

# Specification Patterns for Robotic Missions

Claudio Menghi<sup>1</sup>, Christos Tsigkanos<sup>2</sup>, Patrizio Pelliccione<sup>3,4</sup>, Carlo Ghezzi<sup>5</sup>, Thorsten Berger<sup>4</sup>

<sup>1</sup> University of Luxembourg, Luxembourg

<sup>2</sup> Technische Universität Wien, Austria

<sup>3</sup> University of LAquila, Italy.

<sup>4</sup> Chalmers University of Technology — University of Gothenburg, Sweden

<sup>5</sup> Politecnico di Milano, Italy

**Abstract**—Mobile and general-purpose robots increasingly support our everyday life, requiring dependable robotics control software. Creating such software mainly amounts to implementing their complex behaviors known as missions. Recognizing the need, a large number of domain-specific specification languages has been proposed. These, in addition to traditional logical languages, allow the use of formally specified missions for synthesis, verification, simulation, or guiding the implementation. For instance, the logical language LTL is commonly used by experts to specify missions, as an input for planners, which synthesize the behavior a robot should have. Unfortunately, domain-specific languages are usually tied to specific robot models, while logical languages such as LTL are difficult to use by non-experts.

We present a catalog of 22 mission specification patterns for mobile robots, together with tooling for instantiating, composing, and compiling the patterns to create mission specifications. The patterns provide solutions for recurrent specification problems, each of which detailing the usage intent, known uses, relationships to other patterns, and—most importantly—a template mission specification in temporal logic. Our tooling produces specifications expressed in the LTL and CTL temporal logics to be used by planners, simulators, or model checkers. The patterns originate from 245 realistic textual mission requirements extracted from the robotics literature, and they are evaluated upon a total of 441 real-world mission requirements and 1251 mission specifications. Five of these reflect scenarios we defined with two well-known industrial partners developing human-size robots. We validated our patterns’ correctness with simulators and two real robots.

## I. INTRODUCTION

Mobile robots are increasingly used in complex environments aiming at autonomously realizing missions [1]. The rapid pace of development in robotics hardware and technology demands software that can sustain this growth [2]–[5]. Increasingly, robots will be used for accomplishing tasks of everyday life by end-users with no expertise and knowledge in computer science, robotics, mathematics or logics. Providing techniques that support robotic software development is a major software-engineering challenge [2], [6]–[10].

The mission describes the high-level tasks the robotic software must accomplish [11]. Among the different ways of describing missions that were proposed in the literature [12]–[18], in this work, we consider declarative specifications [19]. These describe the final outcome the software should achieve—rather than describing how to achieve it—and are prominently used in the robotics domain [10], [17], [20]–[28]. Precisely specifying missions and transforming them into a form useful for automatic processing are among the main challenges in

engineering robotics software [9], [29]–[32]. On the one hand, missions should be defined with a notation that is high-level and user-friendly [11], [33], [34]. On the other hand, to enable automatic processing, the notation should be unambiguous and provide a formal and precise description of what robots should do in terms of movements and actions [35]–[37].

Typically, when engineering robotics software, the missions are first expressed using natural-language requirements. These are then specified using domain-specific languages, many of which have been proposed over the last decades [12]–[14], [38]. These languages are often integrated with development environments that are used to generate code that can be executed within simulators or real robots [15]–[17]. However, these languages are typically bound to specific types of robots and support a limited number and type of missions. Other works, especially coming from the robotics domain, advocate to formally specify missions in temporal logics [20], [26], [28], [39]. Unfortunately, defining temporal logic formulae is complicated. As such, the definition of mission specifications is laborious and error-prone, as widely recognized in the software-engineering and robotics communities (e.g., [9], [40]–[42]).

Conceptually, defining a robotic mission entails two problems. First, ambiguities in mission requirements that prevent precise and unambiguous specifications must be resolved [38], [42], [43]. Consider the very simple mission requirement “the robot shall visit the kitchen and the office.” This can be interpreted as “visit the kitchen” and also that at some point the robot should “visit the office” or visit “the kitchen and the office in order.” This highlights the ambiguity in natural language requirements formulation, and common mistakes may be introduced when diverse interpretations are given [37], [44]–[46]. Second, creating specifications that correctly capture requirements is hard and error prone [9], [40]–[42]. Assume that the correct intended behavior requires that “the kitchen and the office are visited in order,” which is a common mission specification problem [47], [48]. When transforming this requirement into a precise mission specification, an expert might come up with the following formula in temporal logic:

$$\phi_1 = \mathcal{F}((r \text{ in } l_1) \wedge \mathcal{F}(r \text{ in } l_2)),$$

where  $r \text{ in } l_1$  and  $r \text{ in } l_2$  signify that robot  $r$  is in the kitchen and office, respectively, and  $\mathcal{F}$  denotes *finally*. Now, recall that the actual requirement is that the robot reaches the kitchen *before* the office. Unfortunately, the logical formula still admits

that the robot reaches the office before entering the kitchen, which may be an unintended behavior. Mitigating this problem requires defining additional behavioral constraints. A correct formula, among others, is the following:

$$\phi_2 = \phi_1 \wedge ((\neg(r \text{ in } l_2)) \mathcal{U} (r \text{ in } l_1)),$$

where  $\mathcal{U}$  stands for *until*. The additional constraint requires the office to not be visited before the kitchen, recalling a specification pattern for temporal logics known as the *absence pattern* [49]. Rather than conceiving such specifications recurrently in an ad hoc way with the risk of introducing mistakes, engineers could re-use validated solutions to existing mission requirements.

Specification patterns are a popular solution to the specification problem. While precise behavioral specifications in logical languages enable reasoning about behavioral properties [50], [51], specification is hard and error prone [52], [53]. The problem is exacerbated, since practitioners are often unfamiliar with the intricate syntax and semantics of logical languages [49]. For instance, Dwyer et al. [49] introduced patterns for safety properties, later extended by Grunske [54] and Konrad et al. [55] to address real-time and probabilistic quality properties. Autili et al. [56] consolidated and organized these patterns into a comprehensive catalog. Bianculli et al. [57] applied specification patterns to the domain of Web services. All these patterns provide template solutions that can be used to specify the respective properties. However, none of these pattern catalogs focuses on the robotic software domain to solve the mission specification problem.

We propose a pattern catalog and supporting tooling that facilitates engineering missions for mobile robots, which implements the original, high level idea that we had recently presented [58]. We focus on robot movement as one of the major aspects considered in the robotics domain [59]–[61], as well as on how robots perform actions as they move within their environment. For each pattern we provide usage intent, known uses, relationships to other patterns, and—most importantly—a template mission specification in temporal logic. The latter relies on LTL and CTL as the most widely used formal specification languages in robotics [10], [17], [20]–[28]. The catalog has been produced by analyzing 245 natural-language mission requirements systematically retrieved from the robotics literature. From these requirements we identified recurrent mission specification problems and conceived solutions were organized as patterns in a catalog. Our patterns provide a formally defined vocabulary that supports robotics developers in defining mission requirements. Relying on the usage of the pattern catalog as a common vocabulary allows mitigating ambiguous natural language formulations [41]. Our patterns also provide validated mission specifications for recurrent mission requirements, facilitating the creation of correct mission specifications [27].

We implemented the tool PsAIM (Pattern bAsed Mission specifier) [62] to further support developers in rigorous mission design. PsAIM allows (i) specifying a mission requirement through a structured English grammar, which uses patterns

as basic building blocks and operators that allow composing these patterns into complex missions, and (ii) automatically generating specifications from mission requirements. PsAIM is robot-agnostic and integrated with: Spectra [63] (a robot development environment), a planner [26], NuSMV [64] (a model checker), and Simbad [65] (a simulator for education and research). The pattern catalog and the PsAIM tool are available in an online appendix [66].

We evaluated the benefits obtained by the usage of our pattern support in rigorous and systematic mission design. We collected 441 mission requirements in natural language: 436 obtained from robotic development environments used by practitioners (i.e., Spectra [63] and LTLMoP [39], [42]), and five defined in collaboration with two well-known robotics companies developing commercial, human-size service robots (BOSCH and PAL Robotics). We show that most of the mission requirements were ambiguous, expressible using the proposed patterns, and that the usage of the patterns reduces ambiguities. We then evaluated the coverage of mission specifications. We collected 1229 LTL and 22 CTL mission specifications, from robotic development environments used by practitioners (i.e., Spectra [63] and LTLMoP [39], [42]) and research publications (i.e., [67]) and show that almost all the specifications can be obtained using the proposed patterns (1154 over 1251). We also generated the specifications for the five mission requirements defined in collaboration with the two robotic companies and fed them into an existing planner. The produced plans were correctly executed by real robots, showing the benefits of the pattern support in real scenarios.

To ensure the correctness of the proposed patterns we manually inspected their template mission specifications. We additionally tested patterns correctness on a set of 12 randomly generated models representing buildings where the robot is deployed. We considered ten mission requirements (each obtained by combining three patterns), converted the mission requirements into LTL mission specifications and used those to generate robots’ plans. We used the Simbad [65] simulator, to verify that the plans satisfied the intended mission requirement. We subsequently generated both LTL and CTL specifications from the considered mission requirements. We verified that the same results are obtained when they are checked on the randomly generated models, confirming the correspondence among the CTL and LTL specifications.

## II. BACKGROUND

In this section, we present the terminology used in the remainder and introduce the temporal logics LTL and CTL we used for defining the patterns’ template solutions.

Recall that for communication and further refinement, the requirements of a software system are typically expressed in natural language or informal models. Refining these requirements into more formal representations avoids ambiguity, allowing automated processing and analysis. Such practices also emerged in the robotics engineering domain.

- *Mission Requirement*: a description in a natural language or in a domain-specific language of the mission (also called

“task”) the robots must perform [20], [38], [43], [68].

- *Mission Specification*: a formulation of the mission in a logical language with a precise semantics [21]–[23], [26]–[28].
- *Mission Specification Problem*: the problem of generating a mission specification from a mission requirement.
- *Mission Specification Pattern*: a mapping between a recurrent mission-specification problem to a template solution and a description of the usage intent, known uses, and relationships to other patterns.
- *Mission Specification Pattern Catalog*: a collection of mission specification patterns organized in a hierarchy aiding at browsing and selecting patterns, in order to support decision making during mission specification.

We consider LTL (Linear Temporal Logic) [69] and CTL (Computation Tree Logic) [70], since they are commonly used to express mission specifications in the robotic domain and are utilized extensively by the community (e.g., [10], [17], [20]–[28]). A temporal logic specification can be used for several purposes, such as (i) for producing plans through the use of planners, (ii) for analysing the mission satisfaction through the use of model checkers, and (iii) to design a robotic application.

We now briefly recall the LTL and CTL syntax and semantics. Let  $\pi$  be a set of atomic propositions, LTL’s syntax is the following:

$$\text{(LTL)} \quad \phi ::= \tau \mid \neg\phi \mid \phi \vee \phi \mid \mathcal{X}\phi \mid \phi \mathcal{U} \phi \text{ where } \tau \in \pi.$$

The semantics of LTL is defined over an infinite sequence of truth assignments to the propositions  $\pi$ . The formula  $\mathcal{X}\phi$  expresses that  $\phi$  is true in the next position in a sequence, and the formula  $\phi_1 \mathcal{U} \phi_2$  expresses the property that  $\phi_1$  is true until  $\phi_2$  holds.

