Improving Formal Methods Visualizations

Avinash Palliyil
Georgia Institute of Technology
Atlanta, USA
apalliyil6@gatech.edu

***Abstract*—Formal methods encompass mathematical techniques for specifying, validating and verifying software and hardware systems. While they are powerful, their use of technical and mathematical notations can be difficult to understand, validate, and debug for both experts and non-experts. Visualizations are an approach to simplifying this process. In this work, we present some early results from an interview study to understand how users of formal methods use visualizations in their workflow, and apply these findings to PENLLOY, a domain-specific visualizer for formal methods.**

***Index Terms*—formal methods, formal specifications, visualizations, diagramming, domain-specific visualizations**

I. INTRODUCTION

Formal methods are powerful tools for solving design problems in complex real-world systems. Despite many potential applications, their complex mathematical notation limits their adoption in software engineering. With the advent of program synthesizers that generate executable programs from human-written specifications, there is a greater need to ensure that these specifications are correct. Developing visualization tools that adequately assist formal modelers in developing specifications can help close this gap [1]. We conducted an interview study with users of formal methods and applied our findings to PENLLOY, a domain-specific visualizer for formal models in the Alloy modeling language [2], which is widely used in education and industrial environments.

II. RELATED AND PRIOR WORK

Extensive prior work has been done on using visual formalisms to model complex systems, such as Statecharts [3] and VisB [4]. Within the Alloy community, prior works have improved Alloy visualizations through layout and transition managers [5], as well as custom hierarchies which dictate how shapes and text can be composed and related [6]. Empirically, Nelson et al. [7] evaluates how students introduce domain-specific visualizations in their models through Sterling [8], and gathers retrospective opinions on their benefits and limitations. On the other hand, our interview study tries to systematically extract some "principles" that make good visualizations for experts of formal methods, inspiring our work on improving current visualizations and evaluating the effects of the improvements.

III. INTERVIEW STUDY

We first conducted a semi-structured interview study with 15 users of formal modeling tools, where we asked questions on how the currently available visualization tools are helpful and not helpful for their work.

A. How are visualizations helpful?

Interviewees identified several ways in which visualizations help them understand, validate, and debug model behavior while iterating a formal model. One interviewee brings up an example where "a bug [in the model] manifests [visually] in an easily recognizable way": if a model is intended to be acyclic in nature, a visualization of an instance that shows a cyclic graph can indicate some mis-specifications.

Visualizations also provide abstraction suitable for non-expert stakeholders. Individual invariants and property checks can be difficult to understand for those unfamiliar with formal logic. The mapping of entities within a system to shapes in a visualized instance helps model authors communicate system design and behavior to these non-experts.

B. How are visualizations not helpful?

The interviews also reveal multiple drawbacks of current tools. One drawback is the lack of "positional consistency" for models with multiple states. For example, Alloy's visualizer makes no attempt to "minimize" the visual difference between different states, making it difficult for stakeholders to notice changes.

Another drawback is weak visual mappings of entities and relationships. Alloy's built-in visualizer represents models as box-arrow diagrams. Another interviewee, who works closely with the Alloy community, comments on this box-arrow style:

> "*It would be really nice to* ***customize to the domain****. If [the domain] is a file system, [there should be] a file icon. . . . I'd like* ***other modes of representing relations*** *other than just arrows, such as having the containment relationship between shapes*".

IV. PENLLOY, A DOMAIN-SPECIFIC VISUALIZER FOR ALLOY

The interview study reveals a need for "domain-specific visualizations" and "positional consistency", where different conceptual domains of formal models could benefit from tailored visual abstractions with minimal positional change between states. We integrate the former into PENLLOY, a domain-specific Alloy visualizer powered by the conceptual visualization tool PENROSE [9]. We chose Alloy because of its prevalence among our interviewees and in educational environments. We chose PENROSE for its ability to reason

