




Intelligence, Surveillance and Reconnaissance Task Specifications in Temporal Logics

Laura L. Pullum 
AI R&D
The POM Group, LLC
Oak Ridge, USA
laurapullum@gmail.com

Sumit Kumar Jha 
Computer Science Department
Florida International University
Miami, USA
sumit.jha@fiu.edu

Rickard Ewetz 
Electrical and Computer Engineering
University of Florida
Gainesville, USA
rewetz@ufl.edu

Abstract—The DoD desires to incorporate autonomous systems into its capabilities. These systems are notoriously difficult to validate, though they require assurance to be used. One area of interest is Intelligence, Surveillance and Reconnaissance (ISR), whose tasks are typically written in natural language. Temporal logics, e.g., Linear Temporal Logic (LTL) and Signal Temporal Logic (STL), have been used in robotics, cyber-physical systems, and electronics design domains to provide confidence and assurance in the correctness, reliability, and safety of system designs. ISR tasks written as temporal logic specifications enables their properties to be validated. We found that recent natural language to LTL/STL specification translators incorrectly translate natural language ISR descriptions. In this paper, we provide manual translations of ten ISR tasks to temporal logic and validate the specifications. We believe this is the first research to provide a set of ISR tasks specified in temporal logic and checked by a satisfiability solver.

Index Terms—specification, linear temporal logic, signal temporal logic, ISR

I. INTRODUCTION

Intelligence, Surveillance and Reconnaissance (ISR) is important as it provides decision-makers with better situational awareness of conditions, be they on the ground, in the air, at sea, in space, or in cyberspace. ISR is used in a variety of domains, including defense, policing, medical surveillance, and humanitarian missions. ISR actions are based on tasking or goals, and those tasks need to be specified in some way. Describing ISR tasks in temporal logic formalizes the tasks and enables verification of their properties. Temporal logics have been used to verify specifications in a variety of domains, including robotics, booking systems, databases, and cyber-physical systems. In this paper, we describe a range of ISR tasks in natural language, translate the tasks into linear and signal temporal logics, and then evaluate the validity of the specifications. A future goal is to develop an automated natural language to linear temporal logic (LTL) and signal temporal logic (STL) translator for ISR tasks.

The authors acknowledge support from DARPA under agreement number FA8750-23-2-0501. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA or the U.S. Government.

Intelligence, Surveillance and Reconnaissance is the coordinated and integrated acquisition, processing and provision of accurate, relevant, timely information and intelligence to support the decision-making process. There are several forms of ISR – route, area, zone and reconnaissance in force [1]. Route search is an operation focused on obtaining detailed information on a specific route and all adjacent terrain. Area search is a directed effort to obtain detailed information on the terrain or activity within a prescribed area of interest. Zone search is a directed effort to obtain detailed information concerning all entities, routes, obstacles, and terrain within a zone defined by boundaries. We do not cover reconnaissance in force.

Temporal logics are used to formalize task descriptions and enable verification of their properties. There are numerous types of temporal logic. We use LTL and STL given their ability to specify ISR tasks and the availability of tools to determine the validity of the logics.

LTL, introduced by Pnueli [2], forms the basis of many practical specification languages, including STL. LTL is used to reason about the behavior of systems that evolve over discrete time, e.g., software and hardware. Its formulas describe properties of paths in a state transition diagram of a system. An LTL formula is defined recursively according to the following syntax:

$$\phi ::= \pi \mid \neg\phi \mid \phi \wedge \psi \mid \mathbf{F}\phi \mid \mathbf{G}\phi \mid \phi\mathbf{U}\psi \mid \mathbf{X}\phi$$

where ϕ and ψ are LTL formulas, and π is an atomic predicate. \neg (negation), \wedge (and), \vee (or), \rightarrow (imply), and \leftrightarrow (equal) are logical operators. \mathbf{F} (eventually/finally), \mathbf{G} (always/globally), \mathbf{U} (until) and \mathbf{X} (next, i.e., the condition must hold in the next state) are temporal operators. LTL is contained by STL when the time is discrete (v. continuous).

STL, introduced by Maler and Ničković [3], extends LTL to continuous signals (or functions of time) and has become increasingly popular as a specification formalism for requirements of cyber-physical systems (CPS), robotics and control theory. It is commonly used to verify safety and performance properties because it allows specifications such as “signal should always remain within a certain range” and “signal should eventually reach a certain threshold” and others. An

STL formula is defined recursively according to the following syntax:

$$\phi ::= \pi \mid \neg\phi \mid \phi \wedge \psi \mid \mathbf{F}_{[a,b]}\phi \mid \mathbf{G}_{[a,b]}\phi \mid \phi \mathbf{U}_{[a,b]}\psi \mid \mathbf{X}_{[a,b]}\phi$$

with the same logical operators as in LTL. $\mathbf{F}_{[a,b]}$ (eventually/finally), $\mathbf{G}_{[a,b]}$ (always/globally), $\mathbf{U}_{[a,b]}$ (until) and $\mathbf{X}_{[a,b]}$ (next) are temporal operators with real-time constraints $t \in [a, b]$.

