MLTL Multi-type (MLTLM): A Logic for Reasoning about Signals of Different Types *

Gokul Hariharan[1,*[0000−0002−3447−2183]],
Brian Kempa[1[0000−0003−2239−4218]], Tichakorn Wongpiromsarn[1[0000−0002−3977−122X]],
Phillip H. Jones[1[0000−0002−8220−7552]], and Kristin Y. Rozier[1[0000−0002−6718−2828]]

Iowa State University
*gokul@iastate.edu, bckempa@iastate.edu,
nok@iastate.edu, phjones@iastate.edu, kyrozier@iastate.edu

Abstract. Modern cyber-physical systems (CPS) operate in complex systems of systems that must seamlessly work together to control safety- or mission-critical functions. Capturing specifications in a logic like LTL enables verification and validation of CPS requirements, yet an LTL formula specification can imply unrealistic assumptions, such as that all signals populating the variables in the formula are of type Boolean and agree on a standard time step. To achieve formal verification of CPS systems of systems, we need to write validate-able requirements that reason over (sub-)system signals of different types, such as signals with different timescales, or levels of abstraction, or signals with complex relationships to each other that populate variables in the same formula. Validation includes both transparency for human validation and tractability for automated validation, e.g., since CPS often run on resource-limited embedded systems. Specifications for correctness of numerical algorithms for CPS need to be able to describe global properties with precise representations of local components. Therefore, we introduce Mission-time Linear Temporal Logic Multi-type (MLTLM), a logic building on MLTL, to enable writing clear, formal requirements over finite input signals (e.g., sensor signals, local computations) of different types, cleanly coordinating the temporal logic and signal relationship considerations without significantly increasing the complexity of logical analysis, e.g., model checking, satisfiability, runtime verification (RV). We explore the common scenario of CPS systems of systems operating over different timescales, including a detailed analysis with a publicly-available implementation of MLTLM.
We contribute: (1) the definition and semantics of MLTLM, a lightweight extension of MLTL allowing a single temporal formula over variables of multiple types; (2) the construction and use of an MLTLM fragment for time-granularity, with proof of the language's expressive power; and (3) the design and empirical study of an MLTLM runtime engine suitable for real-time execution on embedded hardware.

1 Introduction

Design and verification of safety-critical systems, such as aircraft, spacecraft, robots, and automated vehicles, requires precise, unambiguous specifications that enable automated reasoning such as model checking, synthesis, requirements debugging, runtime verification (RV), and checking for satisfiability, reachability, realizability, vacuity, and other important properties

* Artifacts for reproducibility appear at: http://laboratory.temporallogic.org/research/NSV2022. Funded in part by NSF:CPS Award #2038903, NSF:CAREER Award #1664356, and NASA Cooperative Agreement Grant #80NSSC21M0121

of system requirements. Modern, cyber-physical systems-of-systems present a unique challenge for specification, and consequently for scalable verification and validation, due to their distributed and hierarchical nature. To seed automated reasoning for CPS systems-of-systems, we need to be able to seamlessly construct global properties combining local phenomena and coordinate requirements for numerical computations like supervision and signal processing over data and variables of different types and sampling frequencies.

Due to the popularity of timelines in operational concepts for CPS systems-of-systems LTL provides an intuitive way to precisely specify system requirements. The relative computational efficiency of automated reasoning (e.g., model checking, satisfiability checking) adds to the appeal of LTL as a specification logic. Since CPS specifications most often need to describe finite missions with referenceable time steps, variations of LTL over finite signals (sometimes also called "traces") emerged with intervals on the temporal operators. Variations on Metric Temporal Logic (MTL)[28], such as Signal Temporal Logic (STL)[16] and Mission-time Linear Temporal Logic (MLTL)[29,23] vary widely in the types of finite bounds they introduce on LTL's temporal operators and the complexity of automated reasoning (e.g., model checking, satisfiability checking) over these logics. MLTL, which adds finite, closed, integer bounds on LTL's temporal operators, has emerged as a popular specification logic for complex CPS systems-of-systems such as the NASA Lunar Gateway Vehicle System Manager [12], and a JAXA autonomous satellite mission [27]; see [24] for a collection of MLTL patterns over a weather balloon, automated air traffic management system, sounding rocket, and satellite. Again, we see the selection of MLTL center on the balance of expressiveness with computational efficiency; MLTL efficiently reduces to LTL [23] and recent work has contributed very efficient, flight-certifiable, encodings of MLTL for runtime verification in resource-limited embedded hardware [20].

However, realistic requirements for CPS systems-of-systems need to combine variables of different types in the same requirement. For example, a requirement specified as an LTL formula may implicitly presume that the input signals populating its atomic propositions share a common notion of a time step. But we struggle to write a single formula to describe a global property about a system where different sub-systems operate at different times, or, more generally, over different types with a non-obvious comparison function. For one example, this problem emerges when we try to specify global safety properties of deep-space-exploring craft. One subsystem of the spacecraft may regulate monthly cycles to wake from hibernation and execute course corrections whereas another subsystem may operate on the nanosecond frequency to make hyper-sensitive adjustments; it is not obvious how to efficiently reason about these in the same formula. Numerical computations and reasoning on embedded hardware are essential features of CPS, yet they present even more challenges for combining multiple types in a single specification. During long, complex, numerical simulations having a monitor verify statistical patterns in generated data will help detect errors or non-convergence in the early phases, saving computational resources, manual inspection and inefficient postmortem analysis [14,15].

Previous works provide some options for special cases of this problem, with significant complexity drawbacks. These largely center on two philosophies: higher-order logics reasoning over sets of formulas (instead of one formula combining different types), and annotations to deal with multiple time granularities across formula variables, though not necessarily other combinations of different types. Examples of distributed sets of specifications count on locally evaluating sub-system-level synchronous [6] or asynchronous [5,26] signals; this set can

coordinate through a global formula evaluated over the local formulas [6]. HyperLTL focuses on specifications over sets of formulas over signals of the same type [9], oppositely from this work where we focus on constructing single formulas that seamlessly reason over signals of different types.

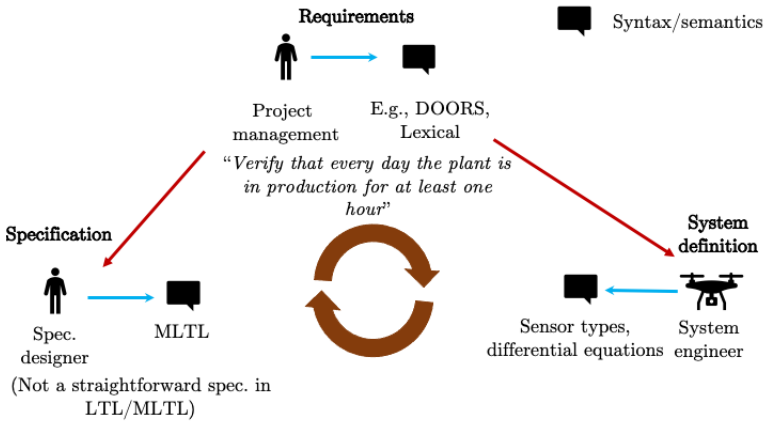
The particular instance of different types in the form of input signals over different time granularities that comprise parts of the same, single temporal logic specification arises frequently in CPS; see [17] for a survey. Most previous works focus on developing well expressible languages to define temporally distributed specifications precisely. Again, this often comes with higher-order reasoning (see for example [18]) and complexity penalties; e.g., [10] introduces the notion of temporal universes and uses a set-theory representation of different timescales to abstract notions of time granularities. Propositional Interval Temporal Logic (PITL) adds chop (";") and project operators to LTL to increase expressivity for time granularities over infinite signals; another variation adds temporal relations like "just before" [11]. First-Order Theory (FOT) enables writing time-granular specifications to account for continuous-in-time events and relate them to discretized-in-time representations [3]. Other methods include using automata to represent time-granularity [21,22] and using spider-diagram representations for time-granular specifications [7], and a two-dimensional metric temporal logic that can be potentially used to represent time granularities [4]. Table 1 collects this related work.

