying from 10 to 60 and the length of prefixes varying from 4 to 14. We set the future-reach bound K to be two.

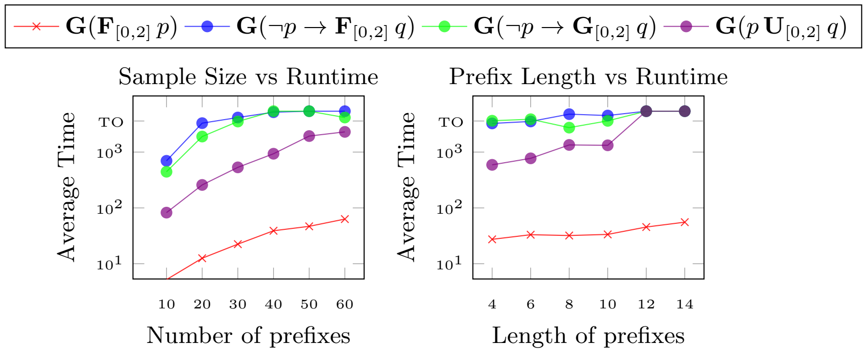


Fig. 2: Runtime change with respect to the number of prefixes and prefix lengths

Figure 2 illustrates the runtime variation of `TEAL` in two cases: increasing the number of prefixes fixing the length of them and increasing the length of prefixes fixing the number of them. We observe that to synthesis a larger formula the time required grows significantly. This trend can be noticed in both the figures.

7 Discussion and Conclusion

We have presented a novel SMT-based algorithm for automatically synthesizing MTL specifications from finite system executions. To be useful for efficient monitoring, we ensure that the synthesized formulas are both concise and have low future-reach. We have shown that our algorithm can synthesize concise formulas from benchmarks generated from commonly used MTL patterns.

While our algorithm is tailored to synthesize globally separating formulas particularly useful for monitoring, we can adapt our algorithm easily to synthesize only *separating* formulas as in the standard temporal logic inference setting [46,43]. Our algorithm includes all the standard temporal operators that are typically used in MTL. However, we believe it is possible to improve the performance of the algorithm by omitting a temporal operator such as $\mathbf{U}_I$ for which the encoding can be substantially large.

From a practical point of view, an interesting future direction will be to lift our techniques to automatically synthesize STL formulas for verification.

References

1. Ammons, G., Bodík, R., Larus, J.R.: Mining specifications. In: Launchbury, J., Mitchell, J.C. (eds.) Conference Record of POPL 2002: The 29th SIGPLAN-SIGACT Symposium on Principles of Programming Languages, Portland, OR, USA, January 16-18, 2002. pp. 4–16. ACM (2002). https://doi.org/10.1145/503272.503275, https://doi.org/10.1145/503272.503275
2. Arif, M.F., Larraz, D., Echeverria, M., Reynolds, A., Chowdhury, O., Tinelli, C.: SYSLITE: syntax-guided synthesis of PLTL formulas from finite traces. In: FMCAD. pp. 93–103. IEEE (2020)
3. Asarin, E., Donzé, A., Maler, O., Nickovic, D.: Parametric identification of temporal properties. In: Proceedings of the Second International Conference on Runtime Verification. p. 147–160. RV'11, Springer-Verlag, Berlin, Heidelberg (2011). https://doi.org/10.1007/978-3-642-29860-8˙12, https://doi.org/10.1007/978-3-642-29860-8_12
4. Baldor, K., Niu, J.: Monitoring dense-time, continuous-semantics, metric temporal logic. In: RV. Lecture Notes in Computer Science, vol. 7687, pp. 245–259. Springer (2012)
5. Barbosa, H., Barrett, C.W., Brain, M., Kremer, G., Lachnitt, H., Mann, M., Mohamed, A., Mohamed, M., Niemetz, A., Nötzli, A., Ozdemir, A., Preiner, M., Reynolds, A., Sheng, Y., Tinelli, C., Zohar, Y.: cvc5: A versatile and industrial-strength SMT solver. In: TACAS (1). Lecture Notes in Computer Science, vol. 13243, pp. 415–442. Springer (2022)
6. Barrett, C.W., Sebastiani, R., Seshia, S.A., Tinelli, C.: Satisfiability modulo theories. In: Handbook of Satisfiability, Frontiers in Artificial Intelligence and Applications, vol. 336, pp. 1267–1329. IOS Press (2021)
7. Bartocci, E., Deshmukh, J.V., Donzé, A., Fainekos, G., Maler, O., Nickovic, D., Sankaranarayanan, S.: Specification-based monitoring of cyber-physical systems: A survey on theory, tools and applications. In: Lectures on Runtime Verification, Lecture Notes in Computer Science, vol. 10457, pp. 135–175. Springer (2018)
8. Bartocci, E., Mateis, C., Nesterini, E., Nickovic, D.: Survey on mining signal temporal logic specifications. Information and Computation **289**, 104957 (2022). https://doi.org/https://doi.org/10.1016/j.ic.2022.104957, https://www.sciencedirect.com/science/article/pii/S0890540122001122
9. Basin, D.A., Klaedtke, F., Zalinescu, E.: Algorithms for monitoring real-time properties. In: Khurshid, S., Sen, K. (eds.) Runtime Verification - Second International Conference, RV 2011, San Francisco, CA, USA, September 27-30, 2011, Revised Selected Papers. Lecture Notes in Computer Science, vol. 7186, pp. 260–275. Springer (2011). https://doi.org/10.1007/978-3-642-29860-8_20, https://doi.org/10.1007/978-3-642-29860-8_20
10. Basin, D.A., Krstic, S., Traytel, D.: Almost event-rate independent monitoring of metric dynamic logic. In: RV. Lecture Notes in Computer Science, vol. 10548, pp. 85–102. Springer (2017)
11. Basin, D.A., Krstic, S., Traytel, D.: AERIAL: almost event-rate independent algorithms for monitoring metric regular properties. In: RV-CuBES. Kalpa Publications in Computing, vol. 3, pp. 29–36. EasyChair (2017)
12. Bjørner, D., Havelund, K.: 40 years of formal methods - some obstacles and some possibilities? In: FM. Lecture Notes in Computer Science, vol. 8442, pp. 42–61. Springer (2014)