Though LTL and STL are similar, LTL is focused on discrete-time systems, whereas STL extends LTL to continuous time signals. Where LTL operates over a sequence of states, STL operates over continuous time. LTL is suitable for specifying systems with discrete event types of behavior, e.g., computer programs or digital circuits. STL is suitable for specifying systems with continuous or analog behaviors, e.g., control systems or cyber-physical systems. Verifying LTL can be computationally expensive, however, reasoning over continuous signals leads to computationally more demanding STL verification. Similarly, while there are mature tools for verifying LTL, research continues in finding efficient algorithms for STL model checking.

There has been a recent surge in research in natural language to temporal logic translators and associated tools [4]–[7]. Despite the recent availability of these tools, we found the resulting specifications are not correct translations of the ISR natural language specifications. The reasons for this are that a) the tools are meant to be domain-agnostic and b) there are few examples of ISR task specifications available in LTL or STL. This and the DOD’s desire to incorporate autonomous systems into its capabilities motivate the research described in this paper.

This paper is organized as follows. This Section (1) provides the introduction. Section 2 presents related work in ISR-relevant specification patterns, real-time system specification languages and structured English grammars. Section 3 presents descriptions of ISR tasks in natural language (English) and their translations to LTL and STL. Section 4 describes specification evaluation results, with Section 5 concluding the paper and noting upcoming research.

II. RELATED WORK

A. ISR-Relevant Specification Patterns

Menghi and colleagues [8] provide mission specification patterns for mobile robots, together with tooling (PSALM) for instantiating, composing, and compiling the patterns to create mission specifications. The mission specifications are written in LTL (Linear Temporal Logic) and CTL (Computation Tree Logic). The PSALM tool is available at [9]. The PSALM tool converts a pattern to LTL or CTL.

Quantitative patterns are built on top of the twenty-two qualitative patterns and are available [10], [11]. Elementary and Composite quantitative patterns are provided. The underlying DSL (Domain-Specific Language) is QUARTET, whose syntax is available [12]. The QUARTET tool is a multi-robot mission specification tool that also translates the specifications into PRISM.

Garcia et al [13], [14] and Dragule et al [15] present a domain specification language (DSL) for robot mission specification and describe PROMISE, a framework for specifying high-level missions for multiple robots. Gil et al [16] continue the multirobot specification work by adding specification decomposition and implementing the work in a tool MutRoSe [17].

Humphrey et al [18], [19] provide some ISR-relevant mission specification patterns in LTL, however, they do not start with a higher-level natural language. The patterns provided are for the following properties: Safety, Reachability, Coverage, Recurrent Coverage, Sequencing, Avoidance, Avoidance with Reachability, Sequencing with Avoidance, Previously, and Never After. Similarly, Salamah et al [20] defined general LTL formulas that can be used to generate LTL specifications of properties defined by patterns, scopes, and composite propositions. They do not provide natural language descriptions of the patterns other than a descriptive pattern name. Though the patterns are general, they can be used for UAV-relevant mission specification. Salameh et al [21] extend this work to provide validated templates for specifying complex LTL patterns. He et al [7] identified common STL templates, namely invariance/reachability, immediate response, temporal response, and stabilization/recurrence.

B. Real-Time System Specification Languages and Structured English Grammars

Early work on real-time system specification languages and patterns was conducted by Dwyer et al [22], Konrad and Cheng [23], [24], and Bellini, Nesi and Rogai [25]. Dwyer et al [22] provided a set of qualitative specification patterns using LTL and CTL (Computational Tree Logic) focusing on properties like occurrence and event ordering. Konrad and Cheng extended Dwyer’s patterns by adding concepts of duration, periodic and real-time ordering. They specified the patterns in MTL (Metric Temporal Logic) among others, but not in LTL or STL. Bellini et al add or renovate two patterns: Time-Constrained Precedence and Time-Constrained Response.

The PSPF Language is based on temporal logic and is implemented by the PSPWizard tool. PSPWizard makes use of a catalog of specification patterns described in Autili et al [26]. The Property Specification Patterns Website [27] provides patterns and a structured English attribute grammar including terms Recurrence, Response, Absence and Probability, for example. A mapping of part of the grammar to MTL and a mapping from the patterns to LTL and CTL are provided.