Time-granular logic	**Syntax elements**	**Ref.**
PITL	empty, proj, ";", $\Box$, $\Diamond$	[8]
Non standard FOT	$\forall$, $<_e$, $<_w$, $<_1$, $\exists$	[3]
ITL	$<$, m, O, s, f etc.	[1]
Euzenate's extension	6×6 table of operators	[11]
Automata representation	Automata	[21,22]
Spider diagrams	Spider diagrams	[7]
2D MTL internal eternal	L_i, L_e etc.	[4]
Monodic SOL	Layered representation of FOT	[18]

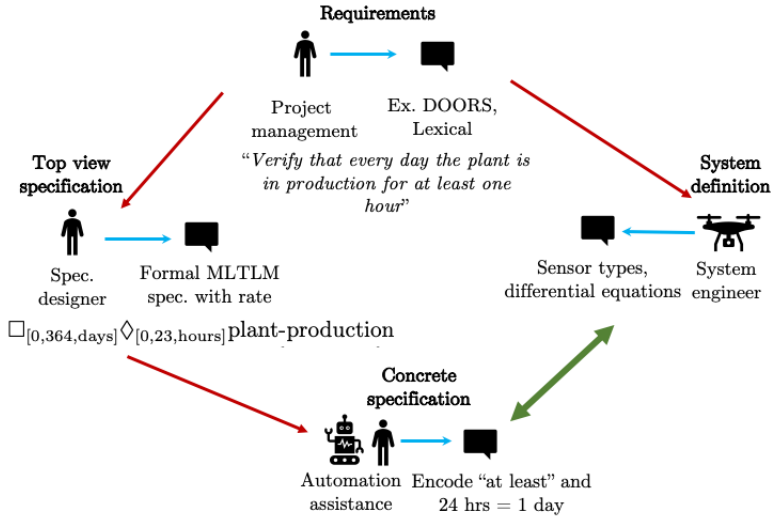
Table 1: Various time-granular specification languages and their syntax elements.

None of the existing solutions enable directly and intuitively specifying linear temporal logic over finite signals containing different types. We need a logic designed for this use, that enables direct specification of common CPS requirements, e.g., for supervision or signal processing, without kludgey syntax that makes correct specifications hard to write, unintuitive constructions that make specifications hard for CPS designers to validate, or introducing complexity blow-ups that make verification techniques like model checking or runtime verification intractable. Therefore, we build upon the popular logic MLTL to create MLTLM, a logic for intuitively and directly expressing bounded temporal logic formulas whose variables may be of different types, including different time granularities. The syntax of MLTLM matches that of MLTL except for the single addition of a signal-type label on each temporal operator to signify the output type of that operator. Figure 1 depicts an example MLTLM specification workflow.

We contribute: (1) a formal definition for the logic MLTLM (Mission-time Linear Temporal Logic Multi-type), including syntax and semantics (Sec. 3); (2) a translation of MLTLM to MLTL with a proof of correctness, enabling use of existing MLTL automated reasoning engine (Sec. 4.1); (3) an open-source implementation of a direct encoding of MLTLM for



(a) Workflow when using LTL/MLTL



(b) Envisioned workflow using MLTLM

Fig. 1: Iteration workflow for CPS runtime verification of project requirements describing the system and specification in simple lexical language. (a) Traditionally, modifications to the system or specification at any level restarts the cycle. (b) We propose that project management first verify a top view specification in a simple syntax while iteration of the detailed specification is contained to system engineering. The automated assistant will provide hints on suggesting the right projection between types.

runtime verification, released as an extension of the flight-certifiable R2U2 engine (Sec. 4.2). We choose R2U2 because it is currently the only runtime verification engine that enables real-time analysis of complex algorithms, such as those for numerical software verification, in real time on embedded hardware [30,20].

Sec. 2 gives a prelude to the conventional single type temporal logic, MLTL, and gives background on R2U2 – an industry-used runtime verification engine for CPS that we will build upon to monitor MLTLM specifications. Sec. 3 defines our new logic, MLTLM, providing semantics, examples, properties, and use-cases. Sec. 4 discusses comparisons to a single signal-type logic, and optimization opportunities for automated reasoning using MLTLM. Finally, Sec. 5 discusses conclusions and scope for future work.

2 Preliminaries

This section formalizes signals and trajectories, overviews MLTL which is extended into MLTLM in Sec. 3, and R2U2 which is adapted to monitor MLTLM in Sec. 4.2.

2.1 Signals and Trajectories

Definition 1. *(Signal) A signal σ over an atomic proposition p is defined as the finite sequence $\sigma = a_0, a_1, \ldots$ where $\sigma[i] = a_i \in \{\mathsf{true}, \mathsf{false}\}$ indicates whether p holds at the discrete time instance i. All signals have a type, written $\sigma^{\mathbb{A}}$ for a signal σ with type $\mathbb{A}$.*

Definition 2. *(Trajectory) A trajectory π over atomic propositions $p_0, \ldots, p_n$ is a set of signals, i.e., $\pi = \{\sigma_0, \sigma_1, \ldots, \sigma_n\}$ where σ_i is a signal over p_i. $\pi_p^{\mathbb{A}}[i]$ refers to the ith value of the signal of type $\mathbb{A}$ over atomic proposition p in π.*

In Sec. 3, we impose that binary logical operators can only operate on signals of the same type. We assume that types represent properties such as frequency that are homogeneous across a type. Related work in linear temporal logic use "traces" or "computations" [2,9], which is typically described as a sequence of sets of atomic propositions. In contrast, we generalize "traces" by allowing member signals to be of different types and call them collectively as a trajectory.

2.2 MLTL

MLTL is a variant of LTL [2] on finite signals with closed temporal bounds [29,30] on natural numbers.

Definition 3. *(MLTL Syntax [29]) The syntax of an MLTL formula φ over a set of atomic propositions $\mathcal{AP}$ is recursively defined as:*

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

where $p \in \mathcal{AP}$, φ_1 and φ_2 are MLTL formulas, $I := [lb, ub]$ is a closed interval bound, such that lb and ub are natural numbers such that $lb \leq ub$.

Abstract Syntax Tree (AST) The AST representation of an MLTL formula has nodes of logical operators and leaves of atomic propositions connected to represent the recursive structure of the expression from Def. 3.