13. Bombara, G., Belta, C.: Offline and online learning of signal temporal logic formulae using decision trees. ACM Trans. Cyber-Phys. Syst. **5**(3) (mar 2021). https://doi.org/10.1145/3433994, `https://doi.org/10.1145/3433994`
14. Bombara, G., Vasile, C.I., Penedo, F., Yasuoka, H., Belta, C.: A decision tree approach to data classification using signal temporal logic. In: Proceedings of the 19th International Conference on Hybrid Systems: Computation and Control. p. 1–10. HSCC '16, Association for Computing Machinery, New York, NY, USA (2016). https://doi.org/10.1145/2883817.2883843, `https://doi.org/10.1145/2883817.2883843`
15. Camacho, A., McIlraith, S.A.: Learning interpretable models expressed in linear temporal logic. In: ICAPS. pp. 621–630. AAAI Press (2019)
16. Camacho, A., McIlraith, S.A.: Learning interpretable models expressed in linear temporal logic. Proceedings of the International Conference on Automated Planning and Scheduling **29**(1), 621–630 (May 2021). https://doi.org/10.1609/icaps.v29i1.3529, `https://ojs.aaai.org/index.php/ICAPS/article/view/3529`
17. Chattopadhyay, A., Mamouras, K.: A verified online monitor for metric temporal logic with quantitative semantics. In: RV. Lecture Notes in Computer Science, vol. 12399, pp. 383–403. Springer (2020)
18. Cimatti, A., Griggio, A., Schaafsma, B.J., Sebastiani, R.: The mathsat5 SMT solver. In: TACAS. Lecture Notes in Computer Science, vol. 7795, pp. 93–107. Springer (2013)
19. Clark, B., Cesare, T.: Satisfiability Modulo Theories, pp. 305–343. Springer International Publishing (2018). https://doi.org/10.1007/978-3-319-10575-8˙11, `https://doi.org/10.1007/978-3-319-10575-8_11`
20. Dang, T., Stolz, V. (eds.): Runtime Verification - 22nd International Conference, RV 2022, Tbilisi, Georgia, September 28-30, 2022, Proceedings, Lecture Notes in Computer Science, vol. 13498. Springer (2022)
21. Deshmukh, J.V., Donzé, A., Ghosh, S., Jin, X., Juniwal, G., Seshia, S.A.: Robust online monitoring of signal temporal logic. In: RV. Lecture Notes in Computer Science, vol. 9333, pp. 55–70. Springer (2015)
22. Dokhanchi, A., Hoxha, B., Fainekos, G.: On-line monitoring for temporal logic robustness. In: RV. Lecture Notes in Computer Science, vol. 8734, pp. 231–246. Springer (2014)
23. Donzé, A., Ferrère, T., Maler, O.: Efficient robust monitoring for STL. In: CAV. Lecture Notes in Computer Science, vol. 8044, pp. 264–279. Springer (2013)
24. Eisner, C., Fisman, D., Havlicek, J., Lustig, Y., McIsaac, A., Van Campenhout, D.: Reasoning with temporal logic on truncated paths. In: Hunt, W.A., Somenzi, F. (eds.) Computer Aided Verification. pp. 27–39. Springer Berlin Heidelberg, Berlin, Heidelberg (2003)
25. Fijalkow, N., Lagarde, G.: The complexity of learning linear temporal formulas from examples. In: ICGI. Proceedings of Machine Learning Research, vol. 153, pp. 237–250. PMLR (2021)
26. Gaglione, J., Neider, D., Roy, R., Topcu, U., Xu, Z.: Learning linear temporal properties from noisy data: A maxsat-based approach. In: ATVA. Lecture Notes in Computer Science, vol. 12971, pp. 74–90. Springer (2021)
27. Gorostiaga, F., Sánchez, C.: Hlola: a very functional tool for extensible stream runtime verification. In: Groote, J.F., Larsen, K.G. (eds.) Tools and Algorithms for the Construction and Analysis of Systems - 27th International Conference, TACAS 2021. Lecture Notes in Computer Science, vol. 12652, pp. 349–356. Springer (2021). https://doi.org/10.1007/978-3-030-72013-1_18

28. Havelund, K., Peled, D.: Runtime verification: From propositional to first-order temporal logic. In: RV. Lecture Notes in Computer Science, vol. 11237, pp. 90–112. Springer (2018)
29. Ho, H., Ouaknine, J., Worrell, J.: Online monitoring of metric temporal logic. In: Bonakdarpour, B., Smolka, S.A. (eds.) Runtime Verification - 5th International Conference, RV 2014, Toronto, ON, Canada, September 22-25, 2014. Proceedings. Lecture Notes in Computer Science, vol. 8734, pp. 178–192. Springer (2014). https://doi.org/10.1007/978-3-319-11164-3_15, https://doi.org/10.1007/978-3-319-11164-3_15
30. Hoxha, B., Dokhanchi, A., Fainekos, G.: Mining parametric temporal logic properties in model-based design for cyber-physical systems. Int. J. Softw. Tools Technol. Transf. **20**(1), 79–93 (2018)
31. Hunter, P., Ouaknine, J., Worrell, J.: Expressive completeness for metric temporal logic. In: 2013 28th Annual ACM/IEEE Symposium on Logic in Computer Science. pp. 349–357 (2013). https://doi.org/10.1109/LICS.2013.41
32. Kane, A., Chowdhury, O., Datta, A., Koopman, P.: A case study on runtime monitoring of an autonomous research vehicle (ARV) system. In: RV. Lecture Notes in Computer Science, vol. 9333, pp. 102–117. Springer (2015)
33. Kempa, B., Zhang, P., Jones, P.H., Zambreno, J., Rozier, K.Y.: Embedding online runtime verification for fault disambiguation on robonaut2. In: FORMATS. Lecture Notes in Computer Science, vol. 12288, pp. 196–214. Springer (2020)
34. Kong, Z., Jones, A., Belta, C.: Temporal logics for learning and detection of anomalous behavior. IEEE Transactions on Automatic Control **62**(3), 1210–1222 (2017). https://doi.org/10.1109/TAC.2016.2585083
35. Kong, Z., Jones, A., Medina Ayala, A., Aydin Gol, E., Belta, C.: Temporal logic inference for classification and prediction from data. In: Proceedings of the 17th International Conference on Hybrid Systems: Computation and Control. p. 273–282. HSCC '14, Association for Computing Machinery, New York, NY, USA (2014). https://doi.org/10.1145/2562059.2562146, https://doi.org/10.1145/2562059.2562146
36. Konrad, S., Cheng, B.H.C.: Real-time specification patterns. In: Roman, G., Griswold, W.G., Nuseibeh, B. (eds.) 27th International Conference on Software Engineering (ICSE 2005), 15-21 May 2005, St. Louis, Missouri, USA. pp. 372–381. ACM (2005). https://doi.org/10.1145/1062455.1062526, https://doi.org/10.1145/1062455.1062526
37. Koymans, R.: Specifying real-time properties with metric temporal logic. Real Time Syst. **2**(4), 255–299 (1990)
38. Lima, L., Herasimau, A., Raszyk, M., Traytel, D., Yuan, S.: Explainable online monitoring of metric temporal logic. In: TACAS (2). Lecture Notes in Computer Science, vol. 13994, pp. 473–491. Springer (2023)
39. Linard, A., Tumova, J.: Active learning of signal temporal logic specifications. In: 2020 IEEE 16th International Conference on Automation Science and Engineering (CASE). pp. 779–785 (2020). https://doi.org/10.1109/CASE48305.2020.9216778
40. Lutz, S., Neider, D., Roy, R.: Specification sketching for linear temporal logic. CoRR **abs/2206.06722** (2022)
41. Maler, O., Nickovic, D.: Monitoring temporal properties of continuous signals. In: Lakhnech, Y., Yovine, S. (eds.) Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems. pp. 152–166. Springer Berlin Heidelberg, Berlin, Heidelberg (2004)

