1 Bardh Hoxha and Georgios Fainekos are with the School of Computing, Informatics and Decision Systems Engineering, Arizona State University  
{bhoxa,fainekos}@asu.edu  
2 Nikolaos Mavridis is with the Interactive Robots and Media Lab and NCSR Demokritos nmav@alum.mit.edu

Abstract—One of the main barriers preventing widespread use of formal methods is the elicitation of formal specifications. Formal specifications facilitate the testing and verification process for safety critical robotic systems. However, handling the intricacies of formal languages is difficult and requires a high level of expertise in formal logics that many system developers do not have. In this work, we present a graphical tool designed for the development and visualization of formal specifications by people that do not have training in formal logic. The tool enables users to develop specifications using a graphical formalism which is then automatically translated to Metric Temporal Logic (MTL). In order to evaluate the effectiveness of our tool, we have also designed and conducted a usability study with cohorts from the academic student community and industry. Our results indicate that both groups were able to define formal requirements with high levels of accuracy. Finally, we present applications of our tool for defining specifications for operation of robotic surgery and autonomous quadcopter safe operation.

I. INTRODUCTION

As robots become commercially available, their correct operation is of paramount importance. Especially for safety critical systems, safety must be guaranteed. As for example in autonomous vehicles [24] and medical robots [17], [13]

Safety requirements are usually expressed in natural language, which is inherently ambiguous, in general. When it is used for defining system specifications, this ambiguity may lead to misunderstandings between development teams that may result in increased costs and delays in development. If the misunderstandings are not detected, then a product that does not meet the intended specifications will be developed.

Ideally, specifications should be defined in a mathematical language, using formal logics. This not only removes ambiguity, but also allows system developers to utilize a vast set of methods [22] that have been developed by the academic community for testing and verification of systems. The academic community has also developed automatic tools such as S-TALiRo [2], [11], FAPAS [25], SpaceEx [9], CheckMate [19], FLOW [4], Breach [6], C2E2 [7], KeYmaera [18] and STRONG [5] that enable developers to conduct system testing and verification.

Even though it has been shown, that utilizing formal specifications can lead to improved testing and verification [8], the industry still utilizes natural language as the premier approach in defining specifications. One may conjecture that the most important reason for doing so is because the development of specifications through a formal logic requires a level of mathematical training that many users may not have [23]. Furthermore, even for expert users, writing formal specifications is an error prone task [10]. As a result, the industry has been less willing to utilize formal specifications in their processes.

In this work, we present a graphical formalism that enables non-expert users to develop formal specifications for control systems. The formalism enables the visualization of a large fragment of MTL. The main challenge in the development of the formalism lies in finding the right balance between expressive power and ease-of-use. It is designed for use with systems and signals and enables both event and time based specifications. This is the first time that a visual formal language representation is developed for specifications for Cyber-Physical Systems (CPS). Here by CPS we define any system that has discontinuous nonlinear dynamics and complex safety critical requirements. Prime examples are medical robotics and autonomous vehicles. A specification visualization tool has been developed based on the graphical formalism presented in this work. To evaluate the usefulness of the tool in terms of usability and ease-of-use, we have conducted a usability study.

SUMMARY OF CONTRIBUTIONS:

• We present a graphical formalism that enables the development of formal specifications.
• We present the visual specification tool based on the graphical formalism.
• We conducted a usability study to evaluate the tool.
• Through the usability study we proved that both non-expert users and expert users are able to define formal requirements accurately using the tool, and derived suggestions for improvement of the tool.
• We present applications of the tool for real-world robots.

RELATED WORKS: In order to help address the formal specification challenge, various graphical formalisms have been studied in the past [20], [1], [15], [3], [26], [21]. The most relevant works appear in [3] and [26]. In [3], the authors extend Message Sequence Charts and UML 2.0 Interaction Sequence Diagrams to propose a scenario based formalism called Property Sequence Chart (PSC). The formalism is mainly developed for specifications on concurrent systems. In [26], PSC is extended to Timed PSC which enables the addition of timing constructs to specifications.