Definition 4. *(MLTL Semantics [29]) The evaluation of an MLTL formula φ on a trajectory π where all signals have uniform type produces a signal σ defined recursively on the signals σ_1 and σ_2 representing the evaluation of its child subformula(s) φ_1 and φ_2 respectively.*

$$\sigma[i] := \begin{cases} \pi_p[i] & \text{if } \varphi = p \\ \neg\sigma_1[i] & \text{if } \varphi = \neg\varphi_1 \\ \sigma_1[i] \wedge \sigma_2[i] & \text{if } \varphi = \varphi_1 \wedge \varphi_2 \\ \mathsf{true} \text{ iff } |\sigma_1|, |\sigma_2| > (i+ub) \text{ and} & \\ \quad \exists j \in [i+lb, i+ub] \text{ such that } \sigma_2[j] = \mathsf{true} & \text{if } \varphi = \varphi_1 \mathcal{U}_{[lb,ub]} \varphi_2 \\ \quad \text{and } \forall k < j \text{ where } k \in [i+lb, i+ub], \sigma_1[k] = \mathsf{true} & \end{cases}$$

Other common operators are defined via equivalences, i. e., false $\Leftrightarrow \neg$true, future $\Diamond_I \varphi \Leftrightarrow \mathsf{true}\, \mathcal{U}_I\, \varphi$, globally $\Box_I \varphi \Leftrightarrow \neg(\Diamond_I \neg\varphi)$, and next $\bigcirc\varphi \Leftrightarrow \Box_{[1,1]}\varphi$.

2.3 R2U2

The *Realizable, Responsive, Unobtrusive Unit*[1] (R2U2) is an MLTL based RV engine for flight mission systems [29] used in robotics [20], NASA drone aircraft [19,31], and is being evaluated for use on the Lunar Gateway space station [12]. R2U2 is *Realizable*: implemented on real hardware, *Responsive*: reports specification violation immediately, and *Unobtrusive:* uses existing data sources instead of modifying the system to add instrumentation. R2U2 features specification reconfiguration and real-time performance with guaranteed memory bounds to better support the needs of flight systems. We have developed our MLTLM verification engine upon R2U2, which is an open-source RV engine with well-documented industrial use to provide users with a seamless move to multi-type logic. The R2U2 based MLTLM verification engine we develop upholds all existing guarantees of R2U2.

3 Mission-time Temporal Logic Multi-type (MLTLM)

We develop the foundations of MLTLM in this section. MLTLM is a lightweight extension to MLTL that enables temporal reasoning over system trajectories composed of signals of different types.

Definition 5. *(MLTLM Syntax) The syntax of an MLTLM formula φ over a set of atomic propositions $\mathcal{AP}$ is recursively defined as:*

$$\varphi := \mathsf{true} \mid p \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \mathcal{U}_J \varphi_2$$

where $p \in \mathcal{AP}$, φ_1 and φ_2 are MLTLM formulas, and $J := [lb, ub, \mathbb{A}]$ is a finite interval bound such that lb and ub are natural numbers, $lb \leq ub < \infty$, and $\mathbb{A}$ is a label indicating the signal type over which an MLTLM temporal operator evaluates.

Notably, MLTLM syntax is MLTL syntax with signal types associated with temporal operators.

[1] r2u2.temporallogic.org

Definition 6. *(MLTLM Semantics) The evaluation of an MLTLM formula φ on a trajectory π produces a signal σ of type $\mathbb{A}$ defined recursively on the signals σ_1 and σ_2 representing the evaluation of its child subformula(s) φ_1 and φ_2 respectively.*

$$\sigma^{\mathbb{A}}[i] := \begin{cases} \pi_p^{\mathbb{A}}[i] & \text{if } \varphi = p \\ \neg\sigma_1^{\mathbb{A}}[i] & \text{if } \varphi = \neg\varphi_1 \\ \sigma_1^{\mathbb{A}}[i] \wedge \sigma_2^{\mathbb{A}}[i] & \text{if } \varphi = \varphi_1 \wedge \varphi_2 \\ \sigma_1^{\mathbb{A}}[i..]\mathcal{U}_{[lb,ub]}\sigma_2^{\mathbb{A}}[i..] & \text{if } \varphi = \varphi_1 \mathcal{U}_{[lb,ub,\mathbb{A}]}\varphi_2 \end{cases}$$

where $\sigma[i..]$ is the subsequence of signal σ starting from discrete point i and all operators are evaluated according to the rules of Def. 4.

Note that when evaluating the fourth case in Def. 6, the signal types produced by subformulas φ_1 and φ_2 must be projected into signals of the type associated with the temporal operator. Additional common operators like implication, disjunction, and globally are constructed by standard equivalence relations as in MLTL, with all derived temporal operators inheriting the type specifier on their interval bounds. If the relationship between types can be expressed as a function that converts the type of signals, that function is called a projection.

Definition 7 (Projection). *The projection function $T_{\mathbb{A}}^{\mathbb{B}}(\sigma^{\mathbb{A}})$ takes the signal σ of type $\mathbb{A}$ and returns a new signal of type $\mathbb{B}$.*

We will examine several projection functions, however writing MLTLM formulas requires only assurance their existence, not their definition; this provides a separation of concerns we leverage to ease specification writing and linearize verification workflow. For example, consider a formula φ specifying that φ_1 should hold every hour for 10 hours, and φ_2 should hold every second for 100 seconds. In MLTLM φ could be written as $\square_{[0,9,\text{hour}]}\varphi_1 \wedge \square_{[0,99,\text{second}]}\varphi_2$. In MLTL, φ would need to be written assuming a monitor rate, say seconds, then specifier would write $\square_{[0,0]}\varphi_1 \wedge \square_{[3600,3600]}\varphi_1 \wedge \square_{[7200,7200]}\varphi_1 \wedge \cdots$ and $\square_{[0,99]}\varphi_2$. The formula is longer and embeds the relation between hours and seconds. If the specification must be evaluated at a monitor rate of minutes instead, the canonical encoding must be updated by the specification author as discussed in more detail in Sec. 1 (Fig. 1). In contrast, in MLTLM, the top view specification remains the same even in the face of implementation details like evaluation rate.

3.1 Equivalent MLTLM Formula for Every MLTL Formula

For a formula naming at most one type, all properties that hold in MLTL hold in MLTLM, i.e., $\Diamond_{[lb,ub,\mathbb{A}]}\varphi \Leftrightarrow \mathsf{true}\,\mathcal{U}_{[lb,ub,\mathbb{A}]}\,\varphi$, $\square_{[lb,ub,\mathbb{A}]}\varphi \Leftrightarrow \neg(\Diamond_{[lb,ub,\mathbb{A}]}\neg\varphi)$ and so on. The following claim expresses that formulas expressible in MLTL form a subset of formulas expressible in MLTLM. The claim attests that there is no loss in using MLTLM compared to MLTL. The transformation is simple, and the formula is, at worst, the same length, though potentially much shorter in MLTLM, as demonstrated in Sec. 4.3.

Claim. An equivalent MLTLM formula of the same length exists for every MLTL formula, and this translation is possible in constant time.