42. Meel, K.S., Strichman, O. (eds.): 25th International Conference on Theory and Applications of Satisfiability Testing, SAT 2022, August 2-5, 2022, Haifa, Israel, LIPIcs, vol. 236. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2022)
43. Mohammadinejad, S., Deshmukh, J.V., Puranic, A.G., Vazquez-Chanlatte, M., Donzé, A.: Interpretable classification of time-series data using efficient enumerative techniques. In: HSCC '20: 23rd ACM International Conference on Hybrid Systems: Computation and Control, Sydney, New South Wales, Australia, April 21-24, 2020. pp. 9:1–9:10. ACM (2020). https://doi.org/10.1145/3365365.3382218
44. de Moura, L.M., Bjørner, N.S.: Z3: an efficient SMT solver. In: TACAS. Lecture Notes in Computer Science, vol. 4963, pp. 337–340. Springer (2008)
45. de Moura, L.M., Bjørner, N.S.: Satisfiability modulo theories: introduction and applications. Commun. ACM **54**(9), 69–77 (2011)
46. Neider, D., Gavran, I.: Learning linear temporal properties. In: Bjørner, N.S., Gurfinkel, A. (eds.) 2018 Formal Methods in Computer Aided Design, FMCAD 2018, Austin, TX, USA, October 30 - November 2, 2018. pp. 1–10. IEEE (2018). https://doi.org/10.23919/FMCAD.2018.8603016, https://doi.org/10.23919/FMCAD.2018.8603016
47. Nenzi, L., Silvetti, S., Bartocci, E., Bortolussi, L.: A robust genetic algorithm for learning temporal specifications from data. In: McIver, A., Horvath, A. (eds.) Quantitative Evaluation of Systems. pp. 323–338. Springer International Publishing, Cham (2018)
48. Ouaknine, J., Worrell, J.: Some recent results in metric temporal logic. In: Cassez, F., Jard, C. (eds.) Formal Modeling and Analysis of Timed Systems. pp. 1–13. Springer Berlin Heidelberg, Berlin, Heidelberg (2008)
49. Pnueli, A.: The temporal logic of programs. In: FOCS. pp. 46–57. IEEE Computer Society (1977)
50. Raha, R., Roy, R., Fijalkow, N., Neider, D.: Scalable anytime algorithms for learning fragments of linear temporal logic. In: Fisman, D., Rosu, G. (eds.) Tools and Algorithms for the Construction and Analysis of Systems. pp. 263–280. Springer International Publishing, Cham (2022)
51. Riener, H.: Exact synthesis of LTL properties from traces. In: FDL. pp. 1–6. IEEE (2019)
52. Roy, R., Fisman, D., Neider, D.: Learning interpretable models in the property specification language. In: IJCAI. pp. 2213–2219. ijcai.org (2020)
53. Rozier, K.Y.: Specification: The biggest bottleneck in formal methods and autonomy. In: VSTTE. Lecture Notes in Computer Science, vol. 9971, pp. 8–26 (2016)
54. Silvetti, S., Nenzi, L., Bortolussi, L., Bartocci, E.: A robust genetic algorithm for learning temporal specifications from data. CoRR (2017), http://arxiv.org/abs/1711.06202
55. Thati, P., Rosu, G.: Monitoring algorithms for metric temporal logic specifications. In: Havelund, K., Rosu, G. (eds.) Proceedings of the Fourth Workshop on Runtime Verification, RV@ETAPS 2004, Barcelona, Spain, April 3, 2004. vol. 113, pp. 145–162. Elsevier (2004). https://doi.org/10.1016/j.entcs.2004.01.029
56. Xu, Z., Topcu, U.: Transfer of temporal logic formulas in reinforcement learning. In: Kraus, S. (ed.) Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019. pp. 4010–4018. ijcai.org (2019). https://doi.org/10.24963/ijcai.2019/557
57. Yang, H., Hoxha, B., Fainekos, G.E.: Querying parametric temporal logic properties on embedded systems. In: Nielsen, B., Weise, C. (eds.) Testing Software and Systems - 24th IFIP WG 6.1 International Conference, ICTSS 2012, Aalborg,

Denmark, November 19-21, 2012. Proceedings. Lecture Notes in Computer Science, vol. 7641, pp. 136–151 (2012), https://doi.org/10.1007/978-3-642-34691-0_11

A Semantics of MTL on infinite signals

This section formalizes the standard continuous semantics of MTL on infinite system executions. Towards this, we follow the framework of [48]. Note that, in [48], the authors provided the semantics for the timed-Until operator, and the semantics for the other temporal operators can be deduced using syntactic relations. As we do not consider the timed-Until operator in our setting, we directly provide the semantics for the temporal operators we opt for in our settings. Given an infinite signal $\boldsymbol{x}$, an MTL formula φ and a time point $t \geq 0$,

$$
\begin{aligned}
(\boldsymbol{x}, t) \models p &\iff p \in \boldsymbol{x}(t) \\
(\boldsymbol{x}, t) \models \neg p &\iff p \notin \boldsymbol{x}(t) \\
(\boldsymbol{x}, t) \models \varphi_1 \wedge \varphi_2 &\iff (\boldsymbol{x}, t) \models \varphi_1 \text{ and } (\boldsymbol{x}, t) \models \varphi_2 \\
(\boldsymbol{x}, t) \models \varphi_1 \vee \varphi_2 &\iff (\boldsymbol{x}, t) \models \varphi_1 \text{ or } (\boldsymbol{x}, t) \models \varphi_2 \\
(\boldsymbol{x}, t) \models \mathbf{F}_{[a,b]}\, \varphi &\iff \exists t' \in [t+a, t+b] \text{ such that } (\boldsymbol{x}, t') \models \varphi \\
(\boldsymbol{x}, t) \models \mathbf{G}_{[a,b]}\, \varphi &\iff \forall t' \in [t+a, t+b], (\boldsymbol{x}, t') \models \varphi \\
(\boldsymbol{x}, t) \models \varphi_1\, \mathbf{U}_{[a,b]}\, \varphi_2 &\iff \exists t' \in [t+a, t+b], (\boldsymbol{x}, t') \models \varphi_2 \text{ and } \forall t'' \in [t, t'], (\boldsymbol{x}, t) \models \varphi_1
\end{aligned}
$$

We read $(\boldsymbol{x}, t) \models \varphi$ as '$\boldsymbol{x}$ satisfies the formula φ at time point t'. The signal $\boldsymbol{x}$ satisfies the formula φ if and only if it satisfies the formula at time point 0, i.e., $\boldsymbol{x} \models \varphi \iff (\boldsymbol{x}, 0) \models \varphi$.

B Proof of Lemma 1

In this section, we prove Lemma 1 that justifies our choice for the semantics of MTL on finite signal prefixes. We, in fact, prove a stronger statement from which Lemma 1 follows: for all $t \in [0, T), (\boldsymbol{x}_{\mathbb{T}}, t) \models_{\mathrm{f}} \varphi$ if there exists a signal $\boldsymbol{x} \in ext(\boldsymbol{x}_{\mathbb{T}})$ such that $(\boldsymbol{x}, t) \models \varphi$.

The proof now proceeds via an induction on the MTL formula φ.