CTL’s syntax is the following:

$$\begin{aligned} \text{(CTL)} \quad \phi &::= \tau \mid \neg\phi \mid \phi \vee \phi \mid \exists\Phi \mid \forall\Phi, \text{ where} \\ \Phi &::= \mathcal{X}\phi \mid \phi \mathcal{U} \phi \text{ and } \tau \in \pi. \end{aligned}$$

CTL allows the specification of properties that predicate on a branching sequence of assignments. Specifically, when a position of a sequence has several successors, CTL enables the specification of a property that must hold for all or one of the paths that start from that position. For this reason, CTL includes two types of formulae: *state* formulae that must hold in one position of the sequence and *path* formulae that predicate on paths that start from a position. The operator  $\forall$  (resp.  $\exists$ ) asserts that  $\phi$  must hold on all paths (resp. on one path) starting from the current position, while  $\mathcal{X}$  and  $\mathcal{U}$  are defined as for LTL.

### III. METHODOLOGY

We derived our pattern catalog in three main steps.

**Collection of Mission Requirements.** We collected mission requirements from scientific papers in the field of robotics. We additionally considered the software engineering literature, but noted a general absence of robotic mission specifications. We chose major venues based on consultation with domain experts and by considering their impact factor. Specifically, we analyzed mission specifications published in the four major [71]

Table I  
PAPERS AND (REQUIREMENTS) ANALYZED PER VENUE AND YEAR

Robotics Venue	2017	2016	2015	2014	2013	Total
Intl. Conf. Robotics & Autom.	9(14)	16(11)	17(18)	27(22)	16(15)	85(80)
Intl. J. of Robotics Research	4(8)	13(12)	12(11)	13(8)	17(12)	59(51)
Trans. on Robotics	2(6)	12(9)	5(1)	8(2)	4(2)	31(20)
Intl. Conf. on Int. Robots & Sys.	10(23)	55(26)	13(8)	20(16)	33(21)	131(94)

robotics venues over the last five years, in line with similar studies for pattern identification [49], [54], [55]. We analyzed all papers published within a venue with two inclusion criteria (considered in order): (i) the paper title implies some notion of robotic movement-related concept, (ii) the paper contains at least one formulation of a mission requirement involving a robot that concerns movement. When the paper contained more than one mission requirement, each was considered separately.

Altogether we obtained 306 papers, through which, matching our inclusion criteria, we obtained 245 mission requirements. Table I shows the venues included in our analysis, together with the number of scientific publications and mission requirements obtained per year. The considered software engineering venues (ICSE, FSE, and ASE) are not present, since they did not contain any paper matching the inclusion criteria.

**Identification of Mission Specification Problems.** We identified these problems as follows.

(STEP.1) We divided the collected mission requirements among two of the authors, who labeled them with keywords that describe the mission specification problems they describe. For example, the mission requirement “The robot has to autonomously patrol the site and measure the state of valve levers and dial gauges at four checkpoints in order to decide if some machines need to be shut down” (occurring in Schillinger et al. [72]) was associated with the keywords “patrol,” since the robot has to patrol the site, and “instantaneous reaction,” since when a valve is reached its level must be checked.

(STEP.2) We created a graph structure representing semantic relations between keywords. Each keyword is associated with a node of the graph structure. Two nodes were connected if their keywords identify two similar mission specification problems. For example, the keywords “visit” and “reach” are related since in both cases the robot has to visit/reach a location.

(STEP.3) Since our interest was not a mere classification of actions and movements that are executed by a robots, but rather detecting mission specification problems that concern how actions and movements are executed by a robot behavior over time, nodes that contain keywords that only refer to actions are removed (e.g., balance).

(STEP.4) Nodes that were connected through edges and contained keywords that identify to the same mission specification problem, e.g., visit and reach, were merged.

(STEP.5) We organized the mission specification problems into a catalog represented through a tree structure that facilitates browsing among mission specification problems.

The material produced in these steps can be found in our online appendix [66].

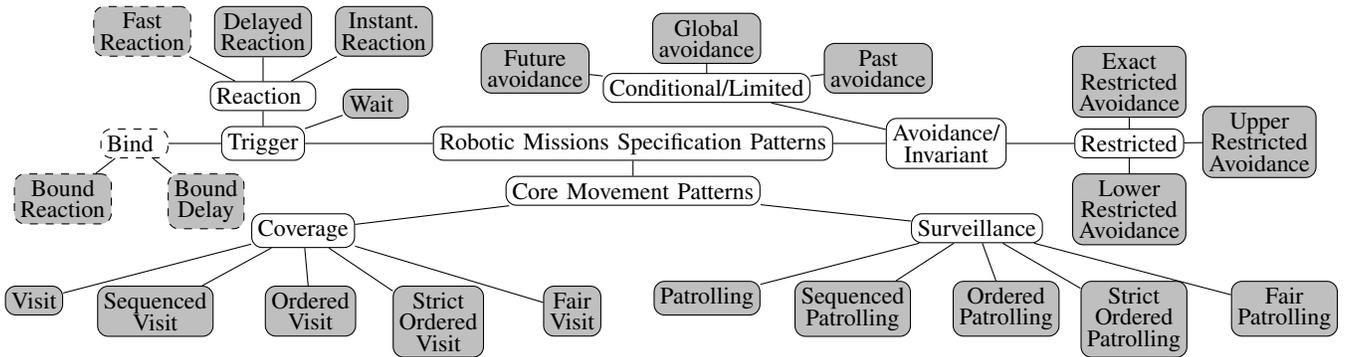


Figure 1. Mission specification pattern catalog. Filled nodes: patterns, non-filled nodes: categories.

**Pattern Formulation.** We formulated patterns by following established practices in the literature [49], [54], [56]. A pattern is characterized by (i) a name; (ii) a statement that captures the pattern intent (i.e., the mission requirement); (iii) a template instance of the mission specification in LTL and CTL; (iv) variations describing possible minor changes that can be applied to the pattern; (v) examples of known uses; (vi) relationships of the pattern to others and; (vii) occurrences of the pattern in literature. For each LTL pattern we also designed a Büchi Automaton (BA) that unambiguously describes the behaviors of the system allowed by the mission specification. The mission specification was designed by consulting specifications encoding requirements already present in the papers surveyed, by crosschecking them, and consulting specification patterns already proposed in the software-engineering literature [56]. If the proposed specification was related (or corresponded) with one of an already existing pattern, we indicated this in the relationships of the pattern to others, meaning that the pattern presented in literature is useful also to solve the identified mission specification problem.

#### IV. MISSION SPECIFICATION PATTERNS

In this section, we present our catalog of mission specification patterns and briefly present one of them (Section IV-A). We also present PsAIM, a tool that supports developers in systematic mission design. PsAIM supports the description of mission requirements through the proposed patterns and the automatic generation of mission specifications (Section IV-B).

##### A. Mission Specification Pattern Catalog

Our catalog of robotic mission specification patterns comprises 22 patterns organized into a pattern tree as illustrated in Fig. 1. Leaves of the tree represent mission specification patterns. Intermediate nodes facilitate browsing within the hierarchy and aid pattern selection and decision making. Patterns identified by following the procedure described in Sec. III are graphically indicated with a solid border.

Due to space limits, we provide a high-level description of all patterns identified, examples of application, and the corresponding LTL mission specifications. The interested reader may refer to our online appendix [66], which contains additional

examples, occurrences of patterns in the literature, relations among the patterns and additional CTL mission specifications.

**Preliminaries.** To aid comprehension of behavior and facilitate precise pattern definitions, we introduce the following notation. Given a finite set of locations  $L = \{l_1, l_2, \dots, l_n\}$  and robots  $R = \{r_1, r_2, \dots, r_n\}$ ,  $PL = \{r_x \text{ in } l_y \mid r_x \in R \text{ and } l_y \in L\}$  is a set of location propositions, each indicating that a robot  $r_x$  is in a specific location  $l_y$  of the environment. Given a finite set of conditions of the environment  $C = \{c_1, c_2, \dots, c_m\}$ , we indicate as  $PE = \{s_1, s_2, \dots, s_m\}$  a set of propositions such that  $s_i \in PE$  is true if and only if condition  $c_i$  holds. Given a finite set of actions that the robots can perform  $A = \{a_1, a_2, \dots, a_m\}$ , we indicate as  $PA = \{r_x \text{ exec } a_y \mid r_x \in R \text{ and } a_y \in A\}$  a set of propositions such that  $r_x \text{ exec } a_y$  is true if and only if action  $a_y$  is performed by robot  $r_x$ . We define the set of propositions  $M$  of a robotic application as  $PL \cup PE \cup PA$ . A trace is an infinite sequence  $M_x \rightarrow M_y \rightarrow M_z \dots$  where  $M_x, M_y, M_z \subseteq M$  indicate a trace in which  $M_z$  holds after  $M_y$ , and  $M_y$  holds after  $M_x$ . For example,  $\{r_1 \text{ in } l_1\} \rightarrow \{r_1 \text{ in } l_2, c_1\} \rightarrow \{c_2, r_2 \text{ exec } a_1\} \dots$  is a trace where the element in position 1 of the trace indicates that the robot  $r_1$  is in location  $l_1$ , then the element in position 2 indicates that the robot  $r_1$  is in location  $l_2$  and condition  $c_1$  holds (indicating, for example, that an obstacle is detected), and then the element in position 3 indicates that condition  $c_2$  holds and robot  $r_2$  is executing action  $a_1$ . In the following, with a slight abuse of notation, when a set is a singleton we will omit brackets. We use the notation  $(M_x \rightarrow \dots \rightarrow M_y)^\omega$ , where  $M_x, \dots, M_y \subseteq M$ , to indicate a sequence  $M_x \rightarrow \dots \rightarrow M_y$  that occurs infinitely. We use the notation  $l_\#$  to indicate any location, e.g.,  $r_1 \text{ in } l_1 \rightarrow r_1 \text{ in } l_\# \rightarrow r_1 \text{ in } l_2$  indicates that a robot  $r_1$  visits location  $l_1$ , afterwards any location, and then location  $l_2$ . We use the notation  $l_\# \setminus K$ , where  $K \subset M$ , to indicate any possible location not in  $K$ , e.g.,  $r_1 \text{ in } l_1 \rightarrow r_1 \text{ in } l_\# \setminus \{l_3\} \rightarrow r_1 \text{ in } l_2$  indicates that  $r_1$  visits  $l_1$ , then any location except  $l_3$  is visited, and finally  $l_2$ .