Proof. We can represent any MLTL formula as an MLTLM formula by appending a signal type to the interval bound of every temporal operator. This follows from the definition of

MLTLM. The formula length, being the total number of operators plus atomic propositions, is not affected by appending a type name to the temporal operators. Hence the resultant MLTLM formula is of the same length as the MLTL formula.

3.2 Evaluation of MLTLM Formula

Evaluation of an MLTLM formula on a trajectory requires signals for all atomic propositions. Evaluating an MLTLM formula naming at most one type over a trajectory is equivalent to evaluating MLTL formulas over a trajectory containing only the required signals.

With projection, a new signal of a different type can be derived from an existing signal in the trajectory. For example, the return of a high-rate sensor can be down-sampled to match the type of low rate sensor. This "derived signal" evaluation is where all signals are first projected to a common type before evaluation. Using signals, types, and projection, we can evaluate a formula with mixed types by considering each subformula to represent the signal of its own evaluation and projecting where necessary as explained further in the next section.

Critically, operator semantics are defined for any type, but only when the input(s) and output types match. The inputs to the temporal logic operator must be projected to the written type in the operator's bound if needed. Fundamentally, MLTLM formulas represent a directed graph of data flow between domains of MLTL connected by projections.

Tutorial Example Application of the MLTLM Semantics (Def. 6) To help clarify how the semantics in Def. 6 are applied, we consider the formula $\square_{[1,2,\mathbb{B}]}(\square_{[2,4,\mathbb{A}]}p)$. The global ($\square$) operator is a common unary temporal operator derived from the definition of $\mathcal{U}$ by the equivalence relation $\square_{[lb,ub,\mathbb{A}]}\varphi \Leftrightarrow \neg(\mathsf{true}\ \mathcal{U}_{[lb,ub,\mathbb{A}]}\ \neg\varphi)$. This is the same as adding the following case to the MLTLM semantics:

$$\sigma[i] := \mathsf{true} \text{ iff } \sigma_1^{\mathbb{A}}[j] = \mathsf{true}\ \forall j \in [i+lb, i+ub], \qquad \text{if } \varphi = \square_{[lb,ub,\mathbb{A}]}\varphi_1.$$

Applying Def. 6, the evaluation of formula $\square_{[1,2,\mathbb{B}]}(\square_{[2,4,\mathbb{A}]}p)$ depends on the type of any known signals for p and the desired output type. Let us consider generating a signal of type $\mathbb{B}$ from the above formula, and that $\pi_p^{\mathbb{A}}$ is known for p. In Fig. 2a, the known signal for p, $\sigma_1^{\mathbb{A}}$, is input to $\square_{[2,4,\mathbb{A}]}$ whose satisfaction signal, $\sigma_2^{\mathbb{A}}$, is input to $\square_{[1,2,\mathbb{B}]}$, finally generating $\sigma_1^{\mathbb{B}}$ which meets the required output of type $\mathbb{B}$.

Now let us consider another case with the same formula where we need an output signal of type $\mathbb{C}$, and know $\pi_p^{\mathbb{B}}$. In Fig. 2b, evaluating the subformula $\square_{[2,4,\mathbb{A}]}p$ requires a signal for p in type $\mathbb{A}$ per the semantics, but we only know p in type $\mathbb{B}$. This implies a projection $T_{\mathbb{B}}^{\mathbb{A}}(\sigma_1^{\mathbb{B}}) = \sigma_1^{\mathbb{A}}$ before the result is input to $\square_{[2,4,\mathbb{A}]}$, generating $\sigma_2^{\mathbb{A}}$. Another type incompatibility arises between $\sigma_2^{\mathbb{A}}$ and $\square_{[1,2,\mathbb{B}]}$, so it is again (implicitly) projected to a type $\mathbb{B}$ through $T_{\mathbb{A}}^{\mathbb{B}}(\sigma_2^{\mathbb{A}}) = \sigma_2^{\mathbb{B}}$. Since the desired output type is $\mathbb{C}$, there is one last projection $T_{\mathbb{B}}^{\mathbb{C}}(\sigma_3^{\mathbb{B}}) = \sigma_1^{\mathbb{C}}$.

3.3 Examples of Projections

Earlier in Sec. 3 we defined an abstract projection (Def. 7). This section will consider a couple of useful projections and discuss some example specifications.

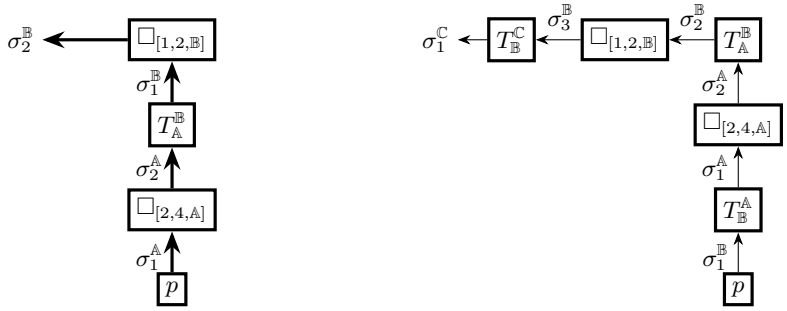


Fig. 2: Illustration of two possible evaluations of a given formula $\Box_{[1,2,\mathbb{B}]}(\Box_{[2,4,\mathbb{A}]}p)$

.

Definition 8. *(Modulo-Reduction Function) The function* $f_s : \sigma^{\mathcal{A}} \to \sigma^{\mathcal{B}}$ *implements the projection* $T_{\mathbb{A}}^{\mathbb{B}}(\sigma^{\mathbb{A}})$ *by modulo-reduction with positive integer stride* s *when:*

$$f_s(\sigma^{\mathbb{A}}) = \sigma^{\mathbb{B}} \text{ such that } \sigma^{\mathbb{B}}[i] = \sigma^{\mathbb{A}}[i \cdot s] \tag{1}$$

The modulo-reduction function outputs every sth value from the input signal.

Definition 9. *(Majority-Reduction Function) The function* $g_s : \sigma^{\mathcal{A}} \to \sigma^{\mathcal{B}}$*, implements the projection* $T_{\mathbb{A}}^{\mathbb{B}}(\sigma^{\mathbb{A}})$ *by majority-reduction with positive integer stride* s *when:*

$$g_s(\sigma^{\mathbb{A}}) = \sigma^{\mathbb{B}} \text{ such that}$$
$$\sigma^{\mathbb{B}}[i] = \begin{cases} \mathsf{true} & \text{if } \mathcal{N}_0(\{j \in [i \cdot s, (i+1) \cdot s] : (\sigma^{\mathbb{A}}[j] = \mathsf{true})\}) \geq \lfloor s/2 \rfloor \\ \mathsf{false} & \text{otherwise} \end{cases} \tag{2}$$

where $\mathcal{N}_0(\cdot)$ *is the set cardinality.*

The majority-reduction function outputs the majority value of every s values of the input signal.

3.4 Example Specifications Across Timescales

We consider a few example specifications taken from literature on time-granularities [17,25], and modify or extend them to the context of RV.