- For the base case, let $\varphi = p \in \mathcal{P}$. Then, for all $t \in [0, T)$, if there exists $\boldsymbol{x} \in ext(\boldsymbol{x}_{\mathbb{T}})$ such that $(\boldsymbol{x}, t) \models p$, then $(\boldsymbol{x}_{\mathbb{T}}, t) \models_{\mathrm{f}} p$ since $(\boldsymbol{x}, t) \models \varphi$ and thus, $(\boldsymbol{x}_{\mathbb{T}}, t) \models \varphi$. The same argument extends to the neg operator.
- Let $\varphi = \varphi_1 \wedge \varphi_2$. Then, for all $t \in [0, T)$, if there exists $\boldsymbol{x} \in ext(\boldsymbol{x}_{\mathbb{T}})$ such that $(\boldsymbol{x}, t) \models \varphi_1$ and $(\boldsymbol{x}, t) \models \varphi_2$. Then, $(\boldsymbol{x}_{\mathbb{T}}, t) \models_{\mathrm{f}} \varphi_1$ and $(\boldsymbol{x}_{\mathbb{T}}, t) \models_{\mathrm{f}} \varphi_2$ by induction hypothesis. The same argument extends to the $\vee$ operator.
- Let $\varphi = \mathbf{F}_{[a,b]}\, \psi$ and fix a time point $t \in [0, T)$. We have to prove if there exists a signal $\boldsymbol{x} \in ext(\boldsymbol{x}_{\mathbb{T}})$ such that, $(\boldsymbol{x}, t) \models \mathbf{F}_{[a,b]}\, \psi$, then $(\boldsymbol{x}_{\mathbb{T}}, t) \models_{\mathrm{f}} \mathbf{F}_{[a,b]}\, \psi$. Now by definition of $\models$, $\exists t' \in [t+a, t+b]$ such that, $(\boldsymbol{x}, t') \models \psi$. Now there are two cases: (i) $t + b \geq T$: in this case, $(\boldsymbol{x}_{\mathbb{T}}, t) \models_{\mathrm{f}} \mathbf{F}_{[a,b]}\, \psi$, and (ii) $t + b < T$: then, $t' < T$ and by induction hypothesis, $(\boldsymbol{x}_{\mathbb{T}}, t') \models_{\mathrm{f}} \psi$ as $(\boldsymbol{x}, t) \models \psi$. Hence, $(\boldsymbol{x}_{\mathbb{T}}, t) \models_{\mathrm{f}} \varphi$. The case for $\varphi = \mathbf{G}_{[a,b]}\, \psi$ can be proved similarly. □

C Proof of Lemma 2

In this section, we prove Lemma 2 that shows that an MTL formula φ with $fr(\varphi) \leq K$ cannot distinguish two signals and their prefixes if they are identical up to time K. We will prove this by induction on the structure of φ. In particular, we will prove the following:

For any K, let φ be a formula with $fr(\varphi) \leq K$ and $\boldsymbol{x}$ and $\boldsymbol{y}$ be two signals such that $\boldsymbol{x}_{[0,K]} = \boldsymbol{y}_{[0,K]}$. Then, for all $T \in \mathbb{R}^{\geq 0}$, $\boldsymbol{x}_{\mathbb{T}} \models_{\mathrm{f}} \varphi$ if and only if $\boldsymbol{y}_{\mathbb{T}} \models_{\mathrm{f}} \varphi$.

- For the base case, let $\varphi = p$. Then, $p \in \boldsymbol{x}_{\mathbb{T}}(0)$ and as $\boldsymbol{x}_{[0,K]} = \boldsymbol{y}_{[0,K]}$, $p \in \boldsymbol{y}_{\mathbb{T}}(0)$. Hence, $\boldsymbol{y}_{\mathbb{T}} \models_{\mathrm{f}} p$. This can be similarly seen for the case where $\varphi = \neg p$.

- The proof for the cases where $\varphi = \varphi_1 \vee \varphi_2$ or $\varphi = \varphi_1 \wedge \varphi_2$ can be derived easily.

- Let $\varphi = \mathbf{F}_{[a,b]}\, \varphi_1$. Let us fix a T such that $\boldsymbol{x}_{\mathbb{T}} \models_{\mathrm{f}} \mathbf{F}_{[a,b]}\, \varphi$. If $b \geq t$, then $\boldsymbol{y}_{\mathbb{T}} \models_{\mathrm{f}} \mathbf{F}_{[a,b]}\, \varphi$ trivially. If not, then there exists a time point $t \in [a,b]$ such that $(\boldsymbol{x}_{\mathbb{T}}, t) \models_{\mathrm{f}} \varphi_1$. Now, let $\boldsymbol{x'} = \boldsymbol{x}^{[t:]}$ and $\boldsymbol{y'} = \boldsymbol{y}^{[t:]}$ be the signals obtained by shifting the original signals by $-t$. Formally, $\forall t' \in \mathbb{R}^{\geq 0}$, $\boldsymbol{x'}(t') = \boldsymbol{x}(t'+t)$ and $\boldsymbol{y'}(t') = \boldsymbol{y}(t'+t)$. Note that, $\boldsymbol{x'}_{[0,K-t]} = \boldsymbol{y'}_{[0,K-t]}$. Also, $fr(\varphi_1) = fr(\varphi) - b \leq K - b \leq K - t$ and $\boldsymbol{x'}_{[0,K-t]} \models_{\mathrm{f}} \varphi$. Then, following induction hypothesis, $\boldsymbol{y'}_{[0,K-t]} \models_{\mathrm{f}} \varphi_1$ which implies that $(\boldsymbol{y}_{\mathbb{T}}, t) \models_{\mathrm{f}} \varphi_1$. Hence, $\boldsymbol{y}_{\mathbb{T}} \models_{\mathrm{f}} \mathbf{F}_{[a,b]}\, \varphi$. The case where $\varphi = \mathbf{G}_{[a,b]}\, \varphi_1$ can be proved similarly.

- Let $\varphi = \varphi_1 \, \mathbf{U}_{[a,b]}\, \varphi_2$. Again, similar to above, fix a T such that $\boldsymbol{x}_{\mathbb{T}} \models_{\mathrm{f}} \varphi_1 \, \mathbf{U}_{[a,b]}\, \varphi_2$. Let us first assume that $b \leq T$. Then, $\exists t \in [a,b]$ such that $(\boldsymbol{x}_{\mathbb{T}}, t) \models_{\mathrm{f}} \varphi_2$ and $\forall t' \in [0,t]$, $(\boldsymbol{x}_{\mathbb{T}}, t') \models_{\mathrm{f}} \varphi_1$. Now as $fr(\varphi_1)$ and $fr(\varphi_2)$ are both $\leq K - b \leq K - t$. Hence again using similar methods as above, one can prove that $(\boldsymbol{y}_{\mathbb{T}}, t) \models_{\mathrm{f}} \varphi_2$ and $\forall t' \in [0,t]$, $(\boldsymbol{y}_{\mathbb{T}}, t') \models_{\mathrm{f}} \varphi_1$. Hence, $\boldsymbol{y}_{\mathbb{T}} \models_{\mathrm{f}} \varphi_1 \, \mathbf{U}_{[a,b]}\, \varphi_2$.