In terms of usability studies for formal requirements very few works exist. In [23], the authors study the ability of expert users to develop requirements in Z. A related usability study for requirement representation is presented in [16], where the authors present and evaluate a system.
for generating, troubleshooting and executing controllers for robots using natural language.

II. VISUAL SPECIFICATION TOOL

The Visual Specification Tool (ViSPEC)\(^1\) enables the development of formal specifications for CPS. Users can develop requirements in a graphical formalism which is then translated to Metric Temporal Logic (MTL) [14].

The topic of capturing requirements through graphical formalisms has been studied in the past [20], [1], [15], [3], [26]. However, to the best of the authors knowledge, the work presented here is the first attempt to do so specifically aimed for the development of specifications for CPS. The initial idea for the graphical formalism was first presented in [11] while the tool was still in the early stages of development. However, in this work we present an updated version of the tool along with its usability study. The improvements over the previous version include: a more streamlined interface; an updated representation of signals in the interface; and an updated template definition process.

For CPS specifications, it is often needed to account for both timing and event sequence occurrence. Both of these are necessary for reasoning over systems and signals. Consider the specification \(\Box_{[0,5]}((speed > 100) \rightarrow \Box_{[0,5]}(rpm > 4000))\). It states that whenever within the first 5 seconds, the vehicle speed goes over 100, then from that moment on, the engine speed (rpm), for the next 5 seconds, should always be over 4000. Here both the sequence and timing of the events are of critical importance.

To ensure that the tool can be utilized by non-expert users, the following goals for the tool are defined: 1) The user interface is intuitive to use, i.e, it does no have a high learning curve; 2) The visual representation of the requirements is visually distinct and unambiguous; 3) There is a one-to-one mapping from the visual representation of the requirement and the corresponding requirement in MTL.

The set of specifications that can be generated from this graphical formalism is a proper subset of the set of MTL specifications. Formally, the following grammar produces the set of formulas that can be expressed by the proposed graphical formalism:

\[
\begin{align*}
S &\rightarrow \neg T \mid T \\
T &\rightarrow A \mid B \mid C \\
A &\rightarrow P \mid (P \land A) \mid (P \Rightarrow A) \\
B &\rightarrow \Box T \mid \Diamond T \\
C &\rightarrow \Box T \Diamond T \mid \Diamond T \Box T \\
D &\rightarrow p \mid (p \Rightarrow A) \mid (p \land A) \mid (p \Rightarrow B) \mid (p \land B) \\
P &\rightarrow p \mid \Box T p \mid \Diamond T p
\end{align*}
\]

where \(p\) is an atomic proposition. In practice, the atomic propositions are automatically derived from the templates.

Throughout the development process of the formalism, it was noticed that the more expressive the formalism, the more challenging to use it became. Therefore, we focused on several widely used classes of specifications which are described in Table I. Examples of the classes of specifications are presented in the rest of this section.

To make the tool easier to use, we placed several constraints on the types of signals used. Specifically, the signals and requirements are one dimensional. This enables clear and structured visualization on a two dimensional user interface.

The main building blocks of the formalism are templates. These are used for defining temporal logic operators, their timing intervals, and the expected signal shape. The user starts with an empty template and a setup assistant presents the user with a sequence of dialog boxes that aid in the development of the template. The process is context dependent where each option selection leads to a potentially different set of options for the next step.

The first step in the template definition process is to define the temporal operator. Among the choices (and their corresponding MTL symbols) are: Always (\(\square\)), At Least Once (\(\Diamond\)), Eventually Always (\(\Diamond \square\)), Repeatedly Often and Finally (\(\square \Diamond\)), and now. The options available enable users to define a wide range of specifications. The following sections will present examples of a subset of formulas that can be generated using this graphical formalism.

After the temporal operator is selected, the user sets the timing bounds for it. Many users might have difficulty defining timing bounds, especially for specifications with temporal operators such as Eventually Always (\(\Diamond \square\)) and Repeatedly Often and Finally (\(\square \Diamond\)). To illustrate the process, the tool provides a fill-in-the-blanks sentence format to the user. For example, if the operator Eventually Always is selected, the user will have to complete the following sentence with the timing bounds: “Eventually, between ___ and ___ seconds, the signal will become true, and from that point on, will stay true in the next ___ to ___ seconds”. The set of timing intervals are visualized with color shaded regions in the template.