**Patterns.** Patterns are organized in three main groups – core movement (Table II), triggers (Table III), and avoidance (Table III), explained in the following. For simplicity, in Tables II and III, we assume that a single robot is considered during the mission specification and we use the notation  $l_x$  as

Table II  
CORE MOVEMENT PATTERNS

	Description	Example	Formula ( $l_1, l_2, \dots$ are location propositions)
Visit	Visit a set of locations in an unspecified order.	Locations $l_1, l_2$ , and $l_3$ must be visited. $l_1 \rightarrow l_4 \rightarrow l_3 \rightarrow l_1 \rightarrow l_4 \rightarrow l_2 \rightarrow (l_{\#})^\omega$ is an example trace that satisfies the mission requirement.	$\bigwedge_{i=1}^n \mathcal{F}(l_i)$
Sequenced Visit	Visit a set of locations in sequence, one after the other.	Locations $l_1, l_2, l_3$ must be covered following this sequence. The trace $l_1 \rightarrow l_4 \rightarrow l_3 \rightarrow l_1 \rightarrow l_4 \rightarrow l_2 \rightarrow (l_{\# \setminus 3})^\omega$ violates the mission since $l_3$ does not follow $l_2$ . The trace $l_1 \rightarrow l_3 \rightarrow l_1 \rightarrow l_2 \rightarrow l_4 \rightarrow l_3 \rightarrow (l_{\#})^\omega$ satisfies the mission requirement.	$\mathcal{F}(l_1 \wedge \mathcal{F}(l_2 \wedge \dots \mathcal{F}(l_n)))$
Ordered Visit	Sequence visit does not forbid to visit a successor location before its predecessor, but only that after the predecessor is visited the successor is also visited. Ordered visit forbids a successor to be visited before its predecessor.	Locations $l_1, l_2, l_3$ must be covered following this order. The trace $l_1 \rightarrow l_3 \rightarrow l_1 \rightarrow l_2 \rightarrow l_3 \rightarrow (l_{\#})^\omega$ does not satisfy the mission requirement since $l_3$ precedes $l_2$ . The trace $l_1 \rightarrow l_4 \rightarrow l_1 \rightarrow l_2 \rightarrow l_4 \rightarrow l_3 \rightarrow (l_{\#})^\omega$ satisfies the mission requirement.	$\mathcal{F}(l_1 \wedge \mathcal{F}(l_2 \wedge \dots \mathcal{F}(l_n)))$ $\bigwedge_{i=1}^{n-1} (\neg l_{i+1}) \mathcal{U} l_i$
Strict Ordered Visit	Ordered visit pattern does not avoid a predecessor location to be visited multiple times before its successor. Strict ordered visit forbids this behavior.	Locations $l_1, l_2, l_3$ must be covered following the strict order $l_1, l_2, l_3$ . The trace $l_1 \rightarrow l_4 \rightarrow l_1 \rightarrow l_2 \rightarrow l_4 \rightarrow l_3 \rightarrow (l_{\#})^\omega$ does not satisfy the mission requirement since $l_1$ occurs twice before $l_2$ . The trace $l_1 \rightarrow l_4 \rightarrow l_2 \rightarrow l_4 \rightarrow l_3 \rightarrow (l_{\#})^\omega$ satisfies the mission requirement.	$\mathcal{F}(l_1 \wedge \mathcal{F}(l_2 \wedge \dots \mathcal{F}(l_n)))$ $\bigwedge_{i=1}^{n-1} (\neg l_{i+1}) \mathcal{U} l_i$ $\bigwedge_{i=1}^{n-1} (\neg l_i) \mathcal{U} (l_i \wedge \mathcal{X}(\neg l_i \mathcal{U} l_{i+1}))$
Fair Visit	The difference among the number of times locations within a set are visited is at most one.	Locations $l_1, l_2, l_3$ must be covered in a fair way. The trace $l_1 \rightarrow l_4 \rightarrow l_1 \rightarrow l_3 \rightarrow l_1 \rightarrow l_4 \rightarrow l_2 \rightarrow (l_{\# - \{1,2,3\}})^\omega$ does not perform a fair visit since it visits $l_1$ three times while $l_2$ and $l_3$ are visited once. The trace $l_1 \rightarrow l_4 \rightarrow l_3 \rightarrow l_1 \rightarrow l_4 \rightarrow l_2 \rightarrow l_2 \rightarrow l_4 \rightarrow (l_{\# \setminus \{1,2,3\}})^\omega$ performs a fair visit since it visits locations $l_1, l_2$ , and $l_3$ twice.	$\bigwedge_{i=1}^n \mathcal{F}(l_i)$ $\bigwedge_{i=1}^n \mathcal{G}(l_i \rightarrow \mathcal{X}(\neg l_i) \mathcal{W} l_{(i+1)\%n})$
Patrolling	Keep visiting a set of locations, but not in a particular order.	Locations $l_1, l_2, l_3$ must be surveilled. The trace $l_1 \rightarrow l_4 \rightarrow l_3 \rightarrow l_1 \rightarrow l_4 \rightarrow l_2 \rightarrow (l_2 \rightarrow l_3 \rightarrow l_1)^\omega$ ensures that the mission requirement is satisfied. The trace $l_1 \rightarrow l_2 \rightarrow l_3 \rightarrow (l_1 \rightarrow l_3)^\omega$ represents a violation, since $l_2$ is not surveilled.	$\bigwedge_{i=1}^n \mathcal{G} \mathcal{F}(l_i)$
Sequenced Patrolling	Keep visiting a set of locations in sequence, one after the other.	Locations $l_1, l_2, l_3$ must be patrolled in sequence. The trace $l_1 \rightarrow l_4 \rightarrow l_3 \rightarrow l_1 \rightarrow l_4 \rightarrow l_2 \rightarrow (l_1 \rightarrow l_2 \rightarrow l_3)^\omega$ satisfies the mission requirement since globally any $l_1$ will be followed by $l_2$ and $l_2$ by $l_3$ . The trace $l_1 \rightarrow l_4 \rightarrow l_3 \rightarrow l_1 \rightarrow l_4 \rightarrow l_2 \rightarrow (l_1 \rightarrow l_3)^\omega$ violates the mission requirement since after visiting $l_1$ , the robot does not visit $l_2$ .	$\mathcal{G}(\mathcal{F}(l_1 \wedge \mathcal{F}(l_2 \wedge \dots \mathcal{F}(l_n))))$
Ordered Patrolling	Sequence patrolling does not forbid to visit a successor location before its predecessor. Ordered patrolling ensures that (after a successor is visited) the successor is not visited (again) before its predecessor.	Locations $l_1, l_2$ , and $l_3$ must be patrolled following the order $l_1, l_2$ , and $l_3$ . The trace $l_1 \rightarrow l_4 \rightarrow l_3 \rightarrow l_1 \rightarrow l_4 \rightarrow l_2 \rightarrow (l_1 \rightarrow l_2 \rightarrow l_3)^\omega$ violates the mission requirement since $l_3$ precedes $l_2$ . The trace $l_1 \rightarrow l_1 \rightarrow l_2 \rightarrow l_4 \rightarrow l_4 \rightarrow l_3 \rightarrow (l_1 \rightarrow l_2 \rightarrow l_3)^\omega$ satisfies the mission requirement	$\mathcal{G}(\mathcal{F}(l_1 \wedge \mathcal{F}(l_2 \wedge \dots \mathcal{F}(l_n))))$ $\bigwedge_{i=1}^{n-1} (\neg l_{i+1}) \mathcal{U} l_i$ $\bigwedge_{i=1}^n \mathcal{G}(l_{(i+1)\%n} \rightarrow \mathcal{X}(\neg l_{(i+1)\%n}) \mathcal{U} l_i)$
Strict Ordered Patrolling	Ordered patrolling pattern does not avoid a predecessor location to be visited multiple times before its successor. Strict Ordered Patrolling ensures that, after a predecessor is visited, it is not visited again before its successor.	Locations $l_1, l_2, l_3$ must be patrolled following the strict order $l_1, l_2$ , and $l_3$ . The trace $l_1 \rightarrow l_4 \rightarrow l_1 \rightarrow l_2 \rightarrow l_4 \rightarrow l_3 \rightarrow (l_1 \rightarrow l_2 \rightarrow l_3)^\omega$ violates the mission requirement since $l_1$ is visited twice before $l_2$ . The trace $l_1 \rightarrow l_4 \rightarrow l_2 \rightarrow l_4 \rightarrow l_3 \rightarrow (l_1 \rightarrow l_2 \rightarrow l_3)^\omega$ satisfies the mission requirement.	$\mathcal{G}(\mathcal{F}(l_1 \wedge \mathcal{F}(l_2 \wedge \dots \mathcal{F}(l_n))))$ $\bigwedge_{i=1}^{n-1} (\neg l_{i+1}) \mathcal{U} l_i$ $\bigwedge_{i=1}^n \mathcal{G}(l_{(i+1)\%n} \rightarrow \mathcal{X}(\neg l_{(i+1)\%n}) \mathcal{U} l_i)$ $\bigwedge_{i=1}^{n-1} \mathcal{G}((l_i) \rightarrow \mathcal{X}(\neg l_i \mathcal{U} l_{(i+1)\%n}))$
Fair Patrolling	Keep visiting a set of locations and ensure that the difference among the number of times locations within a set are visited is at most one.	Locations $l_1, l_2$ , and $l_3$ must be fair patrolled. The trace $l_1 \rightarrow l_4 \rightarrow l_3 \rightarrow l_1 \rightarrow l_4 \rightarrow l_2 \rightarrow (l_1 \rightarrow l_2 \rightarrow l_1 \rightarrow l_3)^\omega$ violates the mission requirements since the robot patrols $l_1$ more than $l_2$ and $l_3$ . The trace $l_1 \rightarrow l_4 \rightarrow l_3 \rightarrow l_4 \rightarrow l_2 \rightarrow l_4 \rightarrow (l_1 \rightarrow l_2 \rightarrow l_3)^\omega$ satisfies the mission requirement since locations $l_1, l_2$ , and $l_3$ are patrolled fairly.	$\bigwedge_{i=1}^n \mathcal{G}(\mathcal{F}(l_i))$ $\bigwedge_{i=1}^n \mathcal{G}(l_i \rightarrow \mathcal{X}(\neg l_i) \mathcal{W} l_{(i+1)\%n})$

shortcut for  $r_1$  in  $l_x$ . The examples assume that the environment is made by four locations, namely  $l_1, l_2, l_3$ , and  $l_4$ .

*Core movement patterns.* How robots should move within an environment can be divided in two major categories representing locations' coverage and locations' surveillance. Coverage patterns require a robot to reach a set of locations of the environment. Surveillance patterns require a robot to *keep* reaching a set of locations of the environment.

*Avoidance patterns.* Robot movements may be constrained in order to avoid occurrence of some behavior (Table III). Avoidance may reflect a condition or be a bound to the

occurrence of some event. *Conditional avoidance* generally holds globally (i.e., for the entire behavior) and applies when avoidance of locations or obstacles is sought that depends on some condition. For example, a cleaning robot may avoid visiting locations that have been already cleaned. In the *restricted avoidance* case, avoidance does not hold globally but accounts for a number of occurrences of an avoidance case. Depending on the number of occurrences being a maximum, minimum or exact number, *upper*, *exact* or *lower* restricted avoidance is yielded. For example, a cleaning robot may avoid cleaning a room more than three times.