D Proof of Lemma 5

In this section, we prove Lemma 5 that proves the correctness of our construction of $\mathcal{I}_{\varphi}(\boldsymbol{x}_{\mathbb{T}})$ given a prefix $\boldsymbol{x}_{\mathbb{T}}$. Recall that the lemma says, for all $t \in [0,T)$, $(\boldsymbol{x}_{\mathbb{T}}, t) \models_{\mathrm{f}} \varphi$ if and only if $t \in I$ for some $I \in \mathcal{I}_{\varphi}(\boldsymbol{x}_{\mathbb{T}})$. We prove both directions together by induction on the structure of the formula φ.

For the base case, one can check that for all $t \in [0,T)$, $t \in \mathcal{I}_p(\boldsymbol{x}_{\mathbb{T}})$ if and only if $t \in I$ for some $I \in \mathcal{I}_{\varphi}(\boldsymbol{x}_{\mathbb{T}})$ by construction. The proof for the *neg* operator and the boolean connectives $\wedge$ and $\vee$ follow from the correctness of the construction in the work of [41]. Here, we provide the proofs for two timed-temporal operators as their semantics differ from the work in [41].

Let $\varphi = \mathbf{F}_{[a,b]}\, \psi$. To show the forward direction, let $t \in I$ for some $I \in \mathcal{I}_{\varphi}(\boldsymbol{x}_{\mathbb{T}})$. We have to prove that, $(\boldsymbol{x}_{\mathbb{T}}, t) \models_{\mathrm{f}} \mathbf{F}_{[a,b]}\, \psi$. In particular, $t \in \mathcal{I}^{\cup}_{\varphi}(\boldsymbol{x}_{\mathbb{T}})$ by definition, i.e., $t \in \big(\bigcup_{I \in \mathcal{I}_{\psi}(\boldsymbol{x}_{\mathbb{T}})} I \ominus [a,b]\big) \cup [T-b, T)$. There are two cases: (i) $t \in [T-b, T)$: in this case, $t + b \geq T$ and by definition of $\models_{\mathrm{f}}$, $(\boldsymbol{x}_{\mathbb{T}}, t) \models_{\mathrm{f}} \varphi$, or (ii) $t \in \big(\bigcup_{I \in \mathcal{I}_{\psi}(\boldsymbol{x}_{\mathbb{T}})} I \ominus [a,b]\big)$: Fix the interval $I' = [t_1, t_2) \in \mathcal{I}_{\psi}(\boldsymbol{x}_{\mathbb{T}})$ such that, $t \in (I' \ominus [a,b])$. By induction hypothesis, for all $t' \in I'$, $(\boldsymbol{x}_{\mathbb{T}}, t') \models_{\mathrm{f}} \psi$. Now, $t < t_2 - a \implies t + a < t_2$ and $t \geq t_1 - b \implies t + b \geq t_1$. Hence,

$I' = [t_1, t_2) \supset [t+a, t+b]$. Hence, $\exists t' \in [t+a, t+b]$ such that, $(\boldsymbol{x}_{\mathbb{T}}, t') \models_{\mathrm{f}} \psi$ and henceforth, $(\boldsymbol{x}_{\mathbb{T}}, t) \models_{\mathrm{f}} \varphi$.

For the backward direction, we assume that, $(\boldsymbol{x}_{\mathbb{T}}, t) \models_{\mathrm{f}} \mathbf{F}_{[a,b]}\, \psi$ and prove that, $t \in I$ for some $I \in \mathcal{I}_{\varphi}(\boldsymbol{x}_{\mathbb{T}})$. In particular, we show that $t \in \mathcal{I}^{\cup}_{\varphi}(\boldsymbol{x}_{\mathbb{T}}) = \left(\bigcup_{I \in \mathcal{I}_{\psi}(\boldsymbol{x}_{\mathbb{T}})} I \ominus [a,b]\right) \cup [T-b, T)$ and the rest of the argument follows from the fact that, $\mathcal{I}_{\varphi}(\boldsymbol{x}_{\mathbb{T}})$ is obtained by taking the maximal disjoint intervals of $\mathcal{I}^{\cup}_{\varphi}(\boldsymbol{x}_{\mathbb{T}})$. Now, by definition of $\models_{\mathrm{f}}$, there are two possibilities: (i) $t+b \geq T$: then, $t \in [T-b, T)$ and hence, $t \in \mathcal{I}^{\cup}_{\varphi}(\boldsymbol{x}_{\mathbb{T}})$, or (ii) $\exists t' \in [t+a, t+b]$ such that, $(\boldsymbol{x}_{\mathbb{T}}, t') \models_{\mathrm{f}} \psi$. Now, by induction hypothesis, $t' \in I$ for some $I \in \mathcal{I}_{\psi}(\boldsymbol{x}_{\mathbb{T}})$. Let $I = [t_1, t_2)$. Now, $t_2 - a > t' - a \geq t$ and $t_1 - b \leq t' - b \leq t$. This implies that, $t \in [t_1 - b, t_2 - a) = (I \ominus [a,b])$ which proves that, $t \in \mathcal{I}^{\cup}_{\varphi}(\boldsymbol{x}_{\mathbb{T}})$.

The proof for the operator $\mathbf{G}_I$ can be derived similarly. □

E All the constraints

In this section, we provide the encodings of $\Phi^{n}_{\mathcal{S},k} \coloneqq \Phi^{str}_{n,\mathcal{S},k} \wedge \Phi^{fr}_{n,\mathcal{S},k} \wedge \Phi^{sem}_{n,\mathcal{S},k}$ that we did not include in the paper.

E.1 Structural constraints

Here we formally introduce the encoding for the structural constraints Φ^{str}. For each Node i containing operator λ, we define the following two functions:

$$\mathit{exactly-one-left}(i) = \Big[\bigwedge_{1 \leq i \leq n} \bigvee_{1 \leq j \leq i} l_{i,j}] \wedge [\bigwedge_{2 \leq i \leq n} \bigwedge_{1 \leq j \leq j' \leq n} \neg l_{i,j} \vee \neg l_{i,j'} \Big], \text{ and}$$

$$\mathit{exactly-one-right}(i) = \Big[\bigwedge_{1 \leq i \leq n} \bigvee_{1 \leq j \leq i} r_{i,j}] \wedge [\bigwedge_{2 \leq i \leq n} \bigwedge_{1 \leq j \leq j' \leq n} \neg r_{i,j} \vee \neg r_{i,j'} \Big]$$

that defines that the node contains exactly one left child and exactly one right child, respectively.