Kapinski et al [28], present ST-LIB, a set of STL formulae particularly suited to the automotive control domain. The parametric formulae provided include Ringing, Spike, Overshoot (Undershoot), Settling Time, Rise Time, Timed Relationships, and Steady State Error. Vogel et al [29] provide a specification pattern catalog for the real-time model checking tool UPPAAL. The qualitative and real-time requirements in patterns in the Property Specification Patterns Website (see above) are included and UPPAAL updated with an automated generator

that translates these requirements to observer automata and TCTL formulae. The tool and patterns are provided at [30]. Klikovits [31] catalogs numerous formalisms used in cyber-physical systems development, however, the most relevant and more recent work is described above.

III. NATURAL LANGUAGE ISR TASKS TO LTL AND STL SPECIFICATIONS

This section provides natural language descriptions of ten ISR tasks and their representation as LTL and STL specifications. The LTL and STL are manually derived from the natural language descriptions.

A. Zone Search

The mission class is a Zone Search with the objective of finding an entity. The boundaries of the zone are defined.

Given the propositions P - the entity is in the zone being searched, and Q - the UAV is in the search zone, the LTL specification for Zone Search is

$$\phi = \mathbf{F}(P \wedge Q) \quad (1)$$

where \mathbf{F} means eventually and $P \wedge Q$ means that eventually the UAV and the entity will be in the search zone together.

For the STL specification of a Zone Search, given the signals $p(t)$ - the entity in the search zone at time t , and $q(t)$ - the UAV in the search zone at time t , then the STL specification for a Zone Search is

$$\phi = \mathbf{G}_{[a,b]}(p(t) \rightarrow \mathbf{F}_{[a,b]}q(t)) \quad (2)$$

where \mathbf{G} means always. This specification states that if, during the real-time constraints $t \in [a, b]$, $p(t)$ holds, then eventually $q(t)$ will hold within the same period.

B. Area Search for Static Entity

The mission class is an Area Search with the objective of finding a static entity within one of the grids in the search area. Given the propositions P_i - the presence of the entity in grid i , where $i = 1, \dots, n$, and Q_i - the search in grid i , the LTL task specification for this case is

$$\phi = \mathbf{F}_{[0,T]} \left(\bigvee_{i=1}^n (P_i \wedge Q_i) \right) \quad (3)$$

Suppose a keep-out zone is added, in addition to prior belief about the static entity's location within grids in the search area. Given the propositions P_i - the presence of the entity in grid i , where $i = 1, \dots, n$, K - the presence of the UAV in the keep-out zone, and B_i - prior belief probability that the entity is in grid i , $B_i \in [0.0, 1.0]$, then the LTL task specification is

$$\phi = \mathbf{F}_{[0,T]} \left(\bigvee_{i=1}^n P_i \right) \wedge \neg \mathbf{F}_{[0,T]} K \wedge \mathbf{F}_{[0,T]} \left(\bigvee_{i=1}^n (P_i \wedge B_i) \right) \quad (4)$$

Next, we define the STL specification of an Area Search with the objective of finding a static entity within one of the grids in the search area. Given the signals $p_i(t)$ - the presence

of the entity in grid i at time t , $t \in [0, T]$, and $q(t)$ - the search action at time t , $t \in [0, T]$, the STL specification is

$$\phi = \mathbf{F}_{[0,T]} \left(\bigvee_{i=1}^n p_i(t) \wedge q(t) \right) \quad (5)$$

To add a keep-out zone and prior belief about the static entity's location within grids in the search area, we add the signals $k(t)$ - the presence of the UAV in the keep-out zone at time t , and $b_i(t)$ prior belief probability that the entity is in grid i at time t , $b_i(t) \in [0.0, 1.0]$, and change the STL specification to

$$\phi = \mathbf{F}_{[0,T]} \left(\bigvee_{i=1}^n p_i(t) \right) \wedge \mathbf{G}_{[0,T]} \neg k(t) \wedge \mathbf{F}_{[0,T]} \left(\bigvee_{i=1}^n (p_i(t) \wedge b_i(t)) \right) \quad (6)$$

C. Route Search

The mission class is a Route Search with the objective of finding a moving entity on a route defined by waypoints. The waypoints are to be visited in order. Following the route is of primary importance, so the UAV must visit the waypoints in order and stay within a band of defined width on either side of the route. Given the propositions P_i - the presence of the UAV at waypoint i , where waypoint number, $i = 1, \dots, n$, and Q_i - the UAV stays within a band around the route defined by the waypoints, the LTL task specification is

$$\phi = \mathbf{G}_{[0,T]} \left(\left(\bigvee_{i=1}^{n-1} ((P_i \wedge Q_i) \rightarrow \mathbf{X}P_{i+1}) \right) \wedge (P_n \wedge Q_n) \right) \quad (7)$$

This equation introduces the LTL unary operator \mathbf{X} or *next*, which asserts the truth of the subformula that follows it for the next system state reached. So $\mathbf{X}P_{i+1}$ asserts that the next waypoint is visited, allowing visitation of the waypoints in order.