1. "Verify that John is present for 8 hours at a stretch each day for the next 6 days."
   This specification can be represented in MLTLM as:

$$\Box_{[0,5,\text{day}]}(\Diamond_{[0,16,\text{hour}]}\Box_{[0,7,\text{hour}]}\text{john-present}) \tag{3}$$

   The specification says that eventually, from the 0th to the 16th hour, there exists an hour such that John is present from the 0th to the 7th hour. The eventually operator has a time going from 0 to 16, and the global operator from 0 to 7, and the total time adds to 0 to 23 hours, which is a 24 hour period (a day).
   This specification is verified on a daily basis, based on the type of the root node of the AST for Eq. (3), the $\Box_{[0,5,\text{day}]}$. The day type must be projected from the hour type used by the subformula. The satisfaction of the formula depends on the projection used to go from the hourly type to the daily type.

2. "Verify that for at least one day in a year the plant works every hour"

$$\Diamond_{[0,364,\text{day}]}\Box_{[0,23,\text{hour}]}\text{plant-works}$$

3. "Verify that every day the plant is in production for some hours"

$$\Box_{[0,364,\text{day}]}\Diamond_{[0,23,\text{hour}]}\text{plant-production}$$

4. "Verify that the plant is monitored by the remote system every minute of every hour for the next 24 hours"

$$\Box_{[0,23,\text{hour}]}\Box_{[0,59,\text{minute}]}\text{system-monitored}$$

5. "On all days of the year, the plant works for at least 12 hours"
   We represent this in MLTLM using the majority-reduction function (Def. 9), with $\mathbb{A} \equiv$ hour and $\mathbb{B} \equiv$ day as

$$\Box_{[0,364,\text{day}]}\Box_{[0,0,\text{hour}]}\text{plant-works}$$

6. "Verify that the system deviates at most for a minute every hour for the next 24 hours."
   We can represent this in MLTLM by modifying the cardinality relation in Eq. (2) to ">1" and using the resultant function with $\mathbb{A} \equiv$ hour and $\mathbb{B} \equiv$ day as the projection,

$$\Box_{[0,23,\text{hour}]}\Box_{[0,0,\text{minute}]}\text{system-deviates}$$

4 Equisatisfiable Formula in MLTL and an Implementation of an MLTLM monitor with the Modulo-Reduction Projection

The previous section introduced MLTLM and demonstrated how it could simplify the workflow and specifications across timescales. We now illustrate space and time optimization possibilities by implementing an MLTLM RV engine. The generic syntax and semantics of MLTLM separates the specification from the signal type, i.e., the specification remains the same irrespective of the signal type. It is apparent from the semantics (Def. 6) that the output signal type is determined only in the fourth case with the temporal operator. For example, the formula $p \wedge q$ represents multiple output signal types depending on the trajectory types used for p and q, whereas the formula $\Box_{[0,0,\mathbb{A}]}(p \wedge q)$ has a single output type $\mathbb{A}$ irrespective of the trajectory types used for p and q. An implementation needs a single output type, and hence we consider a subset of MLTLM formulas that have a temporal operator at the root of the AST, and assume that the type on the root temporal operator is the desired output type.

Furthermore, to make the evaluation of an MLTLM formula complete, two more ingredients are essential, (a) the placement of projections in the AST of an MLTLM formula and (b) defined projections between type signals. Consider the MTLTM formula, $\Box_{[0,0,\mathbb{A}]}(p \wedge q)$. Let us assume that only a signal of type $\mathbb{B}$ is available from p and a signal of type $\mathbb{C}$ is available from q, as denoted in Fig. 3a. From the semantics Def. 6, it is clear that a conjunction is allowed only between signals of the same type, which implies that there are implicit projections to match signal types in the conjunction as shown in Fig. 3b.

We have two (out of many) options here to match types, (a) to project to a common signal type $\mathbb{D}$ at the conjunction, and then to a type $\mathbb{A}$ to match type in $\square_{[0,0,\mathbb{A}]}$ (Fig. 3b), and (b) place a projection to type $\mathbb{A}$ at the conjunction, then a second projection is not needed to match types in $\square_{[0,0,\mathbb{A}]}$ (Fig. 3c). While option (a) is of interest in the broader scope of applications with MLTLM like signal processing, option (b) is the situation with the minimal number of projections. *The generalization for this minimal projection placement is to impose that signals are projected to the type of the closest ancestor node with a type*. All nodes in the unique path connecting a node to the root of the AST are ancestor nodes of the node (the node inclusive). In this example, the closest ancestor of the conjunction is $\square_{[0,0,\mathbb{A}]}$ whose type is $\mathbb{A}$. We further assume that all such projections exist to evaluate a formula.

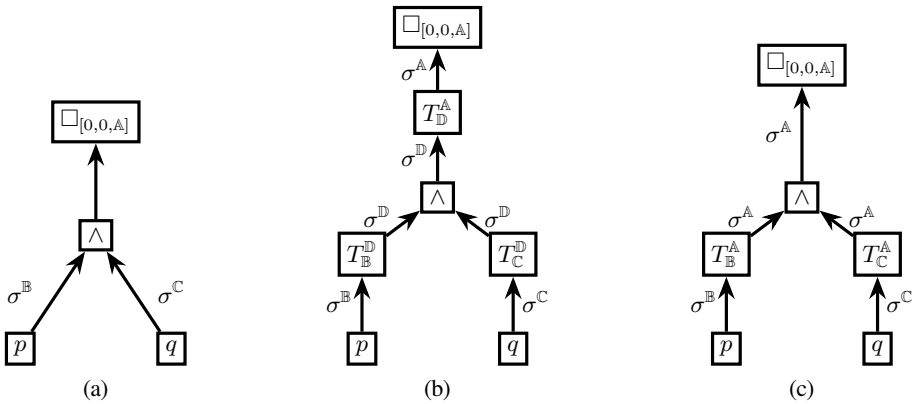


Fig. 3: The evaluation of an MLTLM formula depends on the placement of projections to match types in binary operators.

We consider only the modulo-reduction function (Def. 8) as it is not possible to cover all scenarios in this paper. We develop a theory to derive equisatisfiable MLTL formula for an MLTLM formula with a class of logical projections and then develop a translator based on it with the modulo-reduction projection (Def. 8). We then compare the memory and time needed to evaluate formulas using MLTLM and MLTL. In summary, we find that MLTLM reflects on profound savings in memory compared to its closest single-type logic. The results presented herein are only preliminary observations of optimization possibilities using MLTLM.

4.1 The Translator

Theorem 1. *(Expressive Equivalence of MLTL and MLTLM with Logical Projections) Let F be a projection expressible in MLTL, then F is a logical projection. Let $\mathbb{A}$ be a type and ψ be an MLTL formula that outputs signals of type $\mathbb{A}$. For every MLTLM formula φ such that for every type $\mathbb{t}$ in φ there exists a chain of logical projections from $\mathbb{t}$ to $\mathbb{A}$, the signal generated by φ is equivalent to another signal generated by ψ.*

Proof Sketch. The full proof is available in supplementary material posted online[2]. We give an example sketch over two signal types $\mathbb{B}$ and $\mathbb{C}$ related by a logical projection $F(\sigma^{\mathbb{B}}) = \sigma^{\mathbb{C}}$. We use the semantics of MLTLM (Def. 6) to prove by induction on the structure of the formula that any MLTLM formula that has temporal operators with both types $\mathbb{B}$ and $\mathbb{C}$ can be reduced to an equisatisfiable formula all of whose temporal operators are of type $\mathbb{B}$. An MLTLM formula of a single type can be reduced to an MLTL formula by merely removing the type from the formula.