The next step in the process is in defining whether the predicate will evaluate to true when the signal is above or below a set threshold. For example, for the Always (\(\square\)) operator, a signal is selected that is either always above or below a specified threshold. Once either option is selected, various signals that fit the requirement are automatically generated and presented visually. Instead of drawing the signal, the user will select from one of the generated options. Consider the following example:

**Example 1** A specification from the fragment of MTL formulas called Safety MTL specifications is presented. Specifically, the specification \(\phi_1 = \Box_{[0,36]}(rpm < 4000)\). The formula states that in the next 36 seconds, engine speed should always be less than 4000. The corresponding graphical formalism for this formula is presented in Fig. 1. Note that, in regards to the specification, the signal can be of any shape as long as it is always below the 4000 threshold.

Consider the following example for the At Least Once (\(\Diamond\)) temporal operator:

**Example 2** A specification from the fragment of MTL for-


<table>
<thead>
<tr>
<th>Specification Class</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Safety</td>
<td>Specifications of the form $\square \phi$ used to define specifications where $\phi$ should always be true.</td>
</tr>
<tr>
<td>Reachability</td>
<td>Specifications of the form $\diamond \phi$ used to define specifications where $\phi$ should be true at least once in the future (on now).</td>
</tr>
<tr>
<td>Stabilization</td>
<td>Specifications of the form $\square \diamond \phi$ used to define specifications that, at least once, $\phi$ should be true and from that point on, stay true.</td>
</tr>
<tr>
<td>Recurrence</td>
<td>Specifications of the form $\square \diamond \phi$ used to define specifications that, it is always the case, that at some point in the future, $\phi$ is true.</td>
</tr>
<tr>
<td>Implication</td>
<td>Specifications of the form $\phi \rightarrow \psi$ requires the $\psi$ should hold when $\phi$ is true.</td>
</tr>
<tr>
<td>Reactive Response</td>
<td>Specifications of the form $N(\phi \rightarrow M \psi)$, where $N$ and $M$ are temporal operators, used to define an implicative response between two specifications where the timing of $M$ is relative to timing of $N$.</td>
</tr>
<tr>
<td>Conjunction</td>
<td>Specifications of the form $\phi \land \psi$ used to define the conjunction of two sub-specifications.</td>
</tr>
<tr>
<td>Non-strict Sequencing</td>
<td>Specifications of the form $N(\phi \land M \psi)$, where $N$ and $M$ are temporal operators, used to define a conjunction between two specifications where the timing of $M$ is relative to timing of $N$.</td>
</tr>
</tbody>
</table>

For more examples of specifications and their corresponding graphical formalism see the technical report [12]. In Section VI, we present two application specifications that illustrate the various interactions between templates.

The variety of templates and the connections between them allow users to express a wide variety of specifications.

### III. Graphical Formalism

The specification development process in ViSPEC is divided in two sub processes. First, given a user input in the ViSPEC tool, it is translated to a tree structure where the nodes contain template information such as temporal operators, their corresponding timing parameters, group and the value threshold for the predicates. Secondly, the generated tree structure is traversed by a recursive algorithm to generate the MTL specification.

An example of the tree structure for MTL formula $\phi = \square(a \land \diamond b) \rightarrow ((\square c \land \diamond (d \rightarrow (a \land \square b)))$ is shown in Fig. 4. The recursive algorithm for traversing the tree structure and generating the MTL formula is presented in Alg. 1. Note that the functions ADDPARENCONN{$A,B,C,D$} add the parenthesis and connectives between predicates. A more detailed presentation of the algorithm is presented in the technical report [12].

