VISPEC: A graphical tool for elicitation of MTL requirements

Bardh Hoxha[1], Nikolaos Mavridis[2] and Georgios Fainekos[1]

***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

[1] Bardh Hoxha and Georgios Fainekos are with the School of Computing, Informatics and Decision Systems Engineering, Arizona State University {bhoxha,fainekos}@asu.edu

[2]Nikolaos Mavridis is with the Interactive Robots and Media Lab and NCSR Demokritos nmav@alum.mit.edu

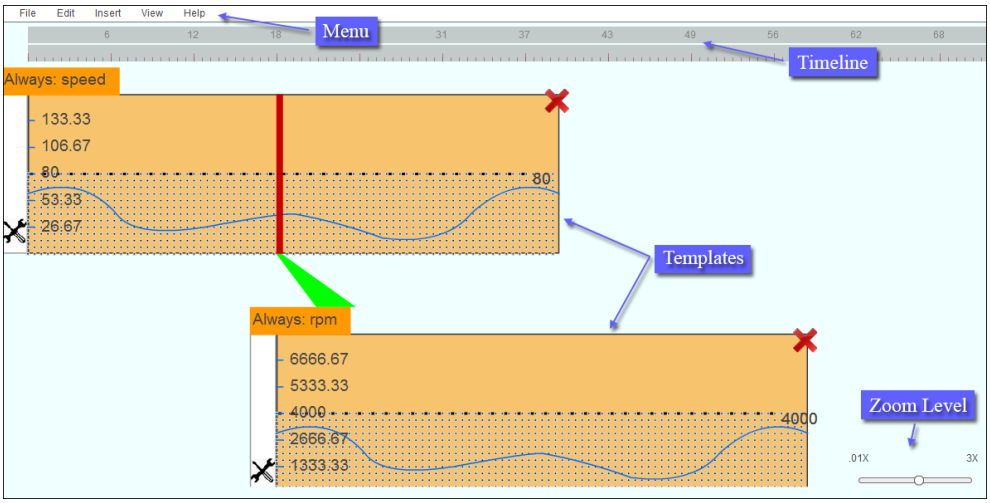

Fig. 1: Overview of the graphical user interface of the MTL specification tool. The example shown represents the MTL specification $\phi = \square_{[0,40]}((speed < 80) \rightarrow \square_{[0,40]}(rpm < 4000))$.

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 representetion 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 $\square_{[0,5]}((speed > 100) \rightarrow \square_{[0,5]}(rpm > 4000))$. 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 visualy 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:

$$
\begin{aligned}
\text{S} &\longrightarrow \neg\text{T} \mid \text{T} \\
\text{T} &\longrightarrow \text{A} \mid \text{B} \mid \text{C} \\
\text{A} &\longrightarrow \text{P} \mid (\text{P} \wedge \text{A}) \mid (\text{P} \Rightarrow \text{A}) \\
\text{B} &\longrightarrow \square_{\mathcal{I}}\text{D} \mid \Diamond_{\mathcal{I}}\text{D} \\
\text{C} &\longrightarrow \square_{\mathcal{I}}\Diamond_{\mathcal{I}}\text{D} \mid \Diamond_{\mathcal{I}}\square_{\mathcal{I}}\text{D} \\
\text{D} &\longrightarrow p \mid (p \rightarrow \text{A}) \mid (p \wedge \text{A}) \mid (p \rightarrow \text{B}) \mid (p \wedge \text{B}) \\
\text{P} &\longrightarrow p \mid \square_{\mathcal{I}}p \mid \Diamond_{\mathcal{I}}p
\end{aligned}
$$

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.

In Fig. 1, the user interface of the tool is presented along with its most critical components. The user interface is composed of a menu, horizontal timeline, rectangular blocks called templates, and a zoom scroll. While the passage of time is represented horizontally, the sequence of events is presented vertically. The formulas are generated from templates as well as the connections between them.

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

[1]Available at https://sites.google.com/a/asu.edu/s-taliro/vispec

TABLE I: Classes of specifications expressible with the graphical formalism

Specification Class	Explanation
Safety	Specifications of the form $\Box\phi$ used to define specifications where ϕ should always be true.
Reachability	Specifications of the form $\Diamond\phi$ used to define specifications where ϕ should be true at least once in the future (on now).
Stabilization	Specifications of the form $\Diamond\Box\phi$ used to define specifications that, at least once, ϕ should be true and from that point on, stay true.
Recurrence	Specifications of the form $\Box\Diamond\phi$ used to define specifications that, it is always the case, that at some point in the future, ϕ is true.
Implication	Specifications of the form $\phi \rightarrow \psi$ requires the ψ should hold when ϕ is true.
Reactive Response	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.
Conjunction	Specifications of the form $\phi \wedge \psi$ used to define the conjunction of two sub-specifications
Non-strict Sequencing	Specifications of the form $N(\phi \wedge M\psi)$, 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.