Table III  
AVOIDANCE AND TRIGGER PATTERNS.

	Description	Example	Formula
<i>Past avoidance</i>	A condition has been fulfilled in the past.	If the robot enters location $l_1$ , then it should have not visited location $l_2$ before. The trace $l_3 \rightarrow l_4 \rightarrow l_1 \rightarrow l_2 \rightarrow l_4 \rightarrow l_3 \rightarrow (l_2 \rightarrow l_3)^\omega$ satisfies the mission requirement since location $l_2$ is not entered before location $l_1$ .	$(\neg(l_1))\mathcal{U}p$ , where $l_1 \in L$ and $p \in M$
<i>Global avoidance</i>	An avoidance condition globally holds throughout the mission.	The robot should avoid entering location $l_1$ . Trace $l_3 \rightarrow l_4 \rightarrow l_3 \rightarrow l_2 \rightarrow l_4 \rightarrow l_3 \rightarrow (l_3 \rightarrow l_2 \rightarrow l_3)^\omega$ , satisfies the mission requirement since the robot never enters $l_1$ .	$\mathcal{G}(\neg(l_1))$ , where $l_1 \in L$
<i>Future avoidance</i>	After the occurrence of an event, avoidance has to be fulfilled.	If the robot enters $l_1$ , then it should avoid entering $l_2$ in the future. The trace $l_3 \rightarrow l_4 \rightarrow l_3 \rightarrow l_1 \rightarrow l_4 \rightarrow l_3 \rightarrow (l_3 \rightarrow l_2 \rightarrow l_3)^\omega$ does not satisfy the mission requirement since $l_2$ is entered after $l_1$ .	$\mathcal{G}(c \rightarrow (\mathcal{G}(\neg(l_1))))$ , where $c \in M$ and $l_1 \in PL$
<i>Upper Rest. Avoidance</i>	A restriction on the maximum number of occurrences is desired.	A robot has to visit $l_1$ at most 3 times. The trace $l_1 \rightarrow l_4 \rightarrow l_1 \rightarrow l_3 \rightarrow l_1 \rightarrow l_4 \rightarrow l_1 \rightarrow (l_3)^\omega$ violates the mission requirement since $l_1$ is visited four times. The trace $l_4 \rightarrow l_3 \rightarrow l_1 \rightarrow l_2 \rightarrow l_4 \rightarrow (l_3)^\omega$ satisfies the mission requirement.	$\neg \mathcal{F}(l_1 \wedge \underbrace{\mathcal{X}(\mathcal{F}(l_1 \wedge \dots \mathcal{X}(\mathcal{F}(l_1))))}_n)$ , where $l_1 \in L$
<i>Lower Rest. Avoidance</i>	A restriction on the minimum number of occurrences is desired.	A robot to enter location $l_1$ at least 3 times. The trace $l_4 \rightarrow l_3 \rightarrow l_2 \rightarrow l_2 \rightarrow l_4 \rightarrow (l_3)^\omega$ violates the mission requirement since location 1 is never entered. The trace $l_1 \rightarrow l_4 \rightarrow l_3 \rightarrow l_1 \rightarrow l_4 \rightarrow l_1 \rightarrow (l_3)^\omega$ satisfies the mission requirement.	$\mathcal{F}(l_1 \wedge \underbrace{\mathcal{X}(\mathcal{F}(l_1 \wedge \dots \mathcal{X}(\mathcal{F}(l_1))))}_n)$ , where $l_1 \in L$
<i>Exact Rest. Avoidance</i>	The number of occurrences desired is an exact number.	A robot must enter location $l_1$ exactly 3 times. The trace $l_4 \rightarrow l_3 \rightarrow l_2 \rightarrow l_2 \rightarrow l_4 \rightarrow (l_3)^\omega$ violates the mission requirement. The trace $l_1 \rightarrow l_4 \rightarrow l_3 \rightarrow l_1 \rightarrow l_4 \rightarrow l_1 \rightarrow (l_3)^\omega$ satisfies the mission requirement since location $l_1$ is entered exactly 3 times.	$(\neg(l_1))\mathcal{U}(l_1 \wedge (\mathcal{X}(\neg(l_1))\mathcal{U}(l_1 \dots \wedge (\mathcal{X}(\neg(l_1))\mathcal{U}(l_1 \dots \wedge (\mathcal{X}(\mathcal{G}(\neg(l_1))))))))))$ , where $l_1 \in L$
<i>Inst. Reaction</i>	The occurrence of a stimulus instantaneously triggers a counteraction.	When location $l_2$ is reached action $a$ must be executed. The trace $l_1 \rightarrow l_3 \rightarrow \{l_2, a\} \rightarrow \{l_2, a\} \rightarrow l_4 \rightarrow (l_3)^\omega$ satisfies the mission requirement since when location $l_2$ is entered condition $a$ is performed. The trace $l_1 \rightarrow l_3 \rightarrow l_2 \rightarrow \{l_1, a\} \rightarrow l_4 \rightarrow (l_3)^\omega$ does not satisfy the mission requirement since when $l_2$ is reached $a$ is not executed.	$\mathcal{G}(p_1 \rightarrow p_2)$ , where $p_1 \in M$ and $p_2 \in PL \cup PA$
<i>Delayed Reaction</i>	The occurrence of a stimulus triggers a counteraction some time later	When $c$ occurs the robot must start moving toward location $l_1$ , and $l_1$ is subsequently finally reached. The trace $l_1 \rightarrow l_3 \rightarrow \{l_2, c\} \rightarrow l_1 \rightarrow l_4 \rightarrow (l_3)^\omega$ satisfies the mission requirement, since after $c$ occurs the robot starts moving toward location $l_1$ , and location $l_1$ is finally reached. The trace $l_1 \rightarrow l_1 \rightarrow \{l_2, c\} \rightarrow l_3 \rightarrow (l_3)^\omega$ does not satisfy the mission requirement since $c$ occurs when the robot is in $l_2$ , and $l_1$ is not finally reached.	$\mathcal{G}(p_1 \rightarrow \mathcal{F}(p_2))$ , where $p_1 \in M$ and $p_2 \in PL \cup PA$
<i>Prompt Reaction</i>	The occurrence of a stimulus triggers a counteraction promptly, i.e. in the next time instant.	If $c$ occurs $l_1$ is reached in the next time instant. The trace $l_1 \rightarrow l_3 \rightarrow \{l_2, c\} \rightarrow l_1 \rightarrow l_4 \rightarrow (l_3)^\omega$ satisfies the mission requirement, since after $c$ occurs $l_1$ is reached within the next time instant. The trace $l_1 \rightarrow l_3 \rightarrow \{l_2, c\} \rightarrow l_4 \rightarrow l_1 \rightarrow (l_3)^\omega$ does not satisfy the mission requirement.	$\mathcal{G}(p_1 \rightarrow \mathcal{X}(p_2))$ , where $p_1 \in M$ and $p_2 \in PL \cup PA$
<i>Bound Reaction</i>	A counteraction must be performed every time and only when a specific location is entered.	Action $a_1$ is bound though a delay to location $l_1$ . The trace $l_1 \rightarrow l_3 \rightarrow \{l_2, c\} \rightarrow \{l_1, a_1\} \rightarrow l_4 \rightarrow \{l_1, a_1\} \rightarrow (l_3)^\omega$ satisfies the mission requirement. The trace $l_1 \rightarrow l_3 \rightarrow \{l_2, c\} \rightarrow \{l_1, a_1\} \rightarrow \{l_4, a_1\} \rightarrow \{l_1, a_1\} \rightarrow (l_3)^\omega$ does not satisfy the mission requirement since $a_1$ is executed in location $l_4$ .	$\mathcal{G}(p_1 \leftrightarrow p_2)$ , where $p_1 \in M$ and $p_2 \in PL \cup PA$
<i>Bound Delay</i>	A counteraction must be performed, in the next time instant, every time and only when a specific location is entered	Action $a_1$ is bound to location $l_1$ . The trace $l_1 \rightarrow l_3 \rightarrow \{l_2, c\} \rightarrow \{l_1\} \rightarrow \{l_4, l_1\} \rightarrow \{l_1\} \rightarrow \{l_4, a_1\} \rightarrow (l_3)^\omega$ satisfies the mission requirement. The trace $l_1 \rightarrow l_3 \rightarrow \{l_2, c\} \rightarrow \{l_1\} \rightarrow \{l_4, l_1\} \rightarrow \{l_1, a_1\} \rightarrow \{l_4\} \rightarrow (l_3)^\omega$ does not satisfy the mission requirement.	$\mathcal{G}(p_1 \leftrightarrow \mathcal{X}(p_2))$ , where $p_1 \in M$ and $p_2 \in PL \cup PA$
<i>Wait</i>	Inaction is desired till a stimulus occurs.	The robot remains in location $l_1$ until condition $c$ is satisfied. The trace $l_1 \rightarrow l_3 \rightarrow \{l_2, c\} \rightarrow l_1 \rightarrow l_4 \rightarrow (l_3)^\omega$ violates the mission requirement since the robot left $l_1$ before condition $c$ is satisfied. The trace $l_1 \rightarrow \{l_1, c\} \rightarrow l_2 \rightarrow l_1 \rightarrow l_4 \rightarrow (l_3)^\omega$ satisfies the mission requirement.	$(l_1)\mathcal{U}(p)$ , where $l_1 \in L$ and $p \in PE \cup PA$

*Trigger patterns.* Robot's reactive behaviour based on stimuli, or robot's inaction until a stimulus occurs are expressed as trigger patterns in Table III.

As an example, the definition of the *Strict Ordered Patrolling* mission specification pattern is presented in Fig. 2. The patterns in detail are available in our online appendix [66].

### B. Specification Pattern Tool Support

We used the proposed pattern catalog to express robotic missions requirements and to automatically generate their mission specifications. To support developers in mission design we implemented the tool PsAIM [62], which allows creating complex mission requirements by composing patterns with

simple operators. PsAIM transforms mission requirements (i.e., composed patterns) into mission specifications in LTL or CTL. Figure 3 illustrates the components of PsAIM.

PsAIM provides a GUI ① that allows the definition of robotic missions requirements through a structured English grammar, which uses patterns as basic building blocks and AND and OR logic operators to compose these patterns. The structured English grammar and the PsAIM tool are provided in our online appendix [66]. The SE2PT component extracts from a mission requirement the set of patterns that are composed through the AND and OR operators ②. The PT2LTL ③ and PT2CTL ④ components automatically generate LTL and CTL specifications from these patterns.

**Name:** Strict Ordered Patrolling

**Intent:** A robot must patrol a set of locations following a strict sequence ordering. Such locations can be, e.g., areas in a building to be surveyed.

