Eurographics Conference on Visualization (EuroVis), Posters Track (2017)
A. Puig Puig and T. Isenberg (Editors)

Visualization of Temporal Logic Specifications

J. Scott-Brown & A. Papachristodoulou

University of Oxford, UK

Abstract

The desired behaviour of a system can be formally specified using a temporal logic, and used to check whether a candidate design meets this specification, or to automatically design a system that does. Such techniques are in principle applicable to a wide range of types of systems, but their current extent of use does not fulfill their potential. One limitation has been the difficulty of writing and reading formal specifications. We present preliminary work on a visual method of viewing and editing specifications that is intended to address this challenge.

Categories and Subject Descriptors (according to ACM CCS): H.5.2 [Information Interfaces and Presentation]: Picture/Image Generation—User Interfaces

1. Introduction

The process of *model checking* verifies whether a model of a system satisfies a formal specification [Bai08]. One application is to validate a design by showing that there are no ways in which the corresponding model may fail to function as intended. Related techniques can also be used to automatically design systems to meet a specification, perform real-time surveillance to detect when a specification has been violated, or to choose a series of control input that will prevent the specification from being violated. These techniques are applicable to many domains, including software, electronics or hardware, embedded or cyberphysical systems, and synthetic biological circuits [MMR*12]. A specicification can also be used to filter a set of time-series, and select only those of interest.

For a specification to capture the behaviour of a system over time, it is necessary to use a temporal logic. There are many temporal logics that differ in their details, but have the common feature of extending Boolean logic by introducing additional, temporal, operators. Two such operators are the 'finally' operator $\diamond_{[a,b]}\phi$, which asserts that there *exists a time* in the interval $[a,b]$ during which the proposition ϕ is true, and the 'globally' operator $\Box_{[a,b]}\phi$, which asserts that the proposition ϕ is true *for all times* in the interval $[a,b]$.

In many application areas, the state of the system of interest is most naturally expressed using real-valued quantities, and the relevant properties of the system can be expressed as inequalities over one of the corresponding quantities. For instance, in electronics, the model may be a set of differential equations describing the evolution of currents and voltages in the circuit, and the specifications may specify circumstances in which particular voltages are above (logical ON) or below (logical OFF) particular thresholds. In a synthetic biology applications, the state variables are likely to be the concentrations or molecule counts of proteins or other species, and the models may be differential equations or Markov processes.

Whilst formal methods are in principle widely applicable, their use is currently limited by several challenges. One challenge is the difficulty of reading and writing specifications. Written conventionally, these are an intimidating string of symbols (*e.g.* $\wedge \vee \Box \diamond \neg$). The problem of writing specifications could in principle be avoided by instead having a user annotate trajectories as showing or failing to show a desired property, and attempting to learn the corresponding specification from these examples [KJA*14]. However, this does not help the user to verify that the learned specification actually captured what was intended.

An alternative approach, which we adopt, it to represent specifications visually. Whilst there has been some previous work on visualising specifications expressed in temporal logics, most considered systems that are characterized by discrete states rather than continuous signals. An exception is the VisSpec system [HMF15], which represents each temporal operator, or combination of temporal operators, as a single block labelled with the corresponding property (*e.g.* 'always' or 'at least once'). Users pick a block from a list of 'templates', then replace placeholders with numerical values. The main innovation of the work presented here is to map each operator of the symbolic logic expression directly onto a graphical element that can be directly manipulated. We are also able to represent a different set of formulae, including some that VisSpec cannot (*e.g.* $\Box \diamond \Box \phi$).

2. Visualization Approach

We present the same specification in three ways: *symbolically*, using the conventional mathematical notation; *textually*, using an ex-