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* ($\Box$), *At Least Once* ($\Diamond$), *Eventually Always* ($\Diamond\Box$), *Repeatedly Often and Finally* ($\Box\Diamond$), 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* ($\Diamond\Box$) and *Repeatedly Often and Finally* ($\Box\Diamond$). 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* ($\Box$) 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. 2. 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 formulas called Reachability MTL specifications is presented. Specifically, the specification* $\phi_2 = \Diamond_{[0,39]}(speed > 100)$. *The formula states that eventually, within the next 39 seconds, the vehicle speed will go over 100. The corresponding graphical formalism for this formula is presented in Fig. 3. Again, in regards to the specification, the signal can be of any shape as long as at one point, within the timing bounds of the temporal operator, it is above the 100 threshold.*

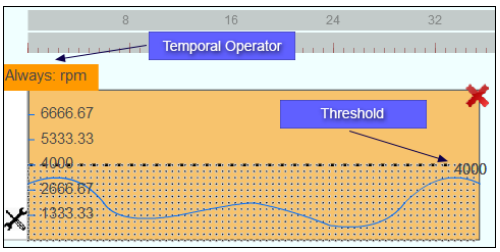

Fig. 2: Example 1: The graphical formalism for the *Safety* MTL specification $\phi_1 = \Box_{[0,36]}(rpm < 4000)$.

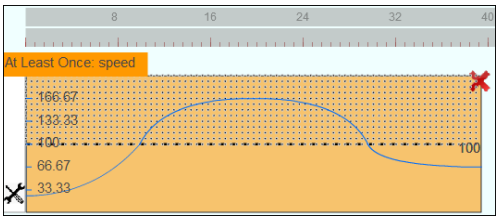

Fig. 3: Example 2: The graphical formalism for the *Reachability* MTL specification $\phi_2 = \Diamond_{[0,39]}(speed > 100)$.

For the *Eventually Always* ($\Diamond\Box$) operator, at least once in the timing interval of the eventually operator, the signal should go above the threshold and stay there for the entire timing interval of the always operator. Two types of shading will indicate the timing bounds of the MTL operators.

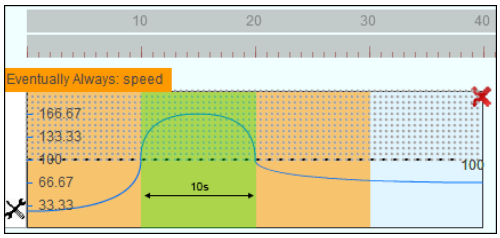

Fig. 4: Example 3: The graphical formalism for the MTL specification $\phi_3 = \Diamond_{[0,30]}\Box_{[0,10]}(speed > 100)$.

Example 3 *Consider the specification* $\phi_3 = \Diamond_{[0,30]}\Box_{[0,10]}(speed > 100)$. *The formula states that at some point in the first 30 seconds, the vehicle speed will go over 100 and stay above for 10 seconds. The corresponding graphical formalism for this formula is presented in Fig. 4.*

For the *Repeatedly Often and Finally* ($\Box\Diamond$) operator, an oscillating signal is presented where two types of shading indicate the timing intervals for each MTL operator. Consider the following example:

Example 4 *The specification* $\phi_4 = \Box_{[0,30]}\Diamond_{[0,10]}(speed > 100)$ *is presented. The formula states that at every timestep of the simulation in the first 30 seconds, the speed will go over 100 within the next 10 seconds. The corresponding graphical formalism for this formula is presented in Fig. 5. No matter how far to the left or right the green shaded region is moved, contained within the orange region, there is always a point where the signal is above the threshold. Recall that the signal is automatically generated so that it satisfies the options previously selected.*

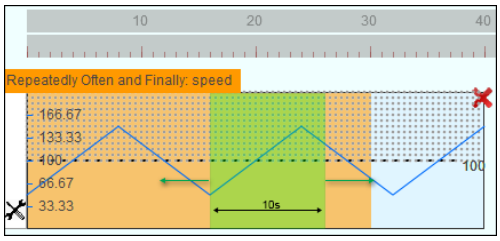

Fig. 5: Example 4: The graphical formalism for the MTL specification $\phi_4 = \Box_{[0,30]}\Diamond_{[0,10]}(speed > 100)$.

The next important concept in this graphical formalism is the relationship between templates.