**Template:** The following formula encodes the mission in LTL for  $n$  locations and a robot  $r$  ( $\%$  is the modulo arithmetic operator):

$$\bigwedge_{i=1}^n \mathcal{G}(\mathcal{F}(l_1 \wedge \mathcal{F}(l_2 \wedge \dots \mathcal{F}(l_n)))) \bigwedge_{i=1}^{n-1} ((\neg l_{i+1}) U l_i) \bigwedge_{i=1}^n \mathcal{G}(l_{(i+1)\%n} \rightarrow \mathcal{X}((\neg l_{(i+1)\%n}) U l_i))$$

Example with two locations.

$$\mathcal{G}(\mathcal{F}(l_1 \wedge \mathcal{F}(l_2))) \wedge ((\neg l_2) U l_1) \wedge \mathcal{G}(l_2 \rightarrow \mathcal{X}((\neg l_2) U l_1)) \wedge \mathcal{G}(l_1 \rightarrow \mathcal{X}((\neg l_1) U l_2))$$

where  $l_1$  and  $l_2$  are expressions that indicate that a robot  $r$  is in locations  $l_1$  and  $l_2$ , respectively.

**Variations:** A developer may want to allow traces in which sequences of *consecutive*  $l_1$  ( $l_2$ ) are allowed, that is strict ordering is applied on sequences of non consecutive  $l_1$  ( $l_2$ ). In this case, traces in the form  $l_1 \rightarrow (\rightarrow l_1 \rightarrow l_1 \rightarrow l_3 \rightarrow l_2)^\omega$  are admitted, while traces in the form  $l_1 \rightarrow (\rightarrow l_1 \rightarrow l_3 \rightarrow l_1 \rightarrow l_2)^\omega$  are not admitted. This variation can be encoded using the following specification:

$$\mathcal{G}(\mathcal{F}(l_1 \wedge \mathcal{F}(l_2))) \wedge ((\neg l_2) U l_1) \wedge \mathcal{G}((l_2 \wedge \mathcal{X}(\neg l_2)) \rightarrow \mathcal{X}((\neg l_2) U l_1)) \wedge \mathcal{G}((l_1 \wedge \mathcal{X}(\neg l_1)) \rightarrow \mathcal{X}((\neg l_1) U l_2))$$

This specification allows for sequences of consecutive  $l_1$  ( $l_2$ ) since the left side of the implication  $l_1 \wedge \mathcal{X}(\neg l_1)$  ( $l_2 \wedge \mathcal{X}(\neg l_2)$ ) is only triggered when  $l_1$  ( $l_2$ ) is exited.

**Examples and Known Uses:** A common usage example of the Strict Ordered Patrolling pattern is a scenario where a robot is performing surveillance in a building during night hours. Strict Sequence Patrolling and Avoidance often go together. Avoidance patterns are used to force robots to avoid obstacles as they guard a location. Triggers can also be used in combination with the Strict Sequence Patrolling pattern to specify conditions upon which Patrolling should start or stop.

**Relationships:** The Strict Ordered Patrolling pattern is a specialisation of the Ordered Patrolling pattern, forcing the strict ordering.

**Occurrences:** Smith et. al. [73] proposed a mission specification forcing a robot to not visit a location twice in a row before a target location is reached.

Figure 2. The Strict Ordered Patrolling pattern

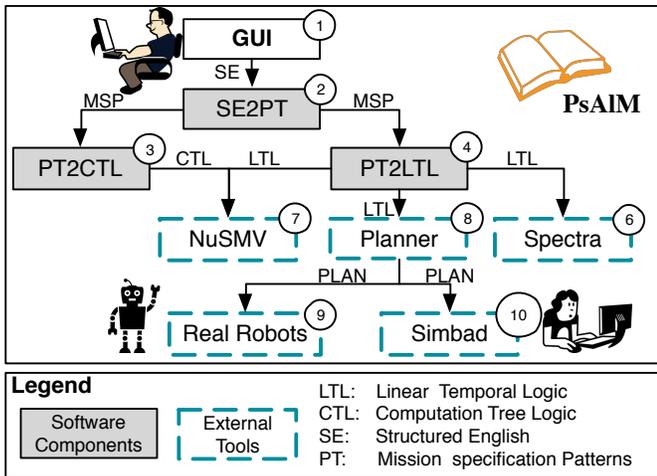


Figure 3. Main components of the PsAIM tool

The produced LTL specifications can be used in different ways – three possible usages are presented in Fig. 3. The LTL formulae are (i) fed into an existing planner and used to generate plans that satisfy the mission specification (5); (ii) converted into Deterministic Büchi automata used as input to the widely used Spectra [63] robotic application modeling tool (6); and (iii) converted into the NuSMV [64] input language to be used as input for model checking (7). The plans produced using the planner are (i) used as inputs by the Simbad [65] simulation package (10), which is an autonomous robot simulation package for education and research; and (ii) performed by actual real robots (9), as also illustrated in the following section. Produced CTL specifications are also converted into the NuSMV [64] input language to be used as

input for model checking (7).

## V. EVALUATION

Our evaluation addressed the following two questions. **RQ1:** How *effective* is the pattern catalog in capturing mission requirements and producing mission specifications? **RQ2:** Are the proposed mission specifications *correct*?

**Coverage of Real-World Missions (RQ1).** We investigated (i) how the pattern catalog supports the specification of mission requirements and (ii) how the pattern catalog reduces ambiguities in mission requirements.

*Exp1.* We checked how the pattern catalog supports the formulation of mission requirements (and the generation of mission specifications) in real-world robotic scenarios. To this end, we defined five scenarios (Table IV) in collaboration with our industrial partners (BOSCH and PAL Robotics).

The pattern catalog supported the creation of mission requirements using the patterns listed in Table IV for the different scenarios. In all the scenarios, PsAIM allowed the automatic creation of LTL mission specifications from the mission requirements without any human intervention. The mission specifications were then executed by the robots by relying on existing planners (see Fig. 3). Videos of the robots performing the described missions are available in our dedicated website [66]. The pattern catalog effectively supports the creation of mission requirements and specifications in realistic, industry-sourced scenarios.

*Exp2.* We collected mission requirements in natural language from available requirements produced from Spectra [63] and LTLMoP [39], [42]. Spectra is a tool that supports the design of the robotic applications. LTLMoP is a software package designed to assist in the development, implementation, and testing of high-level robot controllers. We checked how the

Table IV  
MISSION SPECIFICATION PATTERNS FOR *Exp1*. LABELS SC1, SC2, . . . SC5 IDENTIFY THE CONSIDERED SCENARIOS.

SC	Description	Patterns
SC1	A robot is deployed within a supermarket and reports about the absence of sold items within a set of locations (i.e. <i>l1</i> , <i>l2</i> , <i>l3</i> , and <i>l4</i> ). Furthermore, if in location <i>l4</i> (where water supplies are present) a human is detected, it has to perform a collaborative grasping action and help the human in placing new water supplies.	Ordered Patrolling, Instantaneous Reaction
SC2	Three robots are deployed within an hospital environment: a mobile platform (Summit [74]), a manipulator (PA10 [75]) and a mobile manipulator (Tiago [76]), identified in the following as MP, M and MM, respectively. The robot M is deployed in hospital storage; when items (e.g., towels) are needed by a nurse or doctors, M has to load them on the MP. MP should reach the location where the nurse is located. If the item is heavy (e.g., heavy medical equipment), MM should reach the location where the nurse is to help unloading the equipment. When MP and MM are not required for shipping items they are patrolling a set of locations to avoid unauthorized people entering restricted areas of the hospital (e.g., radiotherapy rooms).	Patrolling, Instantaneous Reaction, Ordered Visit, Wait
SC3	A robot is developed within a university building to deliver coffee to employees. The robot reaches the coffee machine, uses the coffee machine to prepare the coffee and delivers it to the employee.	Strict Ordered Visit, Instantaneous Reaction
SC4	A robot is deployed within a shop to check the presence of intruders during night time. It has to iteratively check for intruders and report on their presence	Patrolling, Instantaneous Reaction
SC5	A robot is deployed within a company to notify employees in presence of a fire alarm. If a fire is detected, the robot is send to different areas of the company to ask employees to leave the building.	Visit, Instantaneous Reaction

Table V  
RESULTS OF EXPERIMENT EXP2. LINES CONTAIN THE TOTAL NUMBER OF MISSION REQUIREMENTS (MR), THE NUMBER OF NOT EXPRESSIBLE (NE) AND AMBIGUOUS (A) MISSION REQUIREMENTS AND THE NUMBER OF CASES THAT LEAD TO A CONSENSUS (C) AND NO CONSENSUS (NC).

Spectra Robotic Application													
	1	2	3	4	5	6	7	8	9	10	11	MP	Total
MR	29	2	22	5	1	159	4	32	47	53	74	8	436
NE	3	0	0	0	0	47	0	0	7	1	8	0	66
A	3	0	2	1	0	35	0	10	12	32	7	0	102
C	13	0	11	2	1	29	4	8	11	8	20	5	112
NC	10	2	9	2	0	48	0	14	17	12	39	3	156

Table VI  
RESULTS OF EXPERIMENT EXP2. NUMBER OF OCCURRENCES OF EACH PATTERN IN THE CONSIDERED MISSION REQUIREMENTS.

Pattern	Occ	Pattern	Occ	Pattern	Occ	Pattern	Occ	Pattern	Occ
Visit	25	SeqVisit	1	OrdVisit	1	InstReact	127	GlobAvoid	25
PastAvoid	60	DelReact	50	Wait	3	FutAvoid	48	SeqVisit	1
StrictOrdPat	1	OrdVisit	1	ExactRest	1				

pattern catalog may have supported developers in the definition of the mission requirements.

In the case of Spectra, we used the Spectra files to extract mission requirements for robotic systems. In total, 11 robotic applications were considered. Note that mission requirements are realistic since they were finally executed with real robots [77]. We automatically extracted 428 mission requirements from the Spectra file. The number of mission requirements (MR) per robotic application is reported in Table V. In the case of LTLMoP, 8 requirements were extracted from the corresponding research papers [39], [42] (Table V MP column).

Each mission requirement was independently analyzed by two of the authors. The authors checked whether it is possible to express the mission requirement using the mission specification patterns. If one the authors stated that the requirement is not expressible the requirement is marked as not expressible (NE). The number of not expressible mission requirements

is presented in Table VI under the column with header NE. If at least one of the authors found the mission requirement is ambiguous she marked it with the flag *A*. Otherwise, the mission requirement is labeled with the mission specification patterns needed to express the mission requirement. Then, the mission specification patterns used to express the mission requirement are considered. If the authors used the same mission specification patterns to express the mission requirement, a consensus is reached. The number of mission requirements that leads to consensus (resp. no consensus) is indicated in the row labeled C (resp. NC). The number of occurrences of each pattern is indicated in Table VI.

The results show that most of the mission requirements (370 over 436) were expressible using the pattern catalog, which is a reasonable coverage for pattern catalog usage. The 66 mission requirements that are not covered suggested the introduction of new patterns identified in Fig. 1 with a dashed border. It also shows that the pattern catalog is effective in real case scenarios. In 102 cases the mission requirements were ambiguous, meaning that different interpretations can be given to the proposed mission requirement. In these cases, alternative combinations of patterns have been proposed by the authors to express the mission requirement. Each of these alternatives represents a possible way of expressing it in a non-ambiguous manner. In 156 cases, while the authors judged that the requirement was not ambiguous, different pattern combinations were proposed. The combinations of patterns encode possible ways of expressing the mission requirement in a non-ambiguous manner.