Suppose once the entity is found, a report is triggered. Then the LTL task specification is

$$\phi = \mathbf{G} \left(\bigvee_{i=1}^{n-1} (Q_i \rightarrow (P_i \wedge \mathbf{X}Q_{i+1})) \wedge (Q_n \rightarrow P_n) \right) \wedge \mathbf{F}(L) \quad (8)$$

where i is the waypoint number, $i = 1, \dots, n$, and L indicates the entity's location.

To specify the STL specification for a route search with given waypoints, we first define the signals $p_i(t)$ - the presence of the UAV at waypoint i at time t , $i = 1, \dots, n$ and $t \in [a, b]$, and $q_i(t)$ - the distance between the UAV and the waypoint i at time t , and the constraint $q_i(t) \leq \text{bandwidth}$ - the UAV's distance from the defined route, on either side, at time t , must be within the *bandwidth*.

$$\phi = \mathbf{G}_{[a,b]} \left(\left(\bigwedge_{i=1}^{n-1} (p_i(t) \wedge q_i(t)) \rightarrow \mathbf{F}_{[t,t+\epsilon]} p_{i+1}(t) \right) \wedge (p_n(t) \wedge q_n(t)) \right) \quad (9)$$

To include a report being triggered once the entity is found, we add the symbol $r(t)$ indicating the presence of a report on the entity at time t and the STL formula changes to the following:

$$\phi = \mathbf{G}_{[a,b]} \left(\bigvee_{i=1}^n (p_i(t) \rightarrow \mathbf{F}_{[a,b]} q(t)) \wedge \mathbf{F}_{[a,b]} r(t) \right) \quad (10)$$

stating that sometime after the entity is found a report is triggered.

D. Area Search and Pursue On-Road

The mission class is an Area Search with the objective of finding a moving entity on a road. Once the entity is detected, the objective is to pursue the found entity and periodically report perceptions. Given the propositions P - the presence of the moving entity, Q - the pursuit of the entity, and R - the reporting of perceptions, the LTL task specification is

$$\phi = \mathbf{F}(P \wedge (\mathbf{F}Q \wedge \mathbf{F}_{[a,b]}R)) \quad (11)$$

where $\mathbf{F}_{[a,b]}$ means eventually during the time period $[a, b]$, where a is after the target is found.

To specify this task in STL, we define the signals $p(t)$ - presence of the entity at time t , $i = 1, \dots, n$ and $t \in [a, b]$, $q(t)$ - pursuit of the entity at time t , and $r(t)$ - periodic reporting at time t .

$$\phi = \mathbf{G}_{[a,b]} \left(p(t) \rightarrow (q(t) \wedge (\mathbf{F}_{[a,b]} r(t))) \right) \quad (12)$$

E. Area Search and Pursue Off-Road

The mission class is an Area Search with the objective of finding a moving entity off-road. With the entity operating off-road, there are more occlusions due to trees and landscape. When the view is blocked due to trees, suppose the agent (e.g., UAV) stops searching and when the view is clear, it resumes the search. As with the on-road pursuit, once the entity is detected, the objective is to pursue the found entity and periodically report perceptions.

To specify this in LTL, we define the following propositions: S - the UAV is searching, B - the UAV's view is blocked by trees, D - the entity is detected, P - the UAV is pursuing the entity, and R - The UAV is reporting perceptions. The LTL task specification for this task is

$$\phi = \mathbf{G} \left((S \rightarrow \mathbf{X}(\neg B)) \wedge (B \rightarrow \mathbf{X}S) \wedge \mathbf{F}D \wedge (\mathbf{G}(D \rightarrow \mathbf{F}P) \wedge \mathbf{G}(P \rightarrow \mathbf{F}(P \wedge \mathbf{X}R))) \right) \quad (13)$$

where the area search is defined by $\mathbf{G}(S \rightarrow \mathbf{X}(\neg B)) \wedge (B \rightarrow \mathbf{X}S)$. Entity detection is defined by $\mathbf{F}D$. And, entity pursuit and reporting is defined by the last part of the equation, i.e., $\mathbf{G}(D \rightarrow \mathbf{F}P) \wedge \mathbf{G}(P \rightarrow \mathbf{F}(P \wedge \mathbf{X}R))$.