![Fig. 1: Example 1: The graphical formalism for the Safety MTL specification $\phi_1 = \square_{[0.36]}(rpm < 4000)$](image1)

![Fig. 2: Example 2: The graphical formalism for the Reachability MTL specification $\phi_2 = \diamond_{[0.39]}(speed > 100)$](image2)

![Fig. 4: The corresponding tree structure for formula $\phi = \square(a \land \diamond b) \rightarrow ((\square c \land \diamond (d \rightarrow (a \land \square b)))$](image4)

![Fig. 3: The specification development process using ViSPEC](image3)
Algorithm 1: WriteMTL - Algorithm for generating the MTL formula given a tree structure of the graphical formalism

**Input:** Tree Structure $T = (V, E)$ where $v \in V$ and $v = \langle G, Op, S \rangle$ where $G$ is the group, $Op$ is the temporal operator and $S$ is the predicate string: formula $\phi$.

**Output:** $\phi$

1. function WRITEMTL($T$, $\phi$)
2. $C \leftarrow T$.getChildren() if $i$.isParent then
3. $\phi \leftarrow CONC(\phi, i.Op)$
4. end if
5. $\phi \leftarrow ADDPARENCONNA(\phi, subC)$
6. $\phi \leftarrow WRITEMTL(i$.subtree, $\phi$)
7. $\phi \leftarrow ADDPARENCONNB(\phi, subC)$
8. end if
9. else
10. $\phi \leftarrow CONC(\phi, '(')$
11. $\phi \leftarrow WRITEMTL(i$.subtree, $\phi$)
12. $\phi \leftarrow ADDPARENCONNC(sC, $\phi$)
13. end if
14. end function

Some of the interesting questions we wanted to investigate, which are also presented as hypotheses in Tab. II, are:

- Whether the graphical formalism enables non-experts and experts to formalize requirements accurately.
- How well the expert cohort performs in comparison to the non-expert cohort.
- How user-friendly and easy-to-use ViSpec is.

Writing formal requirements is a challenging task that requires a significant amount of training. Therefore, it is safe to assume that we can reject Hypothesis 1a as supported by our informal experience. Hypothesis 2a will be tested in a future work. In addition, we analyze user interaction and behavior to measure the ease-of-use of the tool.

**B. Demographics**

The non-expert cohort was comprised of twenty subjects from the student community of Arizona State University. Most of the subjects are from an engineering background with little to no experience working with requirements. The student demographics are presented in Tab. III.

The expert subject cohort was comprised of ten subjects from the industry in the Phoenix area. The subjects have experience working with specifications and come from an engineering background.

**C. Experimental Design**

Each subject received a task list to complete. The task list contained ten tasks related to automotive system specifications. Each task asked the subject to formalize a natural language specification through ViSpec and generate an MTL formula. The list of tasks is presented in Table VI.

The tasks become more complex throughout the session. The higher the number of the task, the more steps necessary to complete the task successfully.

Each session is at most 45 minutes long. Subjects received a one minute and thirty second tutorial on using ViSpec to develop specifications. The computer screen was recorded and actions were logged for each session. The subjects also completed a demographic and post-completion questionnaire.

**D. Metrics**

Two metrics are used for performance evaluation:

- **Task completion:** this is a binary measure, which indicates whether users were able to finish the task within the set time.
- **Measure of Accuracy:** a value from one to five which is used to quantify the accuracy of subject generated formulas. The formulas are graded by formal specification experts which were given the following two suggested criteria: a) How accurate the meaning of the natural language specification is captured, and b) Whether the inaccuracies in the user submitted formula can be easily debugged and corrected in the testing and verification process. Furthermore, in order to decrease subjectivity, the following instructions were given to the expert graders in order to anchor the meanings of the five different grades of the scale used: A grade of one indicates that the generated formula is totally inaccurate. A grade of two indicates that the formula is mostly inaccurate. A grade of three indicates an inaccurate formula which can be easily debugged and corrected to the proper formal logic specification by formal specification experts and thus this is the minimum acceptable satisfactory result. A grade of four indicates that the formula is inaccurate but can be debugged and improved by automated specification debugging tools. A grade of five indicates that the generated formula is completely accurate. The group of expert graders consisted of experts in formal methods and logic.

**V. Results**

1) **Average grade per task:** For both cohorts, the task performance is presented in Fig. 8. It can be observed that overall, the mean grade per task for both cohorts is high.