Now let $\Lambda = \mathcal{P} \cup U_{\Lambda} \cup B_{\Lambda}$, where U_{Λ} denotes the set of unary operators and B_{Λ} denotes the set of binary operators. Then the encoding of the structural constraints contains the following:

$$\Big[\bigwedge_{1\le i\le n}\bigvee_{\lambda\in\Lambda} x_{i,\lambda}\Big]\wedge\Big[\bigwedge_{1\le i\le n}\bigwedge_{\lambda\neq\lambda'\in\Lambda}\neg x_{i,\lambda}\vee\neg x_{i,\lambda'}\Big]\wedge \tag{9}$$

$$\bigwedge_{1\le i\le n}\left(\bigvee_{p\in\mathcal{P}} x_{i,p}\rightarrow\Big[\bigwedge_{1\le j\le n}\neg l_{i,j}\wedge\bigwedge_{1\le j\le n}\neg r_{i,j'}\Big]\right)\wedge \tag{10}$$

$$\bigwedge_{1\le i\le n}\left(\bigvee_{\lambda\in U_\Lambda} x_{i,\lambda}\rightarrow\Big[exactly-one-left(i)\wedge\bigwedge_{1\le j\le n}\neg r_{i,j'}\Big]\right)\wedge \tag{11}$$

$$\bigwedge_{1\le i\le n}\left(\bigvee_{\lambda\in B_\Lambda} x_{i,\lambda}\rightarrow\Big[exactly-one-left(i)\wedge exactly-one-right(i)\Big]\right) \tag{12}$$

$$\bigwedge_{\substack{1\le i\le n\\1\le j<i}} x_{i,\neg}\wedge l_{i,j}\rightarrow\Big[\bigvee_{p\in\mathcal{P}} x_{j,p}\Big] \tag{13}$$

Constraint 9 encodes the fact that each node only contains one operator or one proposition. Constraint 10 imposes that the nodes containing a proposition do not have any child. Constraint 11 says that the nodes containing a unary operator contain exactly one child, while constraint 12 enforces that the nodes containing a binary operator contain exactly one left and exactly one right child. Finally, Constraint 13 imposes that the *neg* operator can occur only in front of propositions.

E.2 Future-reach constraints

Here we formally introduce the constraints for symbolically encoding the future-reach of the prospective formula φ. The formula $\Phi^{fr}_{n,\mathcal{S},k}$ contains the following:

$$\bigwedge_{1\le i\le n} x_{i,p}\rightarrow\big[f_i=0\big]\wedge$$

$$\bigwedge_{\substack{1\le i\le n\\1\le j<i}}(x_{i,\neg}\wedge l_{i,j})\rightarrow\big[f_i=f_j\big]\wedge$$

$$\bigwedge_{\substack{1\le i\le n\\1\le j,j'<i}}((x_{i,\vee}\vee x_{i,\wedge})\wedge l_{i,j}\wedge r_{i,j'})\rightarrow\big[f_i=\max(f_j,f'_j)\big]\wedge$$

$$\bigwedge_{\substack{1\le i\le n\\1\le j<i}}(x_{i,\mathbf{F}_I}\wedge l_{i,j})\rightarrow\big[f_i=f_j+b_i\big]\wedge$$

$$\bigwedge_{\substack{1\le i\le n\\1\le j<i}}(x_{i,\mathbf{G}_I}\wedge l_{i,j})\rightarrow\big[f_i=f_j+b_i\big]\wedge$$

$$(f_n\le K)$$

Each line above encodes the future-reach value for each operator using the inductive definition of the future-reach, as described in Section 3. The last line imposes the fact that the future-reach of the prospective formula is within the bound K.

E.3 Semantic constraints

Here we introduce the constraints to encode the semantics of the $\wedge$ and the $\mathbf{G}_I$ operators that we left out in the main text of the paper.

For the $\wedge$-operator, we have the following constraint:

$$\bigwedge_{\substack{1\leq i\leq n\\ 1\leq j,j'<i}} x_{i,\wedge} \wedge l_{i,j} \wedge r_{i,j'} \rightarrow \big[\bigwedge_{1\leq s\leq|\mathcal{S}|} \mathit{comp}_s(i,k) \wedge \mathit{union}_s(k,j,j')\big],$$

This encodes the relation for $\wedge$ operator as described in Table 1. We introduce an intermediate set of intervals $\tilde{\mathcal{I}}_k$ that contains the maximal disjoint intervals of union of $\mathcal{I}_{\varphi[j]}(\boldsymbol{x}^s_{\mathbb{T}})$ and $\mathcal{I}_{\varphi[j']}(\boldsymbol{x}^s_{\mathbb{T}})$. Then $\mathit{comp}_s(i,k)$ denotes that, $\mathcal{I}_{\varphi[i]}(\boldsymbol{x}^s_{\mathbb{T}})$ is the complement of the set of intervals in $\tilde{\mathcal{I}}_k$, making it the set containing maximal disjoint intervals of *intersection* of $\mathcal{I}^{\cup}_{\varphi[j]}(\boldsymbol{x}^s_{\mathbb{T}})$ and $\mathcal{I}^{\cup}_{\varphi[j']}(\boldsymbol{x}^s_{\mathbb{T}})$.

For the $\mathbf{G}_I$ operator where I is encoded using a_i, b_i we have the following constraint:

$$\bigwedge_{\substack{1\leq i\leq n\\ 1\leq j<i}} x_{i,\mathbf{G}_I} \wedge l_{i,j} \rightarrow \big[\bigwedge_{1\leq s\leq|\mathcal{S}|} \mathit{union}''_s(i,k',k') \wedge \mathit{comp}_s(k',k) \wedge \ominus_s^{[a_i,b_i]}(k,j)\big].$$

based on the relation for the $\mathbf{G}_{[a,b]}$ operator in Table 1. Similar to the encoding of $\mathbf{F}_{[a,b]}$ operator, we rely on an intermediate set of intervals $\tilde{\mathcal{I}}_k$ and $\tilde{\mathcal{I}}_{k'}$ encoded using some auxiliary variables. Also, we use the formula $\ominus_s^{[a_i,b_i]}(k,j)$ to encode that the intervals in $\tilde{\mathcal{I}}_k$ can be obtained by performing $I \ominus [a,b]$ to each interval I in $\mathcal{I}_{\varphi[j]}(\boldsymbol{x}^s_{\mathbb{T}})$, where $a_i = a$ and $b_i = b$. Then $\mathit{comp}_s(k',k)$ encodes that $\tilde{\mathcal{I}}_{k'}$ is the complement of $\tilde{\mathcal{I}}_k$. Finally, the formula $\mathit{union}''(i,k',k')$ encodes that $\mathcal{I}_{\varphi[i]}(\boldsymbol{x}^s_{\mathbb{T}})$ consists of the maximal disjoint intervals obtained by taking the union of the complement of $\mathcal{I}^{\cup}_{\varphi[j]}(\boldsymbol{x}^s_{\mathbb{T}})$ and $\{[T-a,T)\}$.

Similar to union' in the semantic constraints for $\mathbf{F}_I$ operator, the construction of $\mathit{union}''(i,k,k)$ is similar to that of $\mathit{union}(i,j,j')$ in that the constraints involved are similar to Constraints 5 to 7.

F Proof of Lemma 6

The proof of Lemma 6 is a direct consequence of the proofs of the following claims for the different formulas.

Claim. Let ι be a satisfying interpretation of $union_s(i, j, j')$. Then, the set $\mathcal{I}_i = \{[\iota(t^l_{i,1,s}), \iota(t^r_{i,1,s})), \ldots, [\iota(t^l_{i,m,s}), \iota(t^r_{i,m,s}))\}$ consists of the maximal disjoint intervals of the union of $\mathcal{I}_j = \{[\iota(t^l_{j,1,s}), \iota(t^r_{j,1,s})), \ldots, [\iota(t^l_{j,m,s}), \iota(t^r_{j,m,s}))\}$ and $\mathcal{I}_{j'} = \{[\iota(t^l_{j',1,s}), \iota(t^r_{j',1,s})), \ldots, [\iota(t^l_{j',m,s}), \iota(t^r_{j',m,s}))\}$.