To specify this in STL, we define the following predicates (signal functions) given t is time and $t \in [a, b]$: $v(t)$ - UAV's view condition at time t , where $v(t) = 1$ when the view is clear, and $v(t) = 0$ when the view is blocked; $d(t)$ - Detection of entity at time t , where $d(t) = 1$ when the entity is detected

at time t , and $d(t) = 0$ otherwise; $p(t)$ - Pursuit of entity at time t , where $p(t) = 1$ when the UAV is pursuing the entity at time t , and $p(t) = 0$ otherwise; and $r(t)$ - Reporting perceptions of entity at time t , where $r(t) = 1$ when the UAV is reporting at time t , and $r(t) = 0$ otherwise. The STL formula is

$$\begin{aligned} \phi = & \mathbf{G}_{[a,b]} \left(v(t) \rightarrow \mathbf{F}_{[a,b]} (v(t) = 1) \right) \wedge \\ & \mathbf{F}_{[a,b]} (d(t) = 1) \wedge \\ & \mathbf{G}_{[a,b]} \left(d(t) = 1 \rightarrow \mathbf{F}_{[a,b]} (p(t) = 1) \right) \wedge \\ & \mathbf{G}_{[a,b]} \left(p(t) = 1 \rightarrow \mathbf{F}_{[a,b]} \left(p(t) = 1 \wedge \right. \right. \\ & \left. \left. \mathbf{F}_{[a,b]} (r(t) = 1) \right) \right) \end{aligned} \quad (14)$$

where the area search is defined by $\mathbf{G}_{[a,b]} (v(t) \rightarrow \mathbf{F}_{[a,b]} (v(t) = 1))$. Entity detection is defined by $\mathbf{F}_{[a,b]} (d(t) = 1)$. And entity pursuit and reporting is defined by the last part of the equation.

F. Area Search and Pursue Evasive

The mission class is an Area Search with the objective of finding a moving entity within a defined search area. Once found, the UAV pursues the found entity and periodically reports perceptions about the entity. The entity performs evasive maneuvers to add occlusions to the UAV's field of view (FOV). When the FOV is blocked due to the evasive maneuvers, the UAV continues searching in a region with a defined radius centered on where the entity left the FOV. When the entity re-enters the FOV, the pursuit and reporting continue. The entity is within the search area during the search period.

To specify this in LTL, we define the following propositions: E - the entity is present, P - the UAV is pursuing the entity, R - the UAV is reporting perceptions, O - the UAV's view is occluded, and S - the UAV continues searching in a region with a defined radius. The LTL specification for this task is

$$\phi = \mathbf{G} \left((E \rightarrow (P \wedge \mathbf{F}R)) \wedge (\mathbf{G}(O \rightarrow \mathbf{F}S) \wedge \mathbf{G}(S \rightarrow \mathbf{F}O)) \right) \quad (15)$$

To specify this in STL, we define the following predicates (signal functions), given t is time and $t \in [0, T]$: $e(t)$ - presence of the entity at time t , $p(t)$ - pursuit of the entity at time t , $r(t)$ - reporting on the entity at time t , $o(t)$ - UAV's FOV occluded at time t , and $s(t)$ - continuation of search at time t in a region with a defined radius. The STL specification for this task is

$$\begin{aligned} \phi = & \mathbf{G}_{[0,T]} \left(\left(e(t) \rightarrow (p(t) \wedge \mathbf{F}_{[0,T]} r(t)) \right) \wedge \right. \\ & \left. (\mathbf{G}(o(t) \rightarrow \mathbf{F}_{[0,T]} s(t)) \wedge \mathbf{G}(s(t) \rightarrow \mathbf{F}_{[0,T]} o(t))) \right) \end{aligned} \quad (16)$$

G. Overwatch and Route Search

In this task, there are boundaries that entities may try to cross. The goal is to keep them outside the boundaries. There are routes the entities can use to reach the boundaries, so this is an overwatch of routes to search for moving entities. When an entity is located, the UAV is to pursue it and report perceptions.

To specify this in LTL, we define the following propositions: E - the entity is present, B - the presence of entities attempting to cross the boundaries, and P - the UAV is pursuing the entity and reporting on the entity. The LTL specification for this task is

$$\phi = \mathbf{G}\left((E \rightarrow (FP)) \wedge (\mathbf{G}(B \rightarrow FP))\right) \quad (17)$$

To specify this in STL, we define the following predicates (signal functions) given t is time and $t \in [0, T]$: $e(t)$ - presence of entities at time t , $b(t)$ - presence of entities trying to cross a boundary at time t , $p(t)$ - pursuit of the entities at time t , and $r(t)$ - reporting on entities at time t . The STL specification for this task is

$$\phi = \mathbf{G}_{[0,T]}\left(\left(e(t) \rightarrow (p(t) \wedge \mathbf{F}_{[0,T]}r(t))\right) \wedge \mathbf{G}_{[0,T]}(b(t) \rightarrow \mathbf{F}_{[0,T]}p(t))\right) \quad (18)$$

H. Evade and Reappear

The mission class is an area search for a moving entity. The entity attempts to evade perception by going outside the search area bounds, then reappears somewhere within the search area. Once the entity is located, the UAV should pursue it and report perceptions.