First, the sequence relationship between two templates is presented. Assume that the first template is already created. If another template is added below it, then an order in the execution of the events is defined. The second template is only considered if the first template is evaluated to true. Formally, there is an implication relationship from the first template to the second. Consider the following example:

Example 5 *The specification* $\phi_5 = (\Diamond_{[0,40]}(speed > 100)) \rightarrow (\Diamond_{[0,30]}(rpm > 3000))$ *is presented. The formula states that if, within 40 seconds, the vehicle speed is above 100 then within 30 seconds from time 0, the engine speed should be over 3000. The corresponding graphical formalism for this formula is presented in Fig. 6.*

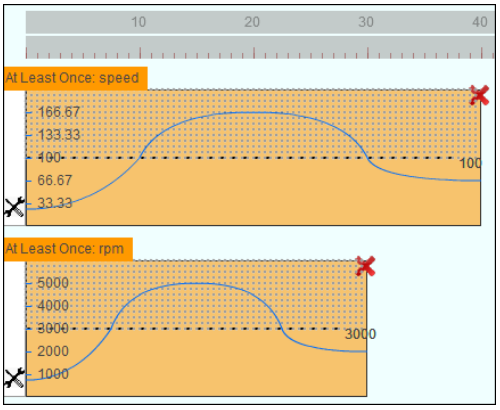

Fig. 6: Example 5: The graphical formalism for the MTL specification $\phi_5 = (\Diamond_{[0,40]}(speed > 100)) \rightarrow (\Diamond_{[0,30]}(rpm > 3000))$.

A second type of relationship enables the user to establish conjunction between two events. To achieve this, templates can be grouped. This is indicated by a bold black box. Doing so requires that both templates evaluate to true. Consider the following example:

Example 6 *Specification* $\phi_6 = (\Box_{[0,40]}(speed < 100)) \wedge (\Box_{[0,40]}(rpm < 4000))$. *The formula states that, within 40 seconds, the vehicle speed should be less than 100 and the engine speed should be under 4000. The corresponding graphical formalism for this formula is presented in Fig. 7.*

The third type of template relationship enables the user to establish relative timing between two templates. Consider the following example:

Example 7 *Specification* $\phi_7 = \Box_{[0,40]}((speed < 80) \rightarrow \Box_{[0,40]}(rpm < 4000))$. *Here, the nested specification* $\Box_{[0,40]}(rpm < 4000)$ *is evaluated every time* $(speed < 80)$ *is true. This formula is represented in the formalism with nested templates, otherwise referred to as parent and child templates. The second template is tabbed and connected to the first template using a green indicator. In the GUI, such a nested template is initiated by clicking on the signal of the parent template. The corresponding graphical formalism is presented in Fig. 8.*

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 formula. There is a bijection between the visual representation of a specification and the MTL formula. An overview of the process is provided in Fig. 10.

An example of the tree structure for MTL formula $\phi = \square(a \wedge \diamond b) \rightarrow (\square c \wedge \diamond(d \rightarrow (a \wedge \square b)))$ is shown in Fig. 11. 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.

IV. USABILITY STUDY

A. Hypotheses

The aim of the study is to evaluate whether VISPEC enables users to develop formal specifications. Two groups were considered:

1) Non-expert users: These are users who declared that they have no experience in working with requirements.
2) Expert users: These are users who declared that they have experience working with system requirements. Note that they do not necessarily have experience in writing requirements using formal logics.

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.

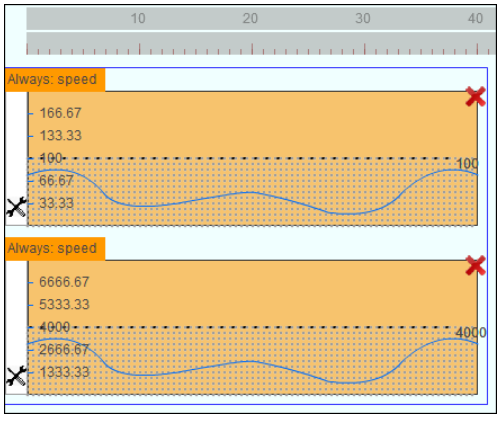

Fig. 7: Example 6: The graphical formalism for the MTL specification $\phi_6 = (\square_{[0,40]}(speed < 100)) \wedge (\square_{[0,40]}(rpm < 4000))$.

Algorithm 1 WriteMTL - Algorithm for generating the MTL formula given a tree structure of the graphical formalism