2) **Hypothesis 1b:** To test Hypothesis 1b, we need to establish what is an acceptable threshold for accuracy in order to test the hypothesis. As discussed in the metrics section, we claim that a mean grade higher than three is an acceptable threshold for non-expert users. Therefore, hypothesis 1b is reduced to the null hypothesis: the mean grade per user is less than or equal to three for non-experts.

Let us define the average grade per user as a random variable $Y$. Specifically, $Y : \Omega \rightarrow \mathbb{R}$, where $\Omega \in \{y : 1 \leq y \leq 5\}$. The sample data from 20 subjects has a mean grade of 4.43 and standard deviation of 0.41. We test for normality with the Kolmogorov-Smirnov test, the Chi-square g.o.f test, and the Anderson-Darling test and all three fail to reject the null hypothesis that the data follows the normal distribution. If we assume that the data constitute a random sample from
TABLE II: Hypotheses and test results with level of significance $\alpha = 0.05$. User groups as defined in section IV.A.

<table>
<thead>
<tr>
<th>Hypothesis</th>
<th>Reject null hypothesis</th>
</tr>
</thead>
<tbody>
<tr>
<td>1a</td>
<td>Non-expert users are able to define formal requirements accurately using formal logics such as MTL.</td>
</tr>
<tr>
<td>1b</td>
<td>Non-expert users are able to define formal requirements accurately using the Visual Specification Tool.</td>
</tr>
<tr>
<td>2a</td>
<td>Expert users from the industry are able to define formal requirements accurately using formal logics such as MTL.</td>
</tr>
<tr>
<td>2b</td>
<td>Expert users from the industry are able to define formal requirements accurately using the Visual Specification Tool.</td>
</tr>
<tr>
<td>3alt</td>
<td>The mean grade per user for expert users is greater than the mean grade per user for non-expert users.</td>
</tr>
</tbody>
</table>

| $T_{x_{alt}}$ | The mean grade per task $x$ for industry users is greater than to the mean grade per task $x$ for non-expert users. |

TABLE III: Hypothesis 1b Subject Demographics

<table>
<thead>
<tr>
<th>Freshman</th>
<th>2</th>
<th>Computer Science</th>
<th>5</th>
<th>Male</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sophomore</td>
<td>2</td>
<td>Software Engineering</td>
<td>3</td>
<td>Female</td>
<td>8</td>
</tr>
<tr>
<td>Junior</td>
<td>5</td>
<td>Electrical Engineering</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Senior</td>
<td>5</td>
<td>Mechanical Engineering</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Masters</td>
<td>4</td>
<td>Engineering, other</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PhD</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

TABLE IV: ViSPEC improvements

<table>
<thead>
<tr>
<th>#</th>
<th>Improve...</th>
<th>Prime Indicators</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>the process of creating child templates</td>
<td>misclicks; user feedback</td>
</tr>
<tr>
<td>2</td>
<td>the tutorial by placing more emphasis on the difference between implication and conjunction between templates</td>
<td>task accuracy grade</td>
</tr>
<tr>
<td>3</td>
<td>the visual representation of grouped templates</td>
<td>task accuracy grade; user feedback</td>
</tr>
</tbody>
</table>

5) Hypothesis $T_x$: Next, we compare the mean grade of both cohorts in regards to each task. A two sample t-test is conducted for each task. The results for the tests are presented in Tab. V. Task 9 is the most difficult task when it comes to the number of errors generated, and this is the only task where there is a clear difference in performance between the expert and non-expert cohorts.

TABLE V: Hypothesis testing of $T_{x_{null}}$ with $\alpha = 0.05$

<table>
<thead>
<tr>
<th>$x$</th>
<th>Rej. $T_{x_{null}}$</th>
<th>p-val.</th>
<th>Conclusion</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>No</td>
<td>0.065</td>
<td>potentially true with more investigation</td>
</tr>
<tr>
<td>5</td>
<td>No</td>
<td>0.165</td>
<td>false</td>
</tr>
<tr>
<td>6</td>
<td>No</td>
<td>0.074</td>
<td>potentially true with more investigation</td>
</tr>
<tr>
<td>7</td>
<td>No</td>
<td>0.100</td>
<td>potentially true with more investigation</td>
</tr>
<tr>
<td>8</td>
<td>No</td>
<td>0.424</td>
<td>false</td>
</tr>
<tr>
<td>9</td>
<td>Yes</td>
<td>0.016</td>
<td>true</td>
</tr>
<tr>
<td>10</td>
<td>No</td>
<td>0.063</td>
<td>potentially true with more investigation</td>
</tr>
</tbody>
</table>

We observe that the only null hypothesis rejected is for task nine indicating that the mean grade for expert users is greater than the mean grade for non-expert users. The subject accuracy grades over tasks for is shown in Fig. 8.