To specify this in LTL, we define the following propositions: E - the moving entity is present in the search area bounds, P - the UAV is pursuing the entity and reporting on the entity, and R - the UAV is reporting perceptions. The LTL specification for this task is

$$\phi = \mathbf{G}(E \rightarrow (P \wedge FR)) \quad (19)$$

To specify this in STL, we define the following signals: $e(t)$ - presence of the moving entity at time t , $p(t)$ - pursuit of the entity at time t , and $r(t)$ - reporting on the entity at time t . The STL specification for this task is

$$\phi = \mathbf{G}_{[0,T]}\left(e(t) \rightarrow (p(t) \wedge \mathbf{F}_{[0,T]}r(t))\right) \quad (20)$$

IV. RESULTS

A. Satisfiability of the LTL Specifications

To evaluate the LTL specification satisfiability, we used the tool BLACK, a recently developed software tool for satisfiability checking of LTL formulas. BLACK uses an incremental SAT encoding of the one-pass tree-shaped tableau for LTL developed by Reynolds [32], which guarantees completeness due to its pruning rule. Given an LTL formula ϕ , the tool encodes in a SAT formula the tableau for ϕ up to depth k , for increasing values of k , until an accepted branch is found or a witness of unsatisfiability is detected. Experimental evaluations performed against other state-of-the-art tools

show BLACK's competitive performance and low memory consumption [33].

BLACK is an easy-to-use tool for checking the satisfiability of LTL formulas. For example, for the Zone Search LTL formula, $\phi = \mathbf{G}(P \rightarrow \mathbf{F}Q)$, translate the formula into the BLACK grammar, that is, $\mathbf{G}(p \rightarrow \mathbf{F}q)$. Then save the BLACK grammar formula in a file, `ZoneSearch.pltl` and run the command `black solve ZoneSearch.pltl`. BLACK returns either SAT or UNSAT, in this case it returns SAT, the LTL specification is satisfiable.

Let's look at a more complex formula, that for Area Search and Pursue Off-Road (see equation (13)). Translating this into the BLACK grammar yields $\mathbf{G}((s \rightarrow \mathbf{X}(!b)) \ \&\& \ (b \rightarrow \mathbf{X}s)) \ \&\& \ \mathbf{F}d \ \&\& \ (\mathbf{G}(d \rightarrow \mathbf{F}p)) \ \&\& \ \mathbf{G}(p \rightarrow \mathbf{F}(p \ \&\& \ \mathbf{X}r)))$. After saving the formula in `AreaSearchPursueOff.pltl`, run the command `black solve AreaSearchPursueOff.pltl`. BLACK returns SAT, the LTL specification is satisfiable.

For formulas that use the OR operator over an interval ($i = 1, \dots, n$) we chose to evaluate the case where $n=4$, as in four quadrants in a grid. This is the case for the following specifications: Area Search for a Static Entity, Route Search, and Route Search with Report. For example, the associated formula in BLACK grammar for an Area Search for a Static Entity with $n=4$ is $\mathbf{F}((p1 \ \&\& \ q1) \ || \ (p2 \ \&\& \ q2) \ || \ (p3 \ \&\& \ q3) \ || \ (p4 \ \&\& \ q4))$. Running the BLACK tool on this formula yields SAT, the LTL specification is satisfiable. Each of the LTL specifications were found to be satisfiable using BLACK.

B. Satisfiability of the STL Specifications

Typically, satisfiability checkers for STL specifications require a model of the system in addition to the specification. Without such a model, a trace of system behavior is used. To evaluate the STL specification satisfiability, we generated a trace of the natural language specification. Using that trace and the STL specification, we used STLrom [34], [35] to evaluate the specifications.

STLrom enables online monitoring of the satisfaction of a formula. In fact, the tool allows robust online monitoring of partial traces, i.e., traces for which there might not be enough data to decide the Boolean satisfaction. The tool incorporates an efficient algorithm to compute a robust satisfaction value, that is, a function mapping a property ϕ and a trace $x(t)$ to a real number.