Input: Tree Structure $T = \langle V, E \rangle$ 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; string ϕ
Output: ϕ

```
function WRITEMTL(T, φ)
    C ← T.getChildren.
    sC ← size(C)
    for node i in C do
        φ ← CONC(φ, i.Op)
        if i.isParent then
            if not(i.S.isEmpty) then
                subC ← t.getChildren(i)
                if i.G == subC(1).G then
                    φ ← CONC(φ, '(', i.S, '∧')
                else
                    if i.isParent then
                        φ ← CONC(φ, '(', i.S,'→ (')
                    else
                        φ ← CONC(φ, '(', i.S,'→')
                    end if
                end if
                φ ← WRITEMTL(i.subtree,φ)
                if i.isParent then
                    if i.G == subC.G then
                        φ ← CONC(φ,')')
                    else
                        φ ← CONC(φ,'))')
                    end if
                else
                    if sC > 1 and i ≠ sC then
                        φ ← CONC(φ,') →')
                    else
                        φ ← CONC(φ,')')
                    end if
                end if
            else
                φ ← CONC(φ,'(')
                φ ← WRITEMTL(i.subtree,φ)
                if i ≠ sC then
                    φ ← CONC(φ,') →')
                else
                    φ ← CONC(φ,')')
                end if
            end if
        else
            φ ← CONC(φ,i.S)
            if i ≠ sC then
                φ ← CONC(φ,'∧')
            else
                φ ← CONC(φ,'→')
            end if
        end if
    end for
end function
```

TABLE II: Hypotheses and test results with level of significance $\alpha = 0.05$. User groups as defined in section IV.A.

Hypothesis		Reject *null* hypothesis
$1a$	Non-expert users are able to define formal requirements accurately using formal logics such as MTL.	
$1b$	Non-expert users are able to define formal requirements accurately using the Visual Specification Tool.	Yes
$2a$	Expert users from the industry are able to define formal requirements accurately using formal logics such as MTL.	
$2b$	Expert users from the industry are able to define formal requirements accurately using the Visual Specification Tool.	Yes
3_{alt}	The mean grade per user for expert users is greater the mean grade per user for non-expert users.	Yes
Tx_{alt}	The mean grade per task x for industry users is greater than to the mean grade per task x for non-expert users.	Partially

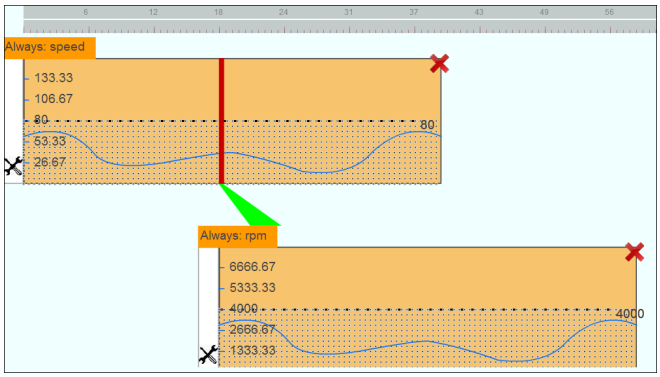

Fig. 8: Example 7: The graphical formalism for the MTL specification $\phi_7 = \square_{[0,40]}((speed < 80) \rightarrow \square_{[0,40]}(rpm < 4000))$.

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.

TABLE III: Hypothesis $1b$ Subject Demographics

Freshman	2	Computer Science	5	Male	12
Sophomore	2	Software Engineering	3	Female	8
Junior	5	Electrical Engineering	3		
Senior	5	Mechanical Engineering	6		
Masters	4	Engineering, other	3		
PhD	2				

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. 9. It can be observed that overall, the mean grade per task for both cohorts is high. Consider the mean grade per task as a random variable $\bar{X}$. Specifically, $\bar{X} : \Omega \rightarrow \mathbb{R}$, where $\Omega \in \{y : 1 \leq y \leq 5\}$. In Figure 12, we present the survival function $S_{\bar{X}}(x) = 1 - F_{\bar{X}}(x) = 1 - P(\bar{X} \leq x) = P(\bar{X} > x)$ based on sample data. Note that x is the threshold of mean grade accuracy.

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 $\bar{Y}$. Specifically, $\bar{Y} : \Omega \rightarrow \mathbb{R}$, where $\Omega \in \{y :$

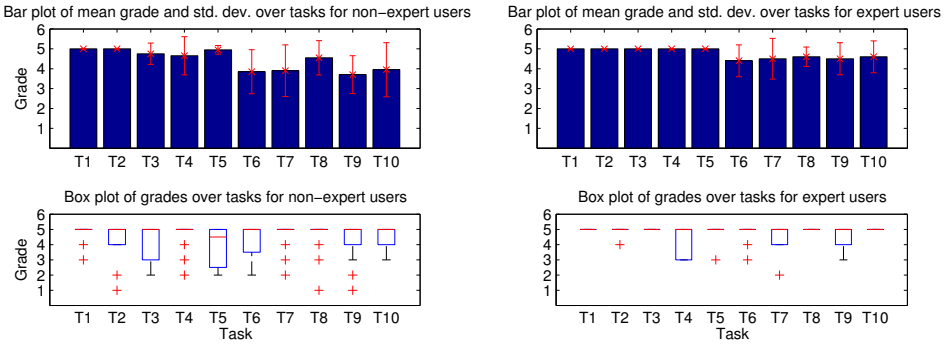

Fig. 9: Subject accuracy grades over tasks for both the expert and non-expert cohorts.