Proof. For simplicity of the proof, we name $\iota(t^\sigma_{\kappa,m})$ as $\tau^\sigma_{\kappa,m}$ for $\sigma \in \{l, r\}$ and $\kappa \in \{i, j, j'\}$, and $[\tau^l_{\kappa,m}, \tau^r_{\kappa,m})$ as $\Gamma_{\kappa,m}$ for $\kappa \in \{i, j, j'\}$. Note that we drop the identifier s representing the prefix since the prefix is fixed throughout the proof.

For the forward direction, we show that any time point $t \in \Gamma_{i,m}$ belongs to some $\Gamma_{j,m'} \in \mathcal{I}_j$ or some $\Gamma_{j',m''} \in \mathcal{I}_{j'}$. Towards contradiction, we assume that $t \notin \Gamma_{j,m'}$ for any $\Gamma_{j,m'} \in \mathcal{I}_j$ and $t \notin \Gamma_{j',m''}$ for any $\Gamma_{j',m''} \in \mathcal{I}_{j'}$. Now, based on Constraint 5, both $\tau^l_{i,m}$ and $\tau^r_{i,m}$ appear in some intervals in $\mathcal{I}_j$ and $\mathcal{I}_{j'}$ as left and right bound, respectively. We consider two cases based on where $\tau^l_{i,m}$ and $\tau^r_{i,m}$ appear. First, $\tau^l_{i,m}$ and $\tau^r_{i,m}$ both appears, w.l.o.g, in $\mathcal{I}_j$. Now, let Γ_{j,m_1} and Γ_{j,m_1+1} be such that $\tau^r_{j,m_1} \leq t < \tau^l_{j,m_1+1}$. Intuitively, this means that t lies in between (and is adjacent to) the intervals Γ_{j,m_1} and Γ_{j,m_1+1}. Note that both τ^r_{j,m_1} and τ^l_{j,m_1+1} is not included in $\mathcal{I}_i$ since $\mathcal{I}_i$ consists of maximal disjoint intervals and $[\tau^r_{j,m_1}, \tau^l_{j,m_1+1}] \subset \Gamma_{i,m}$. Now, based on Constraint 6, τ^r_{j,m_1} and τ^l_{j,m_1+1} are included in some intervals in $\mathcal{I}_{j'}$. Note that if they are included in the same interval, then that interval also contains t raising the contradiction to our assumption that $t \notin \Gamma_{j',m''}$ for any $\Gamma_{j',m''} \in \mathcal{I}_{j'}$. Then τ^r_{j,m_1} and τ^l_{j,m_1+1} are not included in the same interval in $\mathcal{I}_{j'}$. Then, there exists $\Gamma_{j',m_2} \in \mathcal{I}_{j'}$ and $\Gamma_{j',m_2+1} \in \mathcal{I}_{j'}$ such that,

$$\tau^r_{j,m_1} < \tau^r_{j',m_2} \leq t < \tau^l_{j',m_2+1} < \tau^l_{j,m_1+1}$$

Now note that, τ^r_{j',m_2} and τ^l_{j',m_2+1} both are not included in any of the intervals in $\mathcal{I}_j$. Now, based on Constraint 7, both appear in $\mathcal{I}_i$. But that raises the contradiction to our assumption that $t \in \Gamma_{i,m}$.

For the other direction, we show that any time point, w.l.o.g, $t \in \Gamma_{j,m}$ belongs to some $\Gamma_{i,m'} \in \mathcal{I}_j$. For this, there can be three cases based on whether the bounds of $\Gamma_{j,m}$ appear as bounds in some interval $\Gamma_{i,m'} \in \mathcal{I}_i$ or not.

First, assume that both $\tau^l_{j,m}$ and $\tau^r_{j,m}$ appear as bounds τ^l_{i,m_1} and τ^r_{i,m_2} in $\mathcal{I}_k$ as stated by Constraint 5. We now claim that $m_1 = m_2$ meaning that τ^l_{i,m_1} and τ^r_{i,m_2} are bounds of the same intervals. Towards contradiction, let $m_1 + 1 \leq m_2$. Then, τ^r_{i,m_1} belongs to the interval $\Gamma_{j,m}$, and based on Constraint 6, and cannot be one of the bounds of Γ_{i,m_1}. Then, we have $\tau^l_{j,m} = \tau^l_{i,m_1} \leq t < \tau^r_{i,m_1} = \tau^r_{j,m}$

Second, assume that $\tau^l_{j,m}$ does not appear, while $\tau^r_{j,m}$ appears as bounds in $\mathcal{I}_k$. Now, based on Constraint 6, $\tau^l_{j,m}$ appears in one of the intervals $\Gamma_{j',m'}$ in $\mathcal{I}_{j'}$. Also, in that case, $\tau^l_{j',m'}$ appears as a left bound in $\mathcal{I}_k$, say $\mathcal{I}_{i,m_1}$. We now claim that $\tau^r_{i,m_1} > \tau^r_{j,m}$. Towards contradiction, we assume two cases. In first case,

$$\tau^l_{j',m'} = \tau^l_{i,m_1} < \tau^r_{i,m_1} < \tau^l_{j,m} < \tau^r_{j',m}$$

contradicting Constraint 6. In the second case,

$$\tau^l_{j',m'} = \tau^l_{i,m_1} < \tau^l_{j,m} < \tau^r_{i,m_1} < \tau^r_{j,m}$$

contradicting Constraint 7. From the two cases, we conclude $\tau^r_{i,m_1} > \tau^r_{j,m}$ and hence, $\tau^l_{i,m_1} < \tau^l_{j,m} \leq t < \tau^r_{j,m} < \tau^r_{i,m_1}$. The argument in the third case is similar to the arguments in the other two cases and can be seen easily.

Claim. Let ι be a satisfying interpretation of $comp_s(i,j)$. Then, the set $\mathcal{I}_i = \{[\iota(t^l_{i,1,s}), \iota(t^r_{i,1,s})), \ldots, [\iota(t^l_{i,m,s}), \iota(t^r_{i,m,s}))\}$ consists of the maximal disjoint intervals of the complement of $\mathcal{I}_j = \{[\iota(t^l_{j,1,s}), \iota(t^r_{j,1,s})), \ldots, [\iota(t^l_{j,m,s}), \iota(t^r_{j,m,s}))\}$.

Proof. We reuse the naming conventions for $\tau^\sigma_{\kappa,m}$) and $\Gamma_{\kappa,m}$ from the last proof. For the forward direction, we show that if $t \in \Gamma_{i,m}$ for some $\Gamma_{i,m} \in \mathcal{I}_i$ then $t \notin \Gamma_{j,m'}$ for any $\Gamma_{j,m'} \in \mathcal{I}_j$. First, let $m = 1$. Then, if $\tau^l_{j,1} = 0$, then Condition 2 gets triggered and $\tau^l_{i,1} = \tau^r_{j,1}$ and $\tau^r_{i,1} = \tau^r_{j,2}$. Hence, $\tau^r_{j,1} = \tau^l_{i,1} \leq t < \tau^r_{i,1} = \tau^l_{j,2}$. Also, if $\tau^l_{j,1} \neq 0$, then Condition 2 does not get triggered and $\tau^l_{i,1} = 0$ and $\tau^r_{i,1} = \tau^l_{j,1}$. Hence, $0 = \tau^l_{i,1} \leq t < \tau^r_{i,1} = \tau^l_{j,1}$. For $m \neq 1$, the reasoning works similarly.