The robust interval semantics implemented in STLrom map a trace and an STL property to an interval (l, u) , interpreted as follows. For any suffix $u(t)$, l is the greatest lower bound on the quantitative semantics of the trace, and u is the corresponding lowest upper bound. As stated by the tool developers, there is a natural correspondence between the interval semantics and three-valued semantics:

- the truth value of ϕ is false according to the weak view *iff* (if and only if) u is negative, and true otherwise;
- the truth value is true according to the strong view *iff* l is positive, and false otherwise; and

- a neutral semantics, e.g., based on some predictor, can be defined when $l < 0 < u$, i.e., when there exist both suffixes that can violate or satisfy ϕ .

To use STL_{Rob}, we need a trace and the specification. We developed traces for each natural language description of a specification and implemented each specification in a program using STL_{Rob}'s STL_{Driver} and its functionality. The result is [0.5, 0.5, 0.5], that is, [*robustness estimate*, *robustness lower bound*, *robustness upper bound*]. If the specification can be fully evaluated by the data, then all three values are equal to the actual robust satisfaction of the formula. We evaluated all STL specifications and found that three were in error (not satisfiable). We repaired and re-evaluated those formulas, resulting in all STL specifications provided herein found to be robustly satisfied.

The bottom line is that for each LTL and STL specification presented in this paper, the satisfiability was shown to be true.

V. CONCLUSIONS

We have provided natural language descriptions along with LTL and STL specifications for ten ISR tasks. Each specification was evaluated and shown to be satisfiable. Future research includes automating the development of ISR specifications from their natural language descriptions.

ACKNOWLEDGMENT

L.L.P. thanks Alexandre Donzé for his assistance using the STL_{Rob} tool.