*Exp3*. We analyzed the mission specifications contained in the Spectra examples collected in Exp2. We collected 1216 distinct LTL mission specifications and we analyzed each of these specifications<sup>1</sup>. We verified whether it is possible to obtain the mission specifications starting from the proposed patterns, by performing the following steps.

<sup>1</sup>This number differs from the one of Exp2, since some specifications were not related with a mission requirement in the form of natural language.

Table VII  
RESULTS OF EXPERIMENT EXP3. PATTERN OCCURRENCE IN THE  
CONSIDERED MISSION SPECIFICATIONS.

		LTL		CTL
Pattern		Spectra	[67]	[67]
Step 1	Instantaneous reaction	318	0	0
	Visit	52	0	0
	Patrolling	0	1	0
	Strict Ordered Visit	0	9	18
	Wait	0	1	2
	Avoidance/Invariant	21	0	0
	Visit and Instantaneous reaction	18	0	0
	Strict Ordered Visit and Global Avoidance	0	0	1
	Reaction chain (chain of instantaneous reactions)	15	0	0
	Non matching	792	1	1
	Step 2	Init	127	-
Fast reaction		379	-	-
Binded reaction		36	-	-
Binded delay		27	-	-
Non matching for past		155	-	-
Step 3	Actual non matching	69	-	-
	Fast reaction	103	-	-
	Binded delay	26	-	-
	Actual non matching	26	-	-

(STEP.1) For each property we automatically checked whether it was an instance of a mission specification pattern or a simple combination of mission specification patterns. Results are shown in Table VII. Among 1216 mission specifications 424 were obtainable from the proposed patterns.

(STEP.2) We considered the properties that did not match any of the proposed patterns. 127 of these properties are simple statements on the initial state of the system (no temporal operator is used), and thus did not match any of the proposed patterns. 442 formulae concern properties that refer to variation of the trigger patterns that we have added to the pattern catalogue. 224 formulae still did not match any of the proposed patterns. After analysis, 155 among them were expressed using past temporal operators, which are not used in the mission specifications proposed in this work. In step 3 we checked whether these specifications might be reformulated without the past operators. 69 of these properties, while they can be rewritten using the proposed patterns, they are written as complex LTL formulae and thus they do not match any of our patterns or combination of them.

(STEP.3) We considered the 155 properties expressed using past temporal operators and we designed mission specifications for them. We found that 129 of the proposed LTL formulae match one of the proposed pattern, while 26 are complex LTL formulae that did not match any of the patterns. Thus, the final coverage of the proposed pattern catalog is 92%.

We then analyzed 13 mission specifications expressed in the form of LTL properties considered in [67] and 22 PCTL properties considered in [67], transformed in CTL by replacing the probabilistic operator ( $\mathcal{P}$ ) with the universal quantifier ( $\forall$ ). Given the small number of mission specifications we manually checked the presence of patterns in the formulae (Step 1 in Table VII). The results show that the pattern system was able to generate almost all mission specifications (1154 over 1251).

Table VIII  
RESULTS OF EXPERIMENTS *Exp4*, *Exp5* AND *Exp6*. FOR *Exp4* COLUMNS  
CONTAIN THE NUMBER OF TIMES A PLAN IS FOUND ( $\top$ ) AND NOT FOUND  
( $\perp$ ). FOR *Exp5* AND *Exp6* COLUMNS CONTAIN THE NUMBER OF TIMES THE  
MISSION REQUIREMENT IS SATISFIED ( $\top$ ) AND VIOLATED ( $\perp$ ).

Mission Requirement	Exp4		Exp5		Exp6	
	$\top$	$\perp$	$\top$	$\perp$	$\top$	$\perp$
OrdPatrol,UpperRestAvoid,Wait	2	10	1	11	1	11
FairVisit,ExactRestAvoid*,DelReact	5	7	0	12	4	8
StrOrdVisit,GlobalAvoid,InstReact	3	9	1	11	1	11
SeqVisit,FutAvoid,BindDel*	1	11	0	12	2	10
OrdVisit,PastAvoid,InstReact	3	9	1	11	1	11
Visit,LowRestAvoid,BindReact	3	9	1	11	1	11
StrictOrdPatrol,FutAvoid,Wait	1	11	1	11	1	11
Patrol,LowRestAvoid,InstReact	3	9	1	11	1	11
FairPatrol,ExactRestAvoid*,DelReact	3	9	0	12	4	8
SeqPatrol,UpperRestAvoid,FastReact*	1	11	0	12	2	10

**Summary.** The pattern catalog is effective in supporting developers in defining mission requirements and in generating mission specifications. Exp1 and Exp2 show that the pattern catalog effectively supports the definition of mission requirements, and that helps in reducing ambiguities in available mission requirements. Exp1 and Exp3 show that the pattern catalog effectively supports the generation of mission specifications. Exp1 shows how the pattern catalog can be used to generate precise, unambiguous, and formal mission specifications in industry sourced scenarios.

**Correctness of the Patterns (RQ2).** To verify the mission specifications (LTL and CTL formulas) we manually reviewed them and performed a random testing to confirm that the specifications do not permit undesired system behaviors that were not detected during the manual check.

*Manual check.* We manually inspected instances of the patterns obtained by fixing the number of locations to be visited, conditions to be considered etc. For LTL formulae we used SPOT [78] to generate Büchi automata (BA) encoding the traces of the system allowed and forbidden by the specification. We manually inspected the BA of all the proposed patterns.

*Random testing.* We performed some testing by exploiting a set of randomly generated models: a widespread technique to evaluate artifacts in the software engineering community [79]–[85], also used in the robotic community [20], [86]–[89]. We generated 12 scenarios representing the structure of buildings containing 16 locations, where a robot is deployed. The building has been generated by allocating 12 traversable locations and 4 locations that cannot be crossed, on a  $4 \times 4$  matrix. Identifiers  $l_0, l_1, \dots, l_{11}$  are randomly assigned to the traversable locations. In 6 of the 12 scenarios the robot can move among adjacent cells that are traversable, while it cannot move within not crossable locations. In the other 6 scenarios the robot can cross the adjacent cells by respecting the following rules: (i) it can move from a traversable cell with coordinate  $[i, j]$  to a traversable cell with coordinate  $[i, j+1]$  and  $[i+1, j]$ ; (ii) it can move from a traversable cell with coordinate  $[i, j]$  to another with coordinate  $[h, k]$ , where  $i$  (resp.  $h$ ) is the maximum (resp. minimum) row index of a cell that corresponds to a traversable location and  $h$  (resp.  $k$ ) is the maximum (resp. minimum)

column index of the traversable locations at row  $i$  (resp.  $h$ ). Conditions and actions are treated by considering whether a box is present in a location (*cond* in the following), and the capability of the robot in changing its color (*act* in the following). We randomly select 4 traversable locations in which *cond* is true and 4 locations in which *act* can be performed.

For each scenario we considered different mission requirements; each obtained by randomly combining a core movement, a trigger and an avoidance pattern, and by ensuring that each pattern is used in at least one mission requirement. In total we generated 10 mission requirements (Table VIII). Core movement patterns are parametrized with locations  $l_1, l_2$ . The upper, exact and lower restricted avoidance patterns are parametrized by forcing the robot to visit location  $l_3$ , at most, exactly, and at least 2 times, respectively. The global avoidance pattern forces the robot to not visit  $l_3$ , while the future and past avoidance force the robot to not visit  $l_3$  after and before condition *cond* is satisfied, i.e., a room that contains a box is visited. The wait pattern forces the robot to wait in location  $l_4$  if a box is not present. The other trigger patterns are parametrized with the action *act* that must be executed by the robot in relation with the occurrence of condition *cond*. We subsequently performed the following experiments.

*Exp4.* We generated the LTL specifications of the considered mission requirements. We (i) negated the LTL specification; (ii) encoded the specification and the model of the scenario in NuSMV [64]; (iii) used NuSMV to check whether the models contained a path that satisfied the mission specification (violates its negation). If a plan was present we used Simbad [65] to simulate the robot executing the plan. We verified whether the results were correct: when we expected a plan to not be present in the given model, NuSMV was not able to compute it, and, when a plan was expected to be present it was computed by NuSMV. We also checked the correctness of the generated plans using the Simbad simulator. Results confirm the correctness of the LTL mission specifications. The column labeled with the  $\top$  (resp.  $\perp$ ) symbol of Table VIII contains the number of cases in which a plan was (resp. was not) present.

*Exp5.* We generated LTL and CTL specifications for the considered mission requirements. We (i) encoded the LTL and CTL specifications and the model of the scenario in NuSMV [64]; (iii) we used NuSMV to check whether the verification of the specifications returned the same results. Table VIII contains the number of cases in which the mission requirement was satisfied ( $\top$ ) and not satisfied ( $\perp$ ). Mission requirements were generally not satisfied, since for being satisfied they have to hold on all the paths of the models. NuSMV always returned the same results for LTL and CTL specifications confirming the correctness of CTL specifications.

*Exp6.* We investigated why in several cases the mission requirement was always not satisfied. In these cases we relaxed the mission requirements, by removing the patterns marked with the \* symbol in Table VIII. We executed the same steps of *Exp4*. Table VIII confirmed that by relaxing the mission requirements there were cases in which the mission requirement was actually satisfied. This is a further confirmation that the

mission specifications are correct.

## VI. DISCUSSION AND RELATED WORK

We discuss the proposed patterns and present related work.

*Methodology.* The number of mission requirements analyzed is in line with other approaches in the field [49], [54]–[57]. These requirements usually come from exemplar scenarios used to provide evaluation about effectiveness of research-intensive works. As such, we believe that the scope of the pattern system is quite wide. Our study is certainly not exhaustive, as (i) formal specification in robotic application spreads, and (ii) the types of mission specifications change over time. As shown in the evaluation, patterns will grow over time as specifications that do not belong to the catalog are provided.

*Patterns.* While the presented patterns are mainly conceived to address needs of robotic mission specification, they are more generic and can be applied when the need is to specify some “ordering” among events or action execution. Rather than predicate on robots reaching a set of actions, coverage and surveillance patterns may also include propositions that refer to generic events. In this sense, the proposed patterns can be considered as an extension of the property specification patterns [49], [90] that explicitly address different ordering among the occurrence of a set of events. While in this paper we proposed a direct encoding in LTL and CTL, they may also be expressed in terms of standard property specification patterns. The instantaneous reaction pattern may be obtained from the response pattern scoped with the global operator. The precedence chain and the response chains [49], [90] (that illustrate the 2 cause-1 effect and 1 cause-2 effects chain), can be composed with the precedence and response patterns to specify different ordering among a set of events.