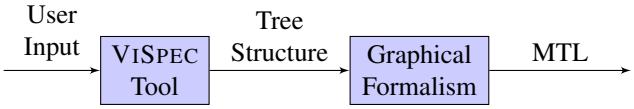

Fig. 10: The specification development process using VISPEC

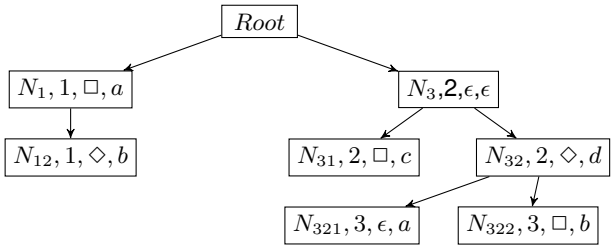

Fig. 11: The corresponding tree structure for formula $\phi = \Box(a \wedge \Diamond b) \rightarrow (\Box c \wedge \Diamond(d \rightarrow (a \wedge \Box b)))$ where a,b,c and d are predicates. Each node is composed of a node name, group number, temporal operator, and predicate. The symbol ϵ indicates empty parameters.

$1 \leq y \leq 5\}$. 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. In figure 14, we plot the non-expert data against a fitted normal distribution and the corresponding Q-Q plot. If we assume that the data constitute a random sample from a normal distribution, i.e. $\bar{Y} \sim \mathcal{N}$, we can use the t-statistic to test the hypothesis. We reject the null hypothesis with a p-value very close to 0.

3) Hypothesis $2b$*:* Similarly, we test Hypothesis $2b$ for the expert cohort. Hypothesis $2b$ is reduced to the null hypothesis: the mean grade per user is less than or equal to three for expert users. We test for normality as in the previous case and all three test fail to reject the null hypothesis that the data follows the normal distribution.

Consider the average grade per user as a random variable $\bar{Z}$. Specifically, $\bar{Z} : \Omega \rightarrow \mathbb{R}$, where $\Omega \in \{y : 1 \leq y \leq 5\}$. The sample data from 10 subjects has a mean grade of 4.76 and standard deviation of 0.26. In figure 14, we plot the non-expert data against a fitted normal distribution and the corresponding Q-Q plot. If we assume that the data constitute a random sample from a normal distribution, i.e. $\bar{Z} \sim \mathcal{N}$ we

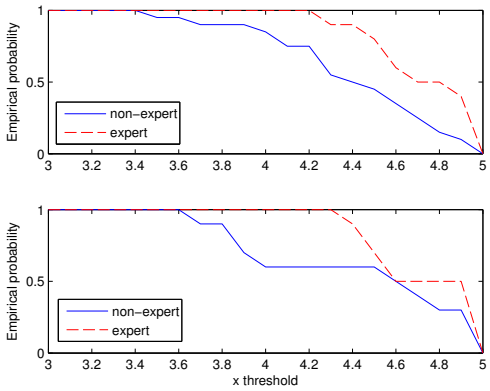

Fig. 12: **Top**: The empirical probability that the mean grade per user is greater than threshold x for the non-expert and expert subjects, i.e., $P(\bar{Y} > x)$.
Bottom: The empirical probability that the mean grade per task is greater than threshold x for the non-expert and expert subjects, i.e., $P(\bar{X} > x)$.

can use the t-statistic to test the hypothesis.We reject the null hypothesis with a p-value very close to 0.

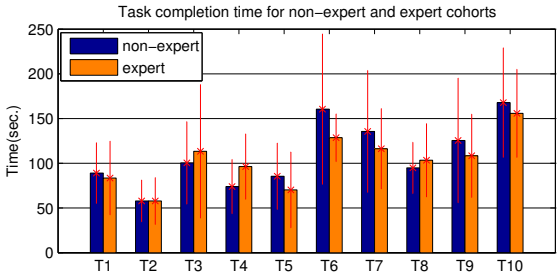

Fig. 13: Example 2: The graphical formalism for the *Reachability* MTL specification $\phi_2 = \Diamond_{[0,39]}(speed > 100)$.

4) Hypothesis 3_{alt}*:* To test Hypothesis 3_{alt}, we conduct a two sample t-test. The p-value returned from the test is 0.0024 and for a significance level of 0.01, we reject the null hypothesis. Therefore we claim that the mean grade per user for expert users is greater than the mean grade per user for non-experts.

5) Hypothesis Tx*:* 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

TABLE IV: VISPEC improvements

#	Improve...	Prime Indicators
1.	the process of creating child templates	misclicks; user feedback
2.	the tutorial by placing more emphasis on the difference between implication and conjunction between templates	task accuracy grade
3.	the visual representation of grouped templates	task accuracy grade; user feedback

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 Tx_{null} with $\alpha = 0.05$

x	Rej. Tx_{null}	p-val.	Conclusion
4	No	0.065	potentially true with more investigation
5	No	0.165	false
6	No	0.074	potentially true with more investigation
7	No	0.100	potentially true with more investigation
8	No	0.424	false
9	Yes	0.016	true
10	No	0.063	potentially true with more investigation

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. 9.