We complete the proof by assuming that a formula of the form $\varphi = \varphi_1 \mathcal{U}_{[lb,ub,\mathbb{C}]} \varphi_2$ can be equivalently expressed with type $\mathbb{B}$ in its AST root using the logical projection. For example, we can show that the modulo-reduction projection (Def. 8) can be equivalently expressed as an MLTL formula without a projection to type $\mathbb{C}$ using the function $p(\varphi)$ where

$$\begin{aligned} p(\varphi) = \bigcirc_{\mathbb{B}}^{lb} (& \varphi_2 \\ & \vee (\varphi_1 \wedge \bigcirc_{\mathbb{B}}^{s} \varphi_2) \\ & \vee (\varphi_1 \wedge \bigcirc_{\mathbb{B}}^{s} \varphi_1 \wedge \bigcirc_{\mathbb{B}}^{2s} \varphi_2) \\ & \vee (\varphi_1 \wedge \bigcirc_{\mathbb{B}}^{s} \varphi_1 \wedge \bigcirc_{\mathbb{B}}^{2s} \varphi_1 \wedge \bigcirc_{\mathbb{B}}^{3s} \varphi_2) \\ & \quad \vdots \\ & \vee (\varphi_1 \wedge \bigcirc_{\mathbb{B}}^{s} \varphi_1 \wedge \bigcirc_{\mathbb{B}}^{2s} \varphi_1 \wedge \bigcirc_{\mathbb{B}}^{3s} \varphi_1 \cdots \wedge \bigcirc_{\mathbb{B}}^{(m-1)s} \varphi_1 \wedge \bigcirc_{\mathbb{B}}^{ms} \varphi_2) \ ), \end{aligned} \tag{4}$$

where $\bigcirc_{\mathbb{B}} = \square_{[1,1,\mathbb{B}]}$ is the next operator (and hence, $\bigcirc_{\mathbb{B}}^{s} = \square_{[s,s,\mathbb{B}]}$), and $m = \lfloor (ub - lb)/s \rfloor$. We then extend this to all cases in the semantics (Def. 6). Thus, any MLTLM formula φ with mixed signal types $\mathbb{B}$ and $\mathbb{C}$ has an equisatisfiable formula $q(\varphi)$, where the entire formula has a single type, $\mathbb{B}$, defined recursively by

$$q(\varphi) := \begin{cases} \varphi, & \text{if } \varphi \text{ has only one type, } \mathbb{B} \text{ in the entire formula,} \\ p(q(\varphi_1) \mathcal{U}_{[lb,ub,\mathbb{C}]} q(\varphi_2)), & \text{if } \varphi = \varphi_1 \mathcal{U}_{[lb,ub,\mathbb{C}]} \varphi_2, \\ \neg q(\varphi_1), & \text{if } \varphi = \neg \varphi_1, \\ q(\varphi_1) \wedge q(\varphi_2), & \text{if } \varphi = \varphi_1 \wedge \varphi_2. \end{cases} \tag{5}$$

It is straightforward to extend this analysis to multiple types that have transitional chain of connected projections. We showed that we could derive an equisatisfiable formula verifiable in the image type for any MLTLM formula when using the logical projections. We implement a translator from MLTLM to MLTL using the theory discussed.

We developed three translators from MLTLM to MLTL based on the recursive formula Eq. 5. The three translators are based on succinct and expanded versions of Eq. 4 the most succinct (to our best capability) being translator 3, and the most expanded being translator 1. The translator's details and proof of correctness will be reported elsewhere in the interest of space. We confirm, however, that verdicts from the three translators on a well-established MLTL engine (R2U2 [20]) and its extended MLTLM monitor developed by us (discussed in Sec. 4.2) produce consistent outputs for the same inputs with more than 70 randomly generated formulas.

[2] http://laboratory.temporallogic.org/research/NSV2022

4.2 An Efficient MLTLM Engine

We implement an RV engine for specifications in MLTLM on top of R2U2 (see Sec. 2, and [20]). Certain notes on how specifications are written out for verification using R2U2 are relegated to supplementary material online[3]. We skip the details of the implementation in the interest of space and report it elsewhere. We summarize the implementation briefly.

As we mentioned many times in this article, if a sensor data is only of interest every hour, then the second-to-second information can be dropped out; and the modulo-reduction function (Def. 8) does this operation. The MLTLM engine has added projection operators (see Def. 7) at appropriate places according to the semantics of MLTLM (Def. 6) using the closest ancestor type projection discussion in Sec. 4 (Fig. 3). The modulo-reduction projection operator drops the appropriate signal values not needed in evaluating a formula and reports the output signal type corresponding to the type in the root of the AST.

4.3 Optimization Results

In Sec. 3.1 we showed that every MLTL formula could be expressed in MLTLM in the same length. This section analyzes how long are the intuitive translations to MLTL compared to MLTLM. We do not claim rigorous proof on the shortest possible formula length but rather compare the most intuitive and succinct translations. We note that the translations contain expressions of the form (see Eq. 4),

$$\varphi_1 \wedge \bigcirc_{\mathbb{B}}^{s} \varphi_1 \wedge \bigcirc_{\mathbb{B}}^{2s} \varphi_1 \wedge \bigcirc_{\mathbb{B}}^{3s} \varphi_2,$$

which to the best of our knowledge cannot be made any shorter in LTL and MLTL [32].

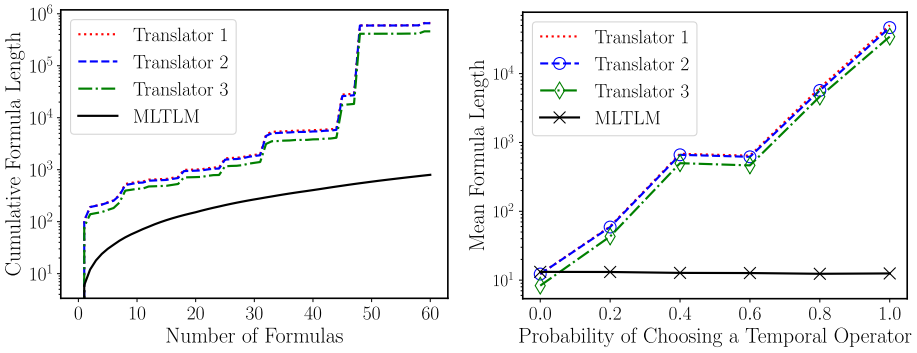
We randomly draw MLTLM formulas using the procedure in [13] and plot the length of the shortest intuitive MLTL translations. The randomly drawn formulas are parametrized by the probability of drawing a temporal operator (P), the maximum difference between the lower and upper bounds (M), and the maximum signal length (T). We will fix $M = T = 6$ in our study here. Furthermore, the memory and time also depends on stride, s of the modulo-reduction function (see Eq. (1)). In real systems specifications may reason over say, seconds, minutes, hours and days, which correspond to $s = 60$, and 24. However, as we mentioned previously, we are reporting preliminary observations on optimization possibilities, and we use four signal types, which we will call $\mathbb{A}$, $\mathbb{B}$, $\mathbb{C}$ and $\mathbb{D}$, where (see Eq. (1) for $f_s(\sigma)$), with

$$f_2(\sigma^{\mathbb{A}}) = \sigma^{\mathbb{B}}, \quad f_3(\sigma^{\mathbb{B}}) = \sigma^{\mathbb{C}}, \quad f_4(\sigma^{\mathbb{C}}) = \sigma^{\mathbb{D}},$$

with stride lengths $s = 2,3,4$. Note that the memory savings will be much larger with a larger stride like $s = 60$ (e.g., from second to minute).