*Evaluation.* The Spectra tool only supports specifications captured by the GR(1) LTL fragment used to describe three types of guarantees: initial, safety, and liveness. Initial guarantees constraint the initial states of the environment. Safety guarantees start with the temporal operator  $\mathcal{G}$  and constraints the current and next state. Liveness guarantees start with the temporal operators  $\mathcal{G}\mathcal{F}$  and may not include the  $\mathcal{X}$  operator. These constraints justify the prevalence of patterns presented in Tables VI, VII, and VIII. While the proposed patterns can be expressed using deterministic Büchi automata (DBA), which can be translated in GR(1) formulae [28], a manual encoding of the proposed patterns in GR(1) is complex and error prone. This is confirmed by the fact that analysis on the standard property specification patterns that can be expressed in GR(1), and an automatic procedure to map these patterns on formulae that are in the GR(1) fragment has been recently conducted [28]. All of the patterns proposed in this work are expressible using GR(1) formulae, and the automatic procedure presented in [28] can be integrated in PsAIM to generate Spectra formulae.

**Related work.** Temporal logic specification patterns are a well-known solution to support developers in requirement specification [49], [54]–[56], [91]–[93]. Property specification patterns use in specific domains have been investigated in literature, including service-based applications [57], safety [94]

and security [95]. However, at the best of our knowledge, no work has considered mission patterns for robotic applications.

Domain Specific Languages (DSLs) [18], [33], [96]–[98] have been proposed for various purposes including production and analysis of behaviour descriptions, property verification and planning. However, features incorporated within DSLs are usually arbitrarily chosen by relying on the domain-specific experience of robotic engineers. Instead, specification patterns presented in this paper are collected from missions encountered in scientific literature, evaluated in industrial uses, and aim at supporting a wide range of robotic needs. We believe that the presented patterns consist of basic building blocks that can be reused within existing and new robotic DSLs. Moreover, support for developers on solving the mission specification problem is also provided in literature by graphical tools that simplify the specification of LTL formulae [35]–[37]. Our work is complementary with those; graphical logic mission specifications can also be integrated within PsAIM.

## VII. CONCLUSION

We proposed a pattern catalog for mission specification of mobile robots. We identified patterns by analyzing mission requirements that have been systematically collected from scientific publications. We presented PsAIM, a tool that uses the proposed patterns to support developers in designing complex missions. We evaluated (i) the support provided by the catalog in the definition of real-world missions; (ii) the correctness of the mission specifications contained in our pattern catalog.

Future extensions of our mission specification pattern catalog will consider also time, space, and probability. We will also investigate the use of spatial logics [99]–[103] to express more complex spatial robotic behaviours and perform user studies.

## REFERENCES

- [1] IFR, “World Robotic Survey,” <https://ifr.org/ifr-press-releases/news/world-robotics-survey-service-robots-are-conquering-the-world->, 2016.
- [2] D. Brugali, *Software engineering for experimental robotics*. Springer, 2007, vol. 30.
- [3] E. A. Lee, “Cyber physical systems: Design challenges,” in *Symposium on Object Oriented Real-Time Distributed Computing (ISORC)*. IEEE, 2008, pp. 363–369.
- [4] J. Pérez, N. Ali, J. A. Carsí, I. Ramos, B. Álvarez, P. Sanchez, and J. A. Pastor, “Integrating aspects in software architectures: Prisma applied to robotic tele-operated systems,” *Information and Software Technology*, vol. 50, no. 9-10, pp. 969–990, 2008.
- [5] N. Gamez and L. Fuentes, “Architectural evolution of famiware using cardinality-based feature models,” *Information and Software Technology*, vol. 55, no. 3, pp. 563–580, 2013.
- [6] D. Brugali and E. Prassler, “Software engineering for robotics,” *IEEE Robotics Automation Magazine*, vol. 16, no. 1, pp. 9–15, March 2009.
- [7] S. Götz, C. Piechnick, and A. Wortmann, “Report on the 4th international workshop on model-driven robot software engineering (MORSE),” *ACM SIGSOFT Software Engineering Notes*, 2018.
- [8] M. Luckcuck, M. Farrell, L. Dennis, C. Dixon, and M. Fisher, “Formal specification and verification of autonomous robotic systems: A survey,” *arXiv preprint arXiv:1807.00048*, 2018.
- [9] S. Maoz and J. O. Ringert, “On the software engineering challenges of applying reactive synthesis to robotics,” in *Workshop on Robotics Software Engineering*, ser. RoSE ’18. ACM, 2018.
- [10] S. Maoz and J. O. Ringert, “On well-separation of GR(1) specifications,” in *Foundations of Software Engineering (FSE)*. ACM, 2016.
- [11] C. Lignos, V. Raman, C. Finucane, M. Marcus, and H. Kress-Gazit, “Provably correct reactive control from natural language,” *Autonomous Robots*, vol. 38, no. 1, pp. 89–105, 2015.
- [12] <http://www.lego.com/eng/education/mindstorms/default.asp>, “Lego.com educational division, mindstorms for schools.” 2018.
- [13] Softbankrobotics. Choregraph. <http://doc.aldebaran.com/1-14/dev/tools/robot-simulation.html>. Accessed: 2018-06-20.
- [14] A. Nordmann, N. Hochgeschwender, and S. Wrede, “A survey on domain-specific languages in robotics,” in *Simulation, Modeling, and Programming for Autonomous Robots*. Springer, 2014.
- [15] R. Arkin, “Missionlab v7. 0,” 2006.
- [16] T. Balch, “Teambots,” 2004. [Online]. Available: [www.teambots.org](http://www.teambots.org)
- [17] S. Maoz and Y. Sa’ar, “Aspectltl: an aspect language for ltl specifications,” in *International conference on Aspect-oriented software development*. ACM, 2011.
- [18] D. D. Ruscio, I. Malavolta, P. Pelliccione, and M. Tivoli, “Automatic generation of detailed flight plans from high-level mission descriptions,” in *Model Driven Engineering Languages and Systems*, ser. MODELS. ACM, 2016.
- [19] M. Broy, “Declarative specification and declarative programming,” in *Software Specification and Design*. IEEE, 1991.
- [20] C. Menghi, S. Garcia, P. Pelliccione, and J. Tumova, “Multi-robot LTL planning under uncertainty,” in *International Symposium on Formal Methods (FM)*. Springer, 2018.
- [21] A. Ulusoy, S. L. Smith, X. C. Ding, C. Belta, and D. Rus, “Optimal multi-robot path planning with temporal logic constraints,” in *International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2011.
- [22] G. E. Fainekos, A. Girard, H. Kress-Gazit, and G. J. Pappas, “Temporal logic motion planning for dynamic robots,” *Automatica*, vol. 45, no. 2, pp. 343–352, 2009.
- [23] M. Guo, K. H. Johansson, and D. V. Dimarogonas, “Revising motion planning under linear temporal logic specifications in partially known workspaces,” in *International Conference on Robotics and Automation (ICRA)*. IEEE, 2013.
- [24] E. M. Wolff, U. Topcu, and R. M. Murray, “Automaton-guided controller synthesis for nonlinear systems with temporal logic,” in *International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2013.
- [25] H. Kress-Gazit, “Robot challenges: Toward development of verification and synthesis techniques [errata],” *IEEE Robotics & Automation Magazine*, vol. 18, no. 4, pp. 108–109, 2011.
- [26] M. Guo and D. V. Dimarogonas, “Multi-agent plan reconfiguration under local LTL specifications,” *The International Journal of Robotics Research*, 2015.
- [27] S. Maoz and J. O. Ringert, “Synthesizing a lego forklift controller in GR(1): A case study,” in *Proceedings Fourth Workshop on Synthesis (SYNT)*, 2015.
- [28] S. Maoz and J. O. Ringert, “GR(1) synthesis for LTL specification patterns,” in *Foundations of Software Engineering (FSE)*. ACM, 2015.
- [29] F. S. Rodriguez, B. C. Diego, V. M. Rodilla, J. Rodriguez-Aragon, R. A. Santos, and C. Fernandez-Carames, “The complete integration of missionlab and carmen,” *International Journal of Advanced Robotic Systems*, vol. 14, no. 3, p. 1729881417703565, 2017.
- [30] M. G. Hinchey, J. L. Rash, and C. A. Rouff, “Requirements to design to code: Towards a fully formal approach to automatic code generation,” 2005.
- [31] J. F. Kramer and M. Scheutz, “Development environments for autonomous mobile robots: A survey,” *Autonomous Robots*, vol. 22, pp. 101–132, 2007.
- [32] S. Maniatopoulos, M. Blair, C. Finucane, and H. Kress-Gazit, “Open-world mission specification for reactive robots,” in *International Conference on Robotics and Automation (ICRA)*. IEEE, 2014.
- [33] D. Bozhinoski, D. D. Ruscio, I. Malavolta, P. Pelliccione, and M. Tivoli, “Flyaq: Enabling non-expert users to specify and generate missions of autonomous multicopters,” in *Automated Software Engineering (ASE)*. IEEE, 2015.
- [34] X. C. Ding, M. Kloetzer, Y. Chen, and C. Belta, “Automatic deployment of robotic teams,” *IEEE Robotics & Automation Magazine*, vol. 18, no. 3, pp. 75–86, 2011.
- [35] I. Lee and O. Sokolsky, “A graphical property specification language,” in *High-Assurance Systems Engineering Workshop*. IEEE, 1997.
- [36] M. H. Smith, G. J. Holzmann, and K. Etesami, “Events and constraints: A graphical editor for capturing logic requirements of programs,” in *International Symposium on Requirements Engineering*. IEEE, 2001.
- [37] S. Srinivas, R. Kermani, K. Kim, Y. Kobayashi, and G. Fainekos, “A graphical language for ltl motion and mission planning,” in *International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, 2013.