For the other direction, we show that if $t \in \Gamma_{j,m}$ for some $\Gamma_{j,m} \in \mathcal{I}_j$ then $t \notin \Gamma_{i,m'}$ for any $\Gamma_{i,m'} \in \mathcal{I}_j$. The proof for this direction is almost identical to the proof for the forward direction and is a simple exercise.

Claim. Let ι be a satisfying interpretation of $\ominus^{[a,b]_s(k,j)}$. Then, the set $\mathcal{I}_i = \{[\iota(t^l_{i,1,s}), \iota(t^r_{i,1,s})), \ldots, [\iota(t^l_{i,m,s}), \iota(t^r_{i,m,s}))\}$ consists of the maximal disjoint intervals by applying $I \ominus [a,b]$ to the intervals I of $\mathcal{I}_j = \{[\iota(t^l_{j,1,s}), \iota(t^r_{j,1,s})), \ldots, [\iota(t^l_{j,m,s}), \iota(t^r_{j,m,s}))\}$.

Proof. The proof of this result follows by construction.

The proofs for the operations *intersection* and $cond - int$ can be derived using the proofs for the previous operations.

G NP-hardness for MTL($\mathbf{G}_I, \vee, \neg$)

To prove the NP-hardness of SYNTL$_d$ for MTL($\mathbf{G}_I, \vee, \neg$), we establish the NP-hardness of an easier problem where the future-reach bound is relaxed, which is the following: given a sample $\mathcal{S}$ and a size bound B, does there exist a formula φ in MTL($\mathbf{G}_I, \vee, \neg$) such that φ is $\mathbf{G}$-sep for $\mathcal{S}$, and $|\varphi| \leq B$? Towards this, we show a polynomial time reduction from the *hitting set* problem, a classical NP-complete problem in the literature. To this end, let us first define the hitting set decision problem: given $C_1, \ldots, C_n$ subsets of $[1, \ell]$ and $k \in \mathbb{N}$, does there exist H subset of $[1, \ell]$ of size at least k such that for every $j \in [1, n]$ we have $H \cap C_j \neq \emptyset$. In that case, we say that H is a hitting set.

We construct a reduction from the hitting set problem. Let $C_1, \ldots, C_n$ subsets of $[1, \ell]$ and $k \in \mathbb{N}$. Let us consider the set of propositions to be $\mathcal{P} = \{p_0, p_1, \ldots, p_\ell\}$. We construct the sample $\mathcal{S} = (P, N)$ with $T = l + 1$ as follows: for each $j \in [1, n]$ we let $[1, \ell] \setminus C_j = \{a_{j,1} < \cdots < a_{j,m_j}\}$, and define a positive signal prefix of the form

$$u_j = 0 : \{p_0\}; a_{j,1} : \{p_{a_{j,1}}\}; \ldots; a_{j,m_j} : \{p_{a_{j,m_j}}\}.$$

Let $P = \{u_1, \ldots, u_n\}$ be the set of all the positive signal prefixes. There is a single negative signal prefix, which is of the form

$$v = 0 : \{p_0\} \; ; 1 : \{p_1\} \; ; 2 : \{p_2\} \; ; \ldots \; ; \ell : \{p_\ell\}.$$

that means in v, only proposition p_i is true. We let N denote the singleton set containing v at time interval $[i, i+1)$.

We claim that there exists a hitting set of size at most k if and only if there exists a formula in MTL($\mathbf{G}_I, \vee, \neg$) of size at most $3k-1$ that is globally separating for $\mathcal{S}$, i.e., satisfies u_i's at all time-points and does not satisfy v at some time-point.

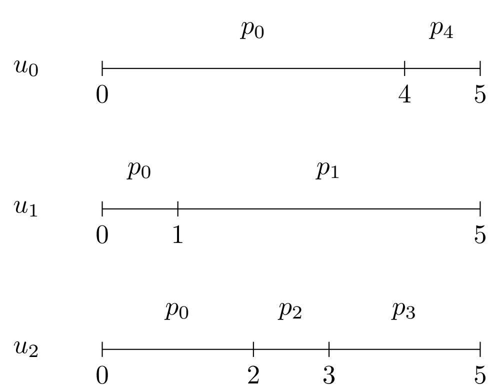
Let $H = \{c_1, \ldots, c_k\}$ be a hitting set of size k with $c_1 < c_2 < \cdots < c_k$, we construct the formula

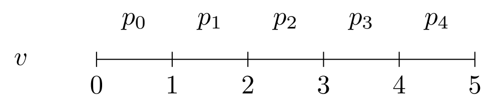
$$\varphi = (\neg p_{c_1} \vee \mathbf{G}_{[0,\ell]}(\cdots \vee \mathbf{G}_{[0,\ell]} \neg p_{c_k}))$$

We argue that φ globally separates $u_1 \ldots u_n$ from v and has size $3k-1$. Indeed, the fact that H is a hitting set means that for every $j \in [1, n]$, there exists i such that $c_i \in H$. This implies that u_j satisfies $\mathbf{G}_{[0,\ell]} \neg p_{c_i}$ globally, hence φ as well. Also, v does not satisfy φ at position c_1.

Conversely, let φ be a $\mathbf{G}$-sep formula in MTL($\mathbf{G}_I, \vee, \neg$) of size $3k-1$. Following [25], we can assume that φ is of the form above. Because it does not satisfy v, we have $c_1 < c_2 < \cdots < c_k$. We let $H = \{c_1, \ldots, c_k\}$, and argue that H is a hitting set. To prove that H is a hitting set, we need to prove that, for every $j \in [1, n]$, we have $H \cap C_j \neq \emptyset$. Now as φ is $\mathbf{G}$-sep, for every $j \in [1, n]$, we have $(u_j, t) \models \varphi$ for all $t \in [0, \ell]$. Then there exists a c_i that does not appear in u_j, implying the fact that $c_i \in H \cap C_j$ by the construction of u_j.

We illustrate the above idea via an example. Let l be four and $C_1, C_2, C_3 \subseteq [1, 4]$ such that, $C_1 = \{1, 2, 3\}, C_2 = \{2, 3, 4\}$ and $C_3 = \{1, 4\}$. Then, we construct the sample $\mathcal{S}$ with $T = 5$ as follows:





where u_1, u_2, u_3 are positive prefixes and v is the negative prefix. Now note that a hitting set for C_1, C_2, C_3 is $H = \{2, 4\}$ such that, $|H| = 2$. Then the corresponding $\mathbf{G}$-sep formula for $\mathcal{S}$ is $\varphi = \neg p_2 \vee \mathbf{G}_{[0,4]}\, \neg p_4$.