6) Ease-of-use analysis: One indicator for the ease-of-use of the application is the total time spent per task. As can be observed in Fig. 5, the mean time spent per task on average is at most 167 seconds. For easier identification of points of difficulty, we divided each task into subtasks. It was observed that there is no correlation between the length of time spent in a subtask and correctness. This potentially indicates, as also verified by correlation testing between times and grades, that the subjects were unaware of mistakes in the process. From these and other observations, such as misclicks, and subject feedback, we have developed a set of refinements on the tool to improve the user experience. A partial list of improvements is presented in Table IV.

VI. APPLICATIONS

A. Robotic Surgery

In the last few decades, there has been a significant increase in the number of robotics systems, especially in the health care system. They have been successfully introduced in multiple areas such as rehabilitation, telesurgery, physical therapy, elderly care, and remote physician care. In the following, we will focus on autonomous robotic systems for surgery where of paramount importance is the safety of these systems [13]. Specifically, we will consider a model of a robotic serial link manipulator as presented in [17].

One of the main tasks in surgery is the puncturing action. The high precision and repeatability of the process, make robot systems ideal for this task. Also, the trauma induced...
TABLE VI: Task list with automotive system specifications presented in natural language

<table>
<thead>
<tr>
<th>Task</th>
<th>Natural Language Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Safety</td>
<td>In the first 40 seconds, vehicle speed should always be less than 160.</td>
</tr>
<tr>
<td>2. Reachability</td>
<td>In the first 30 seconds, vehicle speed should go over 120.</td>
</tr>
<tr>
<td>3. Stabilization</td>
<td>At some point in time in the first 30 seconds, vehicle speed will go over 100 and stay above for 20 seconds.</td>
</tr>
<tr>
<td>4. Recurrence</td>
<td>At every point in time in the first 40 seconds, vehicle speed will go over 100 in the next 10 seconds.</td>
</tr>
<tr>
<td>5. Implication</td>
<td>If, within 40 seconds, vehicle speed is above 100 then within 30 seconds from time 0, engine speed should be over 3000.</td>
</tr>
<tr>
<td>6. Reactive Response</td>
<td>If, at some point in time in the first 40 seconds, vehicle speed goes over 80 then from that point on, for the next 30 seconds, engine speed should be over 4000.</td>
</tr>
<tr>
<td>7. Conjunction</td>
<td>In the first 40 seconds, vehicle speed should be less than 100 and engine speed should be under 4000.</td>
</tr>
<tr>
<td>8. Non-strict sequencing</td>
<td>At some point in time in the first 40 seconds, vehicle speed should go over 80 and then from that point on, for the next 30 seconds, engine speed should be over 4000.</td>
</tr>
<tr>
<td>9. Conjunction</td>
<td>If, at some point in time in the first 40 seconds, vehicle speed goes over 80 then from that point on, if within the next 20 seconds the engine speed goes over 4000, then, for the next 30 seconds, the vehicle speed should be over 100.</td>
</tr>
</tbody>
</table>

around the region is much lower and therefore the recovery process for the patient is quicker. To complete the puncturing action, the robot has to move towards the puncturing location. Test the tissue for various indicators to calibrate for optimal puncture, bring the puncturing needle to a perpendicular position and, finally, puncture with correct force and angle. If the force or angle is miscalculated, it might pose unintended harm to the patient. Consider the specifications from [17] that should hold on a serial manipulator for puncturing:

1) From [17]: The force applied to the patient by the end effector is always less than a given threshold, except for the puncturing subtask. Formally, assuming that the operation time is 30 seconds, we have: $\phi_{s1} = \square_{[0,30]}(\neg\text{puncturing} \rightarrow f \leq f_{\text{max}})$.