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. 13, 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 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: $\phi_{s1} = \square_{[0,30]}(\neg puncturing \rightarrow f \leq f_{max})$.
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: $\phi_{s2} = \diamondsuit_{[0,40]}(Stop \wedge needle \in R))$.
3) Also, other requirements can be expressed for such a system. For example, the end effector speed should not be less than v_{min} and should not be greater than v_{max}. Formally: $\phi_{s3} = \square_{[0,40]}(v_{min} < v_{eff} < v_{max})$

The VISPEC tool is utilized to develop the specifications for the robotic manipulator. For ϕ_{s1}, the specification is presented in Fig. 15. We assume that $f_{max} = 10$. For ϕ_{s2}, the specification is presented in Fig. 17. We assume that $needle \in R \iff 5 < n_x < 10 \wedge 5 < n_y < 10$, where n_x, n_y are the x and y coordinates for the needle. For ϕ_{s3}, the specification is presented in Fig. 18. We assume that $v_{min} = 10$ and $v_{max} = 20$.

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

The VISPEC tool is utilized to develop the specifications for the quadrotor. For ϕ_{q1}, the specification is presented in Fig. 16. We assume that $\alpha_{max} = 45\,\text{deg}$, $\beta_{max} = 45\,\text{deg}$ and $\gamma_{max} = 60\,\text{deg}$. For ϕ_{q2}, the specification is presented in Fig. 19. We assume that $d = 5$ and $v_{max} = 10$. For ϕ_{s3}, the specification is presented in Fig. 18. We assume that $v_{min} = 10$ and $v_{max} = 20$.

TABLE VI: Task list with automotive system specifications presented in natural language

Task	Natural Language Specification
1. Safety	In the first 40 seconds, vehicle speed should always be less than 160.
2. Reachability	In the first 30 seconds, vehicle speed should go over 120.
3. Stabilization	At some point in time in the first 30 seconds, vehicle speed will go over 100 and stay above for 20 seconds.
4. Recurrence	At every point in time in the first 40 seconds, vehicle speed will go over 100 in the next 10 seconds.
5. Recurrence	It is not the case that, for up to 40 seconds, the vehicle speed will go over 100 in every 10 second period.
6. Implication	If, within 40 seconds, vehicle speed is above 100 then within 30 seconds from time 0, engine speed should be over 3000.
7. Reactive Response	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.
8. Conjunction	In the first 40 seconds, vehicle speed should be less than 100 and engine speed should be under 4000.
9. Non-strict sequencing	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.
10. Long sequence	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.

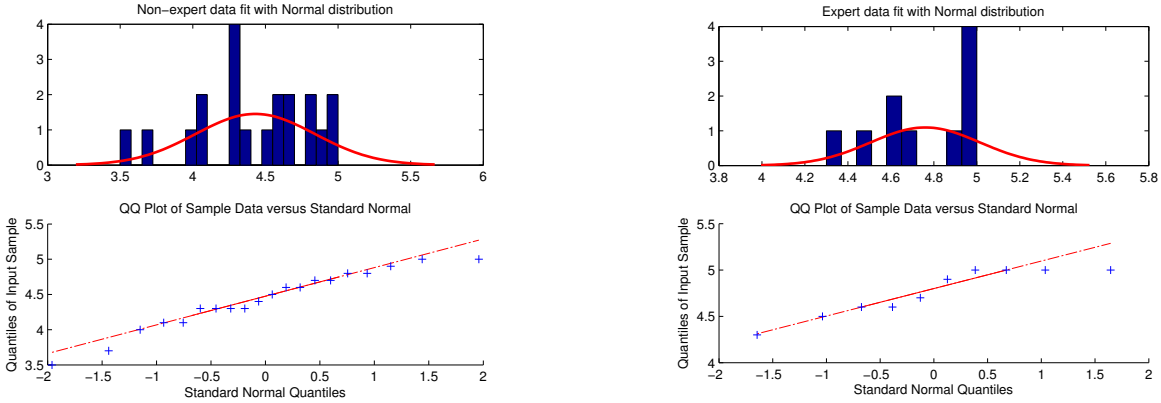

Fig. 14: Subject data fit with a normal distribution and the corresponding Q-Q plot.