DOI: 10.2312/eurp.20171183

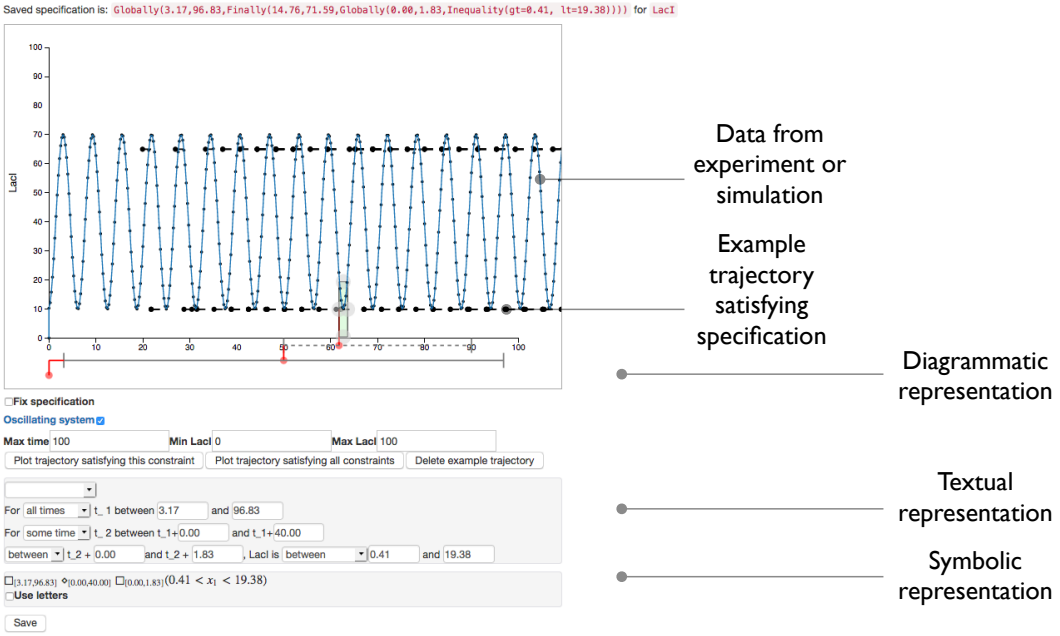

Figure 1: *Representation of a single specification. The specification is represented symbolically, textually, and diagrammatically, in three coupled representations. The user can interact with the textual specification by editing numerical values in* `value` *elements, or changing qualitative aspects of the specification using* `select` *boxes. The user can interact with the diagram by direct manipulation, clicking and dragging to change the size and position of the rectangle or rails; a context menu allows the addition or deletion of rails. In the diagrammatic view, the same axes are used to plot imported data from simulations or experiments, synthetic trajectories automatically generated to meet the specification, and the diagram itself. The rectangle indicates a region within which the system's trajectory must remain for the specification to be satisfied. Dashed rails represent finally operators: for the specification to be satisfied, there must be a position for the slider on this track such that the rectangle is positioned such that the system's trajectory remains within it. Solid rails represent globally operators: for the specification to be satisfied, for every position on the track it must be possible for the rectangle to be positioned such that the system's trajectory remains within it.*

planation in English; and *visually* as a diagram (Figure 1). These representations are coupled, and manipulating the diagram or editing the English description (parts of which are drop-down `select` boxes or `input` elements) simultaneously updates the other views.

The user is able to generate and plot example trajectories meeting the specification, allowing them to quickly verify that they have specified the behaviour that they intended to. They can observe the effect of editing a specification by generating new trajectories, and superimposing them on the previously generated trajectories.

The results of simulations or experiments can be plotted on the same axes as the specification diagram, and used as a guide when adjusting values in the specification.

Diagrammatic Representation

The basic approach we adopt is to plot trajectories of the system, and to overlay a representation of the constraints imposed by the specification.

We represent a constraint on the values that a variable may take using a rectangle; for the specification to be satisfied by the specification, the system must remain within each rectangle at all times between its start and end times.

We represent the globally and finally operators by rails along which rectangles or other rails can slide. The graphical components (rectangles and rails) compose in the same way as the corresponding symbols, and successive elements are linked. The finally operator indicates uncertainty in the time at which a proposition is true; this is represented diagrammatically by a dashed rail, and for a trajectory to satisfy the specification it must be possible to slide along the rail into a position where the trajectory remains inside the rectangle. The globally operator represents a set of constraints; it indicates that a proposition is true for every time in a range. This is represented diagrammatically by a solid rail, and for a trajectory to satisfy the specification the trajectory must remain inside the rectangle for every position along the rail.

2.1. Conclusion and further work

Future work will focus primarily on improving the expressiveness of the system, so as to increase the range and complexity of specifications it is able to represent.

References

[Bai08] BAIER C.: *Principles of model checking*. MIT Press, Cambridge, Mass, 2008. 1

[HMF15] HOXHA B., MAVRIDIS N., FAINEKOS G.: VISPEC: A graphical tool for elicitation of MTL requirements. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (sep 2015), Institute of Electrical and Electronics Engineers (IEEE). doi:10.1109/iros.2015.7353863. 1

[KJA*14] KONG Z., JONES A., AYALA A. M., GOL E. A., BELTA C.: Temporal logic inference for classification and prediction from data. In *Proceedings of the 17th International Conference on Hybrid Systems: Computation and Control* (New York, NY, USA, 2014), HSCC '14, ACM, pp. 273–282. doi:10.1145/2562059.2562146. 1

[MMR*12] MADSEN C., MYERS C. J., ROEHNER N., WINSTEAD C., ZHANG Z.: Utilizing stochastic model checking to analyze genetic circuits. In *2012 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)* (may 2012), Institute of Electrical and Electronics Engineers (IEEE). doi:10.1109/cibcb.2012.6217255. 1