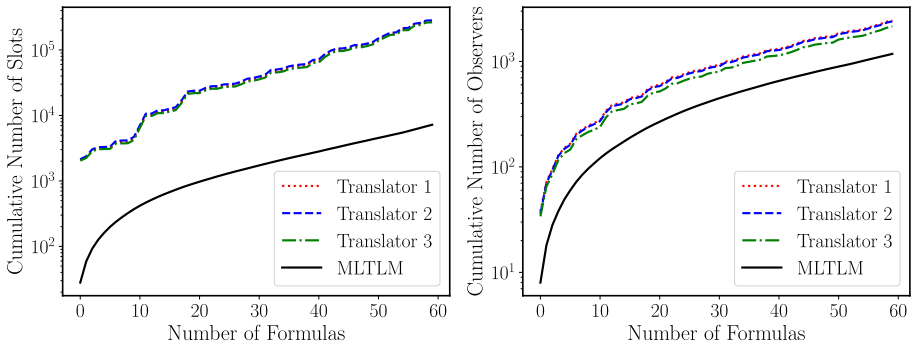
Fig. 4a shows the cumulative formula length with randomly drawn formulas. At $P = 0.5$, the three translators produce MLTL formulas of nearly the same length (the dotted, dashed, and dashed and dotted lines). However, Translator 3 performs slightly better with shorter formula lengths. In contrast, the formula lengths of the MLTLM formulas are substantially smaller. Hence, there is no loss in using MLTLM in comparison to MLTL (see Sec. 3), but using MLTLM may result in much smaller and more intuitive formulas depending on the projection function.

[3] http://laboratory.temporallogic.org/research/NSV2022



(a) (b)

Fig. 4: (a) Cumulative formula length with the number of randomly drawn formulas with $P = 0.5$, and (b) mean formula length against the probability of choosing a temporal operator.



(a) (b)

Fig. 5: Cumulative memory (left) and time (right) needed to verify random MLTLM formulas from the benchmark set in Sec. 4 using an equivalent MLTL formula (translator) compared to an MLTLM engine with a Modulo-Reduction projection operator in R2U2.

Fig. 4b shows the mean formula length (averaged over 60 random formulas) by varying the probability of the temporal operator. $P = 0$ corresponds to no temporal operators, and in that case, the translators and the MLTLM formula perform nearly equally well. This is expected – if the formula contains mere propositional logic, the formula should be independent of the temporal specification language. However, on close observation, the MLTLM formula at $P = 0$ is slightly longer. This is because in MLTL there is only one signal type, hence there is no need for a output signal type specifier, whereas in MLTLM, a proposition (say, $p \wedge q$, $p,q \in \mathcal{AP}$) represents a family of outputs of different types. We always use a temporal operator at the start of any formula (as in $\square_{[0,0,\mathbb{B}](p \wedge q)}$ in the place of $p \wedge q$), and this adds to excess length of an MLTLM formula compared to an MLTL formula with propositions. However, propositions like $p \wedge q$ are valid MLTLM formulas, but verification of the formula needs an output-type identifier.

On increasing the probability of choosing a temporal operator, the equisatisfiable formulas in MLTL become significantly longer owing to the expansion to the base type as discussed in Sec. 4.1. Fig. 5 shows the estimated resource and time requirements on hardware. The memory to evaluate a formula is statically assigned in R2U2 [20] as dynamic memory is often not permitted in flight software. Hence, we compare the amount of static memory that needs to be assigned for equisatisfiable formulas in MLTLM and (translated) MLTL (Fig. 5a). Similarly, the time taken for formula evaluation is directly proportional to the number of nodes created in the AST. We call the nodes in the AST as observers (as seen in the Y axis labels of Fig. 5b). We see that equisatisfiable formula require much lesser memory in MLTLM than MLTL (Fig. 5a). Similarly, the evaluation time is also much faster for MLTLM as it needs much lesser observers (Fig. 5b).

We end this section with a few remarks. We considered random formulas in this section, and they may not be true representatives of real specifications that may have different results on memory and time savings (Fig. 5). Nonetheless, the results show that there is great opportunity to have short intuitive formulas that encode timescales directly in the formula to simplify the workflow (Fig. 1), and in addition, an optimally configured RV engine for MLTLM is likely to have profound memory savings making it more suitable for resource constrained hardware.

5 Conclusion

Writing specifications naturally needs reasoning across multiple signal types, be it signals coming from different sensors at different rates, or belonging to observers in parallel universes (distributed systems), or having a mix of continuous and discrete signals (hybrid systems). We developed a multi-type logic to express such specifications, and then explored an application to time granularities. As discussed, this serves multiple purposes: 1) for the user, specifications are easy to write, 2) the theoretical satisfaction in different types is defined unambiguously, and 3) implementations can better utilize resources when compared with a single signal-type logic. Moreover, we expect that MLTLM will simplify the workflow by keeping the syntax simple and accessible, and postponing the nuances into the projection function. More importantly, MLTLM separates the specification from signal type. For example, let us suppose that a pressure sensor is changed in the Lunar Gateway, and it generates data in a different rate than the old sensor, or perhaps in a different unit like Pascals in the place of atmospheric pressure. Specifications for a single type logic would have to be changed to account for the signal type. MLTLM side-steps this process: The signal type will not affect the specification in any manner. In the future, we plan to have an automated assistant, that will allow a user to choose different projections to use for different contexts in specifications, (like “at least”, “at most”, “only once” etc.), and will also inform the user about the amount of memory he will need to dedicate/save on the hardware (the memory needed may vary based on the type of projection). This will allow the industrial verification community to seamlessly move to a time-granular logic. We will also consider human authored MLTLM specifications on real systems to get a better perspective on optimization opportunities. Lastly, the MLTLM monitor built upon R2U2 was validated across a regression suite of specifications and trajectories, but the current implementation can be improved to have tighter bounds on memory usage, which needs further investigation.