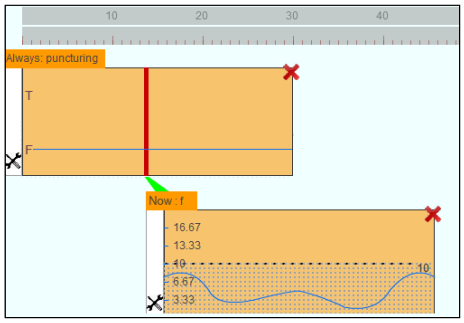

Fig. 15: The graphical formalism for ϕ_{s1}.

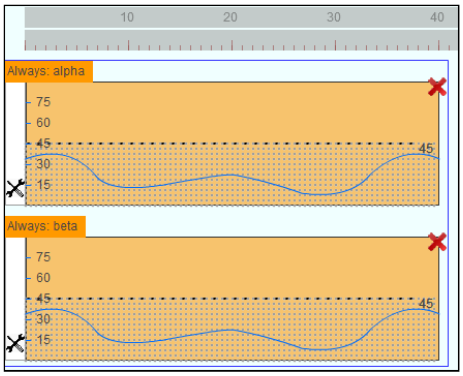

Fig. 16: The graphical formalism for ϕ_{q1}.

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 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

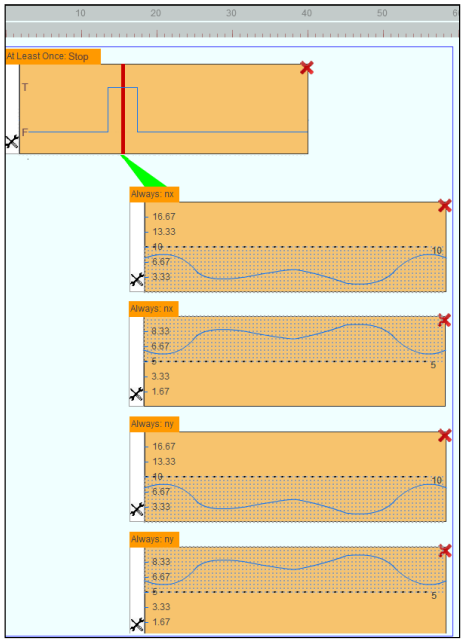

Fig. 17: The graphical formalism for ϕ_{s2}.

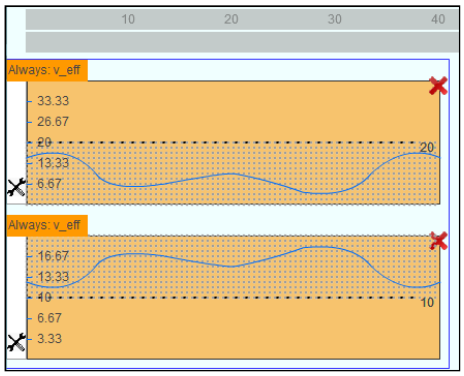

Fig. 18: The graphical formalism for ϕ_{s3}.

reviewers for the detailed reviews.

REFERENCES

[1] A. Alfonso, V. Braberman, N. Kicillof, and A. Olivero. Visual timed event scenarios. In *Proceedings of the 26th Int. Conference on Software Engineering*, pages 168–177. IEEE Computer Society, 2004.

[2] Y. S. R. Annapureddy, C. Liu, G. E. Fainekos, and S. Sankaranarayanan. S-taliro: A tool for temporal logic falsification for hybrid systems. In *Tools and algorithms for the construction and analysis of systems*, volume 6605 of *LNCS*, pages 254–257. Springer, 2011.

[3] M. Autili, P. Inverardi, and P. Pelliccione. Graphical scenarios for specifying temporal properties: an automated approach. *Automated Software Engineering*, 14(3):293–340, 2007.

[4] X. Chen, E. Abraham, and S. Sankaranarayanan. Flow*: An analyzer for non-linear hybrid systems. In *Computer-Aided Verification (CAV)*, volume 8044 of *LNCS*, pages 258–263. Springer-Verlag, 2013.

[5] Y. Deng, A. Rajhans, and A. A. Julius. Strong: A trajectory-based verification toolbox for hybrid systems. In *Quantitative Evaluation of Systems*, pages 165–168. Springer, 2013.

[6] A. Donze. Breach, a toolbox for verification and parameter synthesis of hybrid systems. In *Computer Aided Verification*, volume 6174 of *LNCS*, pages 167–170. Springer, 2010.

[7] P. S. Duggirala, S. Mitra, and M. Viswanathan. Verification of annotated models from executions. In *Proc. of the Eleventh ACM Int. Conf. on Embedded Software*, page 26. IEEE Press, 2013.

[8] G. Fainekos, S. Sankaranarayanan, K. Ueda, and H. Yazarel. Verification of automotive control applications using s-taliro. In *Proceedings of the American Control Conference*, 2012.