2) From [17]: The task is feasible, and the position of the needle once it stops is inside the target region $R$. Formally, assuming that the operation time is 40 seconds, we have: $\phi_{s2} = \diamond_{[0,40]}(\text{Stop} \land \text{needle} \in R)$.

3) Also, other requirements can be expressed for such a system. For example, the end effector speed should not be less than $v_{\text{min}}$ and should not be greater than $v_{\text{max}}$. Formally: $\phi_{s3} = \square_{[0,40]}(v_{\text{min}} < v_{\text{eff}} < v_{\text{max}})$

The ViSPEC tool is utilized to develop the specifications for the robotic manipulator. For $\phi_{s1}$, the specification is presented in Fig. 6. We assume that $f_{\text{max}} = 10$. The visual representations with ViSPEC for item 2 and 3 are presented in the technical report [12].

![Fig. 6: The graphical formalism for $\phi_{s1}$.](image)

B. Quadcopter

In recent years, quadcopters and other unmanned aerial vehicles (UAVs) have become a major focus for research both in the academic community and industry. Among others, they are used in military operations, nuclear disaster assessment, firefighting and entertainment. The challenges faced in developing these devices and their control algorithms come from the flight dynamics and the highly dynamical environment that they operate in. Also, as the complexity of these devices increases, so do the performance and reliability requirements.

Consider the following specifications for a quadrotor:

1) The absolute value of the pitch and roll angle should always be below certain thresholds. Formally, assuming that the operation time is 40 seconds, we have: $\phi_{q1} = \square_{[0,40]}(|\alpha| < \alpha_{\text{max}}) \land \square_{[0,40]}(|\beta| < \beta_{\text{max}})$.

2) If distance to the target region is smaller than a certain threshold $d$, then for the next 20 seconds, the speed should not exceed $v_{\text{max}}$. Formally, assuming that the operation time is 40 seconds, we have: $\phi_{q1} = \diamond_{[0,40]}(d_{\text{dist}} < d \rightarrow \square_{[0,20]}(v < v_{\text{max}}))$.

The ViSPEC tool is utilized to develop the specifications for the quadrotor. For $\phi_{q1}$, the specification is presented in Fig. 7. We assume that $\alpha_{\text{max}} = 45\,\text{deg}$, $\beta_{\text{max}} = 45\,\text{deg}$ and $\gamma_{\text{max}} = 60\,\text{deg}$. The visual representations with ViSPEC for item 2 is presented in the technical report [12].

![Fig. 7: The graphical formalism for $\phi_{q1}$.](image)

VII. CONCLUSION AND FUTURE WORK

As robots and other cyber-physical systems become more complex and ubiquitous, so does the need for better testing and verification. A set of formal methods that improve this process require some formal representation of system specifications. In this work, a graphical formalism and a tool
Fig. 8: Subject accuracy grades over tasks for both the expert and non-expert cohorts.

that enables users to easily develop formal specifications are presented. The ViSPEC tool enables users who have little to no mathematical training in formal logics to develop formal specifications, as was verified by a usability study that was conducted in order to evaluate the usefulness of the tool and to get insights on potential improvements. The tool was utilized to formalize specifications for two robots.

Last but not least, we would like to investigate if the potential inaccuracies of the specifications that users generate with the tool can be attributed mainly to the inherent ambiguity of the natural language descriptions which were given, or if not, which other factors contribute and to what extent. Thus, in an improved usability study, we aim towards exploring alternative methods of generation of requirements from engineers for a system, that do not involve the administration of a natural language description by the experimenter. This would enable us to study to what extent inherent natural language ambiguity causes the observed less-than-perfect accuracy that is sometimes, even if rarely, exhibited.

ACKNOWLEDGMENT: Partial support under NSF awards CNS-1319560, CNS-1116136, IIP-1454143, IIP-1361926 and the NSF I/UCRC Center for Embedded Systems. We thank all the participants in the usability study and the reviewers for the detailed reviews.

REFERENCES