References

1. Allen, J.F., Hayes, P.J.: A common-sense theory of time. In: Proceedings of the 9th International Joint Conference on Artificial Intelligence - Volume 1. p. 528–531. IJCAI'85, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1985)
2. Baier, C., Katoen, J.P.: Principles of model checking. MIT press (2008)
3. Balbiani, P.: Time representation and temporal reasoning from the perspective of non-standard analysis. In: Proceedings of the Eleventh International Conference on Principles of Knowledge Representation and Reasoning. p. 695–704. KR'08, AAAI Press (2008)
4. Baratella, S., Masini, A.: A two-dimensional metric temporal logic. Mathematical Logic Quarterly **66**(1), 7–19 (2020). https://doi.org/https://doi.org/10.1002/malq.201700036
5. Bataineh, O., Rosenblum, D.S., Reynolds, M.: Efficient decentralized LTL monitoring framework using tableau technique **18**(5s) (2019)
6. Bauer, A., Falcone, Y.: Decentralised LTL monitoring. In: Giannakopoulou, D., Méry, D. (eds.) FM 2012: Formal Methods. pp. 85–100. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)
7. Bottoni, P., Fish, A.: Policy specifications with timed spider diagrams. In: 2011 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC). pp. 95–98 (2011). https://doi.org/10.1109/VLHCC.2011.6070385
8. Bowman, H., Thompson, S.: A decision procedure and complete axiomatization of finite interval temporal logic with projection. Journal of Logic and Computation **13**(2), 195–239 (2003). https://doi.org/10.1093/logcom/13.2.195
9. Clarkson, M.R., Finkbeiner, B., Koleini, M., Micinski, K.K., Rabe, M.N., Sánchez, C.: Temporal logics for hyperproperties. In: Abadi, M., Kremer, S. (eds.) Principles of Security and Trust. pp. 265–284. Springer Berlin Heidelberg, Berlin, Heidelberg (2014)
10. Clifford, J., Rao, A.: A simple, general structure for temporal domains (1986)
11. Cohen-Solal, Q., Bouzid, M., Niveau, A.: An algebra of granular temporal relations for qualitative reasoning. In: Proceedings of the 24th International Conference on Artificial Intelligence. p. 2869–2875. IJCAI'15, AAAI Press (2015)
12. Dabney, J.B., Badger, J.M., Rajagopal, P.: Adding a verification view for an autonomous real-time system architecture. In: AIAA Scitech 2021 Forum. p. 0566 (2021)
13. Daniele, M., Giunchiglia, F., Vardi, M.Y.: Improved automata generation for linear temporal logic. In: International Conference on Computer Aided Verification. pp. 249–260. Springer (1999)
14. Dinh, M.N., Abramson, D., Jin, C.: Runtime verification of scientific codes using statistics. Procedia Computer Science **80**, 1473–1484 (2016). https://doi.org/https://doi.org/10.1016/j.procs.2016.05.468, international Conference on Computational Science 2016, ICCS 2016, 6-8 June 2016, San Diego, California, USA
15. Dinh, M.N., Trung Vo, C., Abramson, D.: Tracking scientific simulation using online time-series modelling. In: 2020 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID). pp. 202–211 (May 2020). https://doi.org/10.1109/CCGrid49817.2020.00-73
16. Donzé, A.: On signal temporal logic. In: Legay, A., Bensalem, S. (eds.) Runtime Verification. pp. 382–383. Springer Berlin Heidelberg, Berlin, Heidelberg (2013)
17. Euzenat, J., Montanari, A.: Time granularity. Handbook of Temporal Reasoning in Artificial Intelligence (January 2005)
18. Franceschet, M., Montanari, A., Peron, A., Sciavicco, G.: Definability and decidability of binary predicates for time granularity. Journal of Applied Logic **4**(2), 168–191 (Jun 2006). https://doi.org/10.1016/j.jal.2005.06.004
19. Geist, J., Rozier, K.Y., Schumann, J.: Runtime Observer Pairs and Bayesian Network Reasoners On-board FPGAs: Flight-Certifiable System Health Management for Embedded Systems. In: Proceedings of the 14th International Conference on Runtime Verification (RV14). vol. 8734, pp. 215–230. Springer-Verlag (September 2014)

20. Kempa, B., Zhang, P., Jones, P.H., Zambreno, J., Rozier, K.Y.: Embedding online runtime verification for fault disambiguation on Robonaut2. In: Proceedings of the 18th International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS). Lecture Notes in Computer Science (LNCS), vol. 12288, pp. 196–214. Springer, Vienna, Austria (September 2020)
21. Lago, U.D., Montanari, A., Puppis, G.: Compact and tractable automaton-based representations of time granularities. Theoretical Computer Science **373**(1), 115–141 (2007). https://doi.org/https://doi.org/10.1016/j.tcs.2006.12.014
22. Lago, U.D., Montanari, A., Puppis, G.: On the equivalence of automaton-based representations of time granularities. In: 14th International Symposium on Temporal Representation and Reasoning (TIME'07). pp. 82–93 (2007). https://doi.org/10.1109/TIME.2007.56
23. Li, J., Vardi, M.Y., Rozier, K.Y.: Satisfiability checking for Mission-time LTL. In: Proceedings of 31st International Conference on Computer Aided Verification (CAV). LNCS, vol. 11562, pp. 3–22. Springer, New York, NY, USA (July 2019)
24. Luppen, Z., Jacks, M., Baughman, N., Hertz, B., Cutler, J., Lee, D.Y., Rozier, K.Y.: Elucidation and Analysis of Specification Patterns in Aerospace System Telemetry. In: Proceedings of the 14th NASA Formal Methods Symposium (NFM 2022). Lecture Notes in Computer Science (LNCS), vol. 13260. Springer, Cham, Caltech, California, USA (May 2022)
25. Montanari, A., Ratto, E., Corsetti, E., Morzenti, A.: Embedding time granularity in logical specifications of real-time systems. Proceedings. EUROMICRO '91 Workshop on Real-Time Systems pp. 88–97 (1991)
26. Mostafa, M., Bonakdarpour, B.: Decentralized runtime verification of LTL specifications in distributed systems. In: 2015 IEEE International Parallel and Distributed Processing Symposium. pp. 494–503 (2015)
27. Okubo, N.: Using R2U2 in JAXA program. Electronic correspondence (November–December 2020), series of emails and zoom call from JAXA to PI with technical questions about embedding R2U2 into an autonomous satellite mission with a provable memory bound of 200KB
28. 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)
29. Reinbacher, T., Rozier, K.Y., Schumann, J.: Temporal-logic based runtime observer pairs for system health management of real-time systems. In: Proceedings of the 20th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS). Lecture Notes in Computer Science (LNCS), vol. 8413, pp. 357–372. Springer-Verlag (April 2014)
30. Rozier, K.Y., Schumann, J.: R2U2: Tool overview. In: Proceedings of International Workshop on Competitions, Usability, Benchmarks, Evaluation, and Standardisation for Runtime Verification Tools (RV-CUBES). vol. 3, pp. 138–156. Kalpa Publications, Seattle, WA, USA (September 2017)
31. Schumann, J., Moosbrugger, P., Rozier, K.Y.: Runtime Analysis with R2U2: A Tool Exhibition Report. In: Proceedings of the 16th International Conference on Runtime Verification (RV15). Springer-Verlag, Madrid, Spain (September 2016)
32. Wolper, P.: Temporal logic can be more expressive. Information and Control **56**(1), 72–99 (1983)