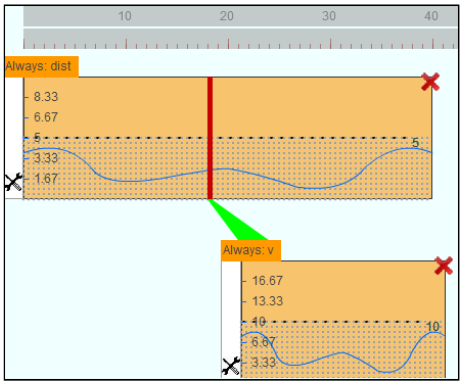

Fig. 19: The graphical formalism for ϕ_{q2}.

[9] G. Frehse, C. L. Guernic, A. Donz, S. Cotton, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang, and O. Maler. Spaceex: Scalable verification of hybrid systems. In *Proceedings of the 23d CAV*, 2011.

[10] G. J. Holzmann. The logic of bugs. In *Proc. of the 10th ACM SIGSOFT symp. on Foundations of soft. eng.*, pages 81–87. ACM, 2002.

[11] B. Hoxha, H. Bach, H. Abbas, A. Dokhanchi, Y. Kobayashi, and G. Fainekos. Towards formal specification visualization for testing and monitoring of cyber-physical systems. In *Int. Workshop on Design and Implementation of Formal Tools and Systems*. October 2014.

[12] B. Hoxha, N. Mavridis, and G. Fainekos. VISPEC: a graphical tool for elicitation of MTL requirements. Available at https://sites.google.com/a/asu.edu/s-taliro/ViSpecTechRpt15.pdf.

[13] Y. Kouskoulas, D. W. Renshaw, A. Platzer, and P. Kazanzides. Certifying the safe design of a virtual fixture control algorithm for a surgical robot. In C. Belta and F. Ivancic, editors, *Hybrid Systems: Computation and Control (part of CPS Week 2013), HSCC'13, Philadelphia, PA, USA, April 8-13, 2013*, pages 263–272. ACM, 2013.

[14] R. Koymans. Specifying real-time properties with metric temporal logic. *Real-Time Systems*, 2(4):255–299, 1990.

[15] H. Kugler, D. Harel, A. Pnueli, Y. Lu, and Y. Bontemps. Temporal logic for scenario-based specifications. In *Tools and Alg. for the Construction and Analysis of Systems*, pages 445–460. Springer, 2005.

[16] C. Lignos, V. Raman, C. Finucane, M. Marcus, and H. Kress-Gazit. Provably correct reactive control from natural language. *Autonomous Robots*, 38(1):89–105, 2015.

[17] R. Muradore, D. Bresolin, L. Geretti, P. Fiorini, and T. Villa. Robotic surgery. *Robotics & Automation Magazine, IEEE*, 18(3):24–32, 2011.

[18] A. Platzer and J.-D. Quesel. KeYmaera: A hybrid theorem prover for hybrid systems. In A. Armando, P. Baumgartner, and G. Dowek, editors, *International Joint Conference on Automated Reasoning*, volume 5195 of *LNCS*, pages 171–178. Springer, 2008.

[19] B. I. Silva and B. H. Krogh. Formal verification of hybrid systems using CheckMate: a case study. In *Proceedings of the American Control Conference*, volume 3, pages 1679 – 1683, June 2000.

[20] M. H. Smith, G. J. Holzmann, and K. Etessami. Events and constraints: A graphical editor for capturing logic requirements of programs. In *Requirements Engineering, 2001. Proceedings. Fifth IEEE International Symposium on*, pages 14–22. IEEE, 2001.

[21] S. Srinivas, R. Kermani, K. Kim, Y. Kobayashi, and G. Fainekos. A graphical language for LTL motion and mission planning. In *Robotics and Biomimetics (ROBIO), 2013 IEEE International Conference on*, pages 704–709. IEEE, 2013.

[22] S. Tripakis and T. Dang. *Model-Based Design for Embedded Systems*, chapter Modeling, Verification and Testing using Timed and Hybrid Automata, pages 383–436. CRC Press, 2009.

[23] R. Vinter, M. Loomes, and D. Kornbrot. Applying software metrics to formal specifications: A cognitive approach. In *Software Metrics Symposium, 1998. Metrics 1998. Proceedings. Fifth International*, pages 216–223. IEEE, 1998.

[24] T. Wongpiromsarn, S. Mitra, A. Lamperski, and R. M. Murray. Verification of periodically controlled hybrid systems: Application to an autonomous vehicle. *ACM Trans. Embed. Comput. Syst.*, 11(S2):53:1–53:24, Aug. 2012.

[25] B. Yordanov, J. Tmov, I. ern, J. Barnat, and C. Belta. Formal analysis of piecewise affine systems through formula-guided refinement. *Automatica*, 49(1):261 – 266, 2013.

[26] P. Zhang, B. Li, and L. Grunske. Timed property sequence chart. *Journal of Systems and Software*, 83(3):371–390, 2010.