- [38] V. Raman, C. Lignos, C. Finucane, K. C. Lee, M. Marcus, and H. Kress-Gazit, "Sorry dave, i'm afraid i can't do that: Explaining unachievable robot tasks using natural language," University of Pennsylvania Philadelphia United States, Tech. Rep., 2013.
- [39] C. Finucane, G. Jing, and H. Kress-Gazit, "LTLMoP: Experimenting with language, temporal logic and robot control," in *International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2010, pp. 1988–1993.
- [40] X. C. Ding, M. Kloetzer, Y. Chen, and C. Belta, "Automatic deployment of robotic teams," *IEEE Robotics Automation Magazine*, vol. 18, no. 3, pp. 75–86, Sept 2011.
- [41] Y. Endo, D. C. MacKenzie, and R. C. Arkin, "Usability evaluation of high-level user assistance for robot mission specification," *Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 34, no. 2, pp. 168–180, 2004.
- [42] W. Wei, K. Kim, and G. Fainekos, "Extended LTLvis motion planning interface," in *International Conference on Systems, Man, and Cybernetics*. IEEE, 2016.
- [43] C. Lignos, V. Raman, C. Finucane, M. Marcus, and H. Kress-Gazit, "Provably correct reactive control from natural language," *Autonomous Robots*, vol. 38, no. 1, pp. 89–105, Jan 2015.
- [44] U. S. Shah and D. C. Jinwala, "Resolving ambiguities in natural language software requirements: a comprehensive survey," *ACM SIGSOFT Software Engineering Notes*, 2015.
- [45] N. Kiyavitskaya, N. Zeni, L. Mich, and D. M. Berry, "Requirements for tools for ambiguity identification and measurement in natural language requirements specifications," *Requirements engineering*, 2008.
- [46] J. O. Ringert, B. Rumpe, and A. Wortmann, "A requirements modeling language for the component behavior of cyber physical robotics systems," *arXiv preprint arXiv:1409.0394*, 2014.
- [47] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas, "Temporal-logic-based reactive mission and motion planning," *Transactions on robotics*, vol. 25, no. 6, pp. 1370–1381, 2009.
- [48] C. Yoo, R. Fitch, and S. Sukkariéh, "Online task planning and control for fuel-constrained aerial robots in wind fields," *The International Journal of Robotics Research*, vol. 35, no. 5, pp. 438–453, 2016.
- [49] M. B. Dwyer, G. S. Avrunin, and J. C. Corbett, "Patterns in property specifications for finite-state verification," in *International Conference on Software Engineering (ICSE)*. IEEE, 1999.
- [50] E. A. EMERSON, "{CHAPTER} 16 - temporal and modal logic," in *Formal Models and Semantics*, ser. Handbook of Theoretical Computer Science, J. V. LEEUWEN, Ed. Elsevier, 1990, pp. 995 – 1072.
- [51] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas, "Temporal-logic-based reactive mission and motion planning," *Transactions on Robotics*, vol. 25, no. 6, pp. 1370–1381, 2009.
- [52] G. J. Holzmann, "The logic of bugs," in *Foundations of Software Engineering (FSE)*. ACM, 2002.
- [53] M. Autili, P. Inverardi, and P. Pelliccione, "Graphical scenarios for specifying temporal properties: An automated approach," *Automated Software Engg.*, vol. 14, no. 3, 2007.
- [54] L. Grunske, "Specification patterns for probabilistic quality properties," in *International Conference on Software Engineering (ICSE)*. IEEE, 2008.
- [55] S. Konrad and B. H. Cheng, "Real-time specification patterns," in *International conference on Software engineering (ICSE)*. IEEE, 2005.
- [56] 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," *Transactions on Software Engineering*, vol. 41, no. 7, pp. 620–638, 2015.
- [57] D. Bianculli, C. Ghezzi, C. Pautasso, and P. Senti, "Specification patterns from research to industry: a case study in service-based applications," in *International Conference on Software Engineering (ICSE)*. IEEE, 2012.
- [58] C. Menghi, C. Tsigkanos, T. Berger, P. Pelliccione, and C. Ghezzi, "Property specification patterns for robotic missions," in *International Conference on Software Engineering (ICSE): Companion Proceedings*, 2018.
- [59] R. A. Brooks *et al.*, "Intelligence without reason," *Artificial intelligence: critical concepts*, vol. 3, pp. 107–63, 1991.
- [60] D. Brugali and M. Reggiani, "Software stability in the robotics domain: issues and challenges," in *International Conference on Information Reuse and Integration*. IEEE, 2005.
- [61] D. Brugali, "Stable analysis patterns for robot mobility," in *Software Engineering for Experimental Robotics*. Springer, 2007, pp. 9–30.
- [62] C. Menghi, C. Tsigkanos, T. Berger, and P. Pelliccione, "PsAIM: Specification of dependable robotic missions," in *International Conference on Software Engineering (ICSE): Companion Proceedings*, 2019.
- [63] S. Maoz and J. O. Ringert. Spectra. <http://smlab.cs.tau.ac.il/syntech/spectra/>. Accessed: 2018-06-20.
- [64] A. Cimatti, E. Clarke, F. Giunchiglia, and M. Roveri, "NuSMV: A new symbolic model verifier," in *Computer Aided Verification (CAV)*. Springer, 1999.
- [65] L. Hugues and N. Bredeche, "Simbad: an autonomous robot simulation package for education and research," in *International Conference on Simulation of Adaptive Behavior*. Springer, 2006.
- [66] "Accompanied material and data for this paper," <http://claudiomenghi.github.io/RobotPatterns/>, 2018.
- [67] I. Ruchkin, J. Sunshine, G. Iraci, B. Schmerl, and D. Garlan, "IPL: An integration property language for multi-model cyber-physical systems," in *International Symposium on Formal Methods*. Springer, 2018.
- [68] P. Ulam, Y. Endo, A. Wagner, and R. Arkin, "Integrated mission specification and task allocation for robot teams - design and implementation," in *International Conference on Robotics and Automation*. IEEE, 2007.
- [69] A. Pnueli, "The temporal logic of programs," in *Foundations of Computer Science*. IEEE, 1977.
- [70] M. Ben-Ari, A. Pnueli, and Z. Manna, "The temporal logic of branching time," *Acta informatica*, 1983.
- [71] "Google Scholar Robotic Venues," [https://scholar.google.com/citations?view\\_op=top\\_venues&hl=en&vq=eng\\_robotics](https://scholar.google.com/citations?view_op=top_venues&hl=en&vq=eng_robotics), 2017.
- [72] P. Schillinger, S. Kohlbrecher, and O. von Stryk, "Human-robot collaborative high-level control with application to rescue robotics," in *International Conference on Robotics and Automation (ICRA)*. IEEE, 2016.
- [73] S. L. Smith, J. Tumová, C. Belta, and D. Rus, "Optimal path planning for surveillance with temporal-logic constraints," *The International Journal of Robotics Research*, vol. 30, no. 14, pp. 1695–1708, 2011.
- [74] Robotnik. <https://www.robotnik.eu/mobile-robots/summit-xl-hl/>. Accessed: 2018-06-20.
- [75] Mitsubishi. <https://robotik.dfki-bremen.de/en/research/robot-systems/mitsubishi-pa-10-7c.html>. Accessed: 2018-06-20.
- [76] P. robotics. <http://tiago.pal-robotics.com/>. Accessed: 2018-06-20.
- [77] Syntech. <http://smlab.cs.tau.ac.il/syntech/lego/>. Accessed: 2018-06-20.
- [78] A. Duret-Lutz, A. Lewkowicz, A. Fauchille, T. Michaud, E. Renault, and L. Xu, "Spot 2.0 — a framework for LTL and  $\omega$ -automata manipulation," in *Automated Technology for Verification and Analysis*. Springer, 2016.
- [79] D. Tabakov and M. Y. Vardi, "Experimental Evaluation of Classical Automata Constructions," in *International Conference on Logic for Programming Artificial Intelligence and Reasoning*. Springer, 2005.
- [80] M. De Wulf, L. Doyen, T. A. Henzinger, and J.-F. Raskin, "Antichains: A New Algorithm for Checking Universality of Finite Automata," in *International Conference on Computer Aided Verification*. Springer, 2006.
- [81] D. Tabakov and M. Y. Vardi, "Model Checking Buchi Specifications," in *International Conference on Language and Automata Theory and Applications*, 2007.
- [82] P. Saadatpanah, M. Famelis, J. Gorzny, N. Robinson, M. Chechik, and R. Salay, "Comparing the Effectiveness of Reasoning Formalisms for Partial Models," in *Workshop on Model-Driven Engineering, Verification and Validation*. ACM, 2012.
- [83] M. Famelis, R. Salay, and M. Chechik, "Partial Models: Towards Modeling and Reasoning with Uncertainty," in *International Conference on Software Engineering (ICSE)*. IEEE, 2012.
- [84] C. Menghi, P. Spoletini, and C. Ghezzi, "Dealing with incompleteness in automata-based model checking," in *Formal Methods (FM)*. Springer, 2016.
- [85] C. Menghi, P. Spoletini, M. Chechik, and C. Ghezzi, "Supporting verification-driven incremental distributed design of components," in *Fundamental Approaches to Software Engineering*. Springer, 2018.
- [86] C. Menghi, S. García, P. Pelliccione, and J. Tumova, "Towards multi-robot applications planning under uncertainty," in *International Conference on Software Engineering (ICSE): Companion Proceeding*, 2018.
- [87] G. Best, J. Faigl, and R. Fitch, "Multi-robot path planning for budgeted active perception with self-organising maps," in *International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016.
- [88] B. Takács and Y. Demiris, "Multi-robot plan adaptation by constrained minimal distortion feature mapping," in *International Conference on Robotics and Automation (ICRA)*. IEEE, 2009.

- [89] A. Stentz, "Map-based strategies for robot navigation in unknown environments," in *AAAI spring symposium on planning with incomplete information for robot problems*, 1996, pp. 110–116.
- [90] "Order Specification Patterns," <http://patterns.projects.cs.ksu.edu/documentation/patterns/order.shtml>.
- [91] D. O. Paun and M. Chechik, "Events in linear-time properties," in *International Symposium on Requirements Engineering*. IEEE, 1999.
- [92] D. Remenska, T. A. C. Willemse, J. Templon, K. Verstoep, and H. Bal, "Property specification made easy: Harnessing the power of model checking in uml designs," in *International Federated Conference on Distributed Computing Techniques*. Springer, 2014.
- [93] K. C. Castillos, F. Dadeau, J. Julliland, B. Kanso, and S. Taha, *A Compositional Automata-Based Semantics for Property Patterns*. Springer, 2013.
- [94] F. Bitsch, "Safety patterns - the key to formal specification of safety requirements," in *International Conference on Computer Safety, Reliability and Security*, 2001.
- [95] G. Spanoudakis, C. Kloukinas, and K. Androutsopoulos, "Towards security monitoring patterns," in *Symposium on Applied Computing*. ACM, 2007.
- [96] D. C. Schmidt, "Guest editor's introduction: Model-driven engineering," *Computer*, vol. 39, no. 2, pp. 25–31, 2006.
- [97] F. Ciccozzi, D. D. Ruscio, I. Malavolta, and P. Pelliccione, "Adopting MDE for specifying and executing civilian missions of mobile multi-robot systems," *Journal of IEEE Access*, vol. 2, no. 1, 2016.
- [98] S. Adam, M. Larsen, K. Jensen, and U. P. Schultz, *Towards Rule-Based Dynamic Safety Monitoring for Mobile Robots*. Springer, 2014, pp. 207–218.
- [99] M. Aiello, I. Pratt-Hartmann, J. van Benthem *et al.*, *Handbook of spatial logics*. Springer, 2007, vol. 4.
- [100] C. H. Papadimitriou, D. Suciu, and V. Vianu, "Topological queries in spatial databases," in *Symposium on Principles of database systems*. ACM, 1996.
- [101] R. S. Bivand, E. Pebesma, and V. Gómez-Rubio, *Spatial Data Import and Export*. Springer, 2013.
- [102] R. Kontchakov, A. Kurucz, F. Wolter, and M. Zakharyashev, "Spatial logic+ temporal logic=" in *Handbook of spatial logics*. Springer, 2007, pp. 497–564.
- [103] L. Cardelli, P. Gardner, and G. Ghelli, "A spatial logic for querying graphs," in *Automata, Languages and Programming*. Springer, 2002.