REFERENCES

- [1] "Offense and defense," Department of the Army, Washington, DC, Tech. Rep. ADP 3-90, 1987.
- [2] A. Pnueli, "The temporal logic of programs," in *Symposium on Foundations of Computer Science*, Providence, USA, 1977, pp. 46–57.
- [3] O. Maler and D. Ničković, "Monitoring temporal properties of continuous signals," in *FORMATS/TRTFT*, Y. Lakhnech and S. Yovine, Eds. Heidelberg, Germany: Springer, 2004, pp. 152–166.
- [4] Y. Chen, R. Gandhi, Y. Zhang, and C. Fan, "NI2tl: Transforming natural languages to temporal logics using large language models," in *Conference on Empirical Methods in Natural Language Processing*, Singapore, Dec. 2023, pp. 15 880–15 903.
- [5] M. Cosler, C. Hahn, D. Mendoza, F. Schmitt, and C. Trippel, "nl2spec: Interactively translating unstructured natural language to temporal logics with large language models," in *Computer Aided Verification*, Paris, France, Jul. 2023, pp. 383–396.
- [6] F. Fuggitti and T. Chakraborti, "NI2tl – a python package for converting natural language (nl) instructions to linear temporal logic (ltl) formulas," in *AAAI Conference on Artificial Intelligence AAAI-23*, Washington, DC, Feb. 2023, pp. 16 428–16 430.
- [7] J. He, E. Bartocci, D. Ničković, H. Isakovic, and R. Grosu, "Deepstl - from english requirements to signal temporal logic," in *International Conference on Software Engineering*, Pittsburgh, USA, May 2022, pp. 610–622.
- [8] C. Menghi, C. Tsigkanos, P. Pelliccione, C. Ghezzi, and T. Berger, "Specification patterns for robotic missions," *IEEE Trans. Software Eng.*, vol. 47, no. 10, pp. 2208–2224, 2021. [Online]. Available: <https://doi.org/10.1109/TSE.2019.2945329>
- [9] "Specification patterns for robotic missions," <http://www.roboticpatterns.com/>, 2023, accessed 2023-11-06.
- [10] C. Menghi, C. Tsigkanos, M. Askarpour, P. Pelliccione, G. Vázquez, R. Calinescu, and S. García, "Mission specification patterns for mobile robots: providing support for quantitative properties," *IEEE Transactions on Software Engineering*, vol. 49, no. 4, pp. 2741–2760, Apr. 2023.
- [11] "Quantitative specification patterns for robotic missions," <http://www.roboticpatterns.com/quantitative>, 2023, accessed 2023-11-06.
- [12] "Quantitative specification patterns for robotic missions – quartet dsl: syntax," <https://dsg.tuwien.ac.at/staff/ctsiganos/patterns/quantitative/quartet/>, 2023, accessed 2023-11-06.
- [13] S. García, P. Pelliccione, C. Menghi, T. Berger, and T. Bures, "Promise: High-level mission specification for multiple robots," in *International Conference on Software Engineering*, Seoul, South Korea, Jul. 2020, pp. 5–8.
- [14] "Promise implementation," https://github.com/SergioGarG/PROMISE_implementation/tree/dev, 2023, accessed 2023-11-08.
- [15] S. Dragule, S. G. Gonzalo, T. Berger, , and P. Pelliccione, "Languages for specifying missions of robotic applications," in *Software Engineering for Robotics*, A. Cavalcanti *et al.*, Eds. Cham: Springer, 2020, pp. 377–411.
- [16] E. B. Gil, G. N. Rodrigues, P. Pelliccione, and R. Calinescu, "Mission specification and decomposition for multi-robot systems," *Robotics and Autonomous Systems*, vol. 163, no. C, p. 104386, May 2023.
- [17] "Mutrose-artifacts," <https://github.com/lesunb/MutRoSe-Artifacts>, 2023, accessed 2023-11-06.
- [18] L. R. Humphrey, E. M. Wolff, and U. Topcu, "Formal specification and synthesis of mission plans for unmanned aerial vehicles," in *Formal Verification and Modeling in Human-Machine Systems Workshop of the AIAA Spring Symposium*, Palo Alto, USA, Mar. 2014.
- [19] B. U. Kim and L. R. Humphrey, "Satisfiability checking of ltl specifications for verifiable uav mission planning," in *AIAA SciTech Forum*, National Harbor, USA, Jan. 2014.
- [20] S. Salamah, A. Q. Gates, V. Kreinovich, and S. Roach, "Automatic generation of complex ltl specifications through patterns and composite propositions," University of Texas at El Paso, El Paso, TX, Tech. Rep., 2007.
- [21] S. Salamah, A. Q. Gates, and V. Kreinovich, "Validated templates for specification of complex ltl formulas," *Journal of Systems I & Software*, vol. 85, no. 8, pp. 1915–1929, Aug. 2012.
- [22] M. B. D. G. S. Avrunin and J. C. Corbett, "Patterns in property specifications for finite-state verification," in *IEEE International Conference on Software Engineering*, Los Angeles, USA, May 1999, pp. 411–420.
- [23] S. Konrad and B. H. C. Cheng, "Real-time specification patterns," in *IEEE International Conference on Software Engineering*, St. Louis, USA, May 2005, pp. 372–381.
- [24] —, "Facilitating the construction of specification pattern-based properties," in *IEEE International Conference on Requirements Engineering*, La Sorbonne, France, Aug. 2005, pp. 329–338.
- [25] P. Bellini, P. Nesi, and D. Rogai, "Expressing and organizing real-time specification patterns via temporal logics," *Journal of Systems I & Software*, vol. 82, no. 2, pp. 183–196, Feb. 2009.
- [26] M. Autili, L. Grunske, M. Lumpe, P. Pelliccione, and A. Tang, "Aligning qualitative, real-time, and probabilistic property specification patterns using a structured english grammar," *IEEE Transactions on Software Engineering*, vol. 41, no. 7, p. 620–638, Jul. 2015.
- [27] "Property specification patterns," <http://ps-patterns.wikidot.com>, 2023, accessed 2023-11-01.
- [28] J. Kapinski *et al.*, "St-lib: A library for specifying and classifying model behaviors," SAE, Tech. Rep., 2016.
- [29] T. Vogel, M. Carwehl, G. N. Rodrigues, and L. Grunske, "A property specification pattern catalog for real-time system verification with uppaal," *Information and Software Technology*, vol. 154, p. 107100, Nov. 2022.
- [30] "Specification pattern catalogue for uppaal," <https://github.com/hub-se/PSP-UPPAAL/wiki>, 2021, accessed 2024-02-12.
- [31] S. Klikovits *et al.*, "State-of-the art on current formalisms used in cyber-physical systems development," COST European Cooperation in Science and Technology, Tech. Rep. hal-03168832, 2019.
- [32] M. Reynolds, "A new rule for ltl tableaux," in *7th International Symposium on Games, Automata, Logics and Formal Verification*, Catania, Italy, Sep. 2016, p. 287–301.
- [33] L. Geatti, N. Gigante, and A. Montanari, "A sat-based encoding of the one-pass and tree-shaped tableau system for ltl," in *28th International Conference on Automated Reasoning with Analytic Tableaux and Related Methods*, London, England, Sep. 2019, p. 3–20.
- [34] J. V. Deshmukh, A. Donzé, S. Ghosh, X. Jin, G. Juniwal, , and S. A. Seshia, "Robust online monitoring of signal temporal logic," *Form Methods Syst Des*, vol. 51, p. 5–30, 2017.
- [35] (2024) Github decyphir / stlrom. [Online]. Available: <https://github.com/decyphir/STLrom>