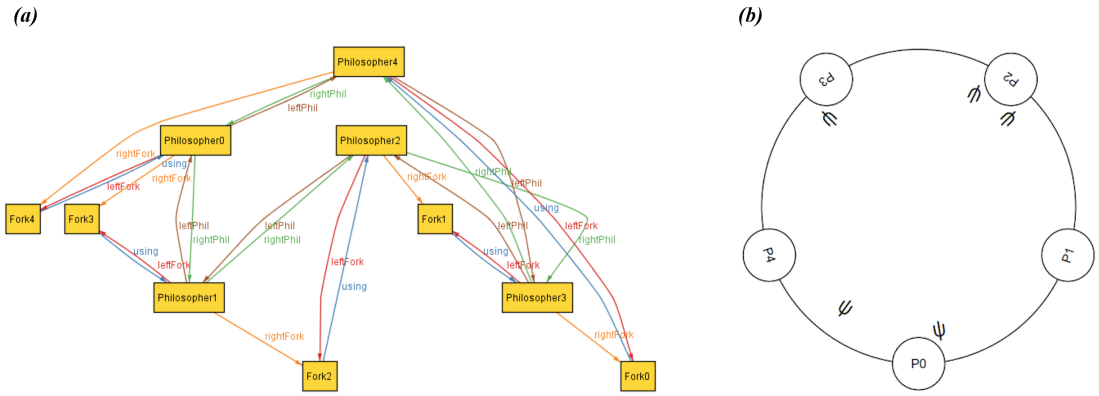

Fig. 1. Visualizations of an instance of the Dining Philosophers Problem ***(a)*** using Alloy's built-in visualizer and ***(b)*** using PENLLOY's domain-specific visualizer. The PENLLOY visualization provides a familiar, customizable view that can help domain experts understand and debug their specifications and explain them to non-experts.

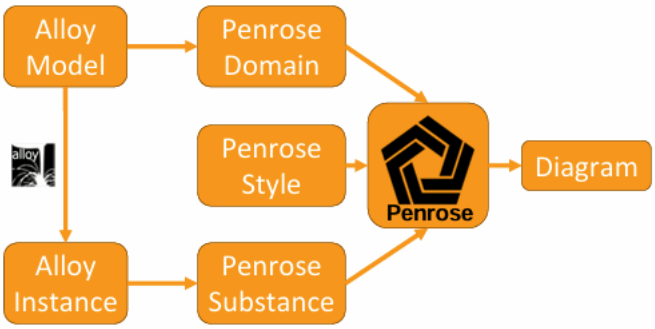

Fig. 2. PENLLOY converts Alloy models and instances into PENROSE *domain* and *substance* programs. Modelers then provide a custom *style* program.

about constraints between shape relations and for its parallels to Alloy: Alloy signatures and relations map well to PENROSE types and predicates. PENLLOY works by translating Alloy models and instances into PENROSE programs, which, when combined with a custom visual *style* file that specifies geometric constraints between shapes, get translated into conceptual diagrams (see Figure 2).

As an example, Figure 1 compares the built-in visualization of Alloy with the domain-specific visualization generated by PENLLOY, on an instance of the Dining-Philosophers problem [10].

V. FUTURE WORK

Our current work with PENLLOY improves the mapping of formal models to visualization by incorporating domain-specificity, but domain-specificity is just one of the visual properties that can help stakeholders. For example, the "consistency" principle addresses the lack of "positional consistency" that the interviewees have identified. We plan to integrate these and other principles into PENLLOY and empirically evaluate them in a classroom setting to better understand their effectiveness on model understanding and debugging. Building on our work in PENLLOY, we also envision incorporating domain-specificity into other formal methods techniques that do not currently use visualization.

REFERENCES

[1] N. Dulac, T. Viguier, N. Leveson, and M.-A. Storey, "On the use of visualization in formal requirements specification," in *Proceedings IEEE Joint International Conference on Requirements Engineering*, 2002, pp. 71–80.

[2] D. Jackson, "Alloy: a language and tool for exploring software designs," *Commun. ACM*, vol. 62, no. 9, p. 66–76, Aug. 2019. [Online]. Available: https://doi.org/10.1145/3338843

[3] D. Harel, "Statecharts: a visual formalism for complex systems," *Science of Computer Programming*, vol. 8, no. 3, pp. 231–274, 1987. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0167642387900359

[4] M. Werth and M. Leuschel, "Visb: A lightweight tool to visualize formal models with svg graphics," in *Rigorous State-Based Methods*, A. Raschke, D. Méry, and F. Houdek, Eds. Cham: Springer International Publishing, 2020, pp. 260–265.

[5] R. Couto, J. C. Campos, N. Macedo, and A. Cunha, "Improving the visualization of alloy instances," *Electronic Proceedings in Theoretical Computer Science*, vol. 284, p. 37–52, Nov. 2018. [Online]. Available: http://dx.doi.org/10.4204/EPTCS.284.4

[6] L. Gammaitoni and P. Kelsen, "Domain-specific visualization of alloy instances," in *Abstract State Machines, Alloy, B, TLA, VDM, and Z*, Y. Ait Ameur and K.-D. Schewe, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 324–327.

[7] T. Nelson, B. Greenman, S. Prasad, T. Dyer, E. Bove, Q. Chen, C. Cutting, T. Del Vecchio, S. LeVine, J. Rudner, B. Ryjikov, A. Varga, A. Wagner, L. West, and S. Krishnamurthi, "Forge: A tool and language for teaching formal methods," *Proc. ACM Program. Lang.*, vol. 8, no. OOPSLA1, Apr. 2024. [Online]. Available: https://doi.org/10.1145/3649833

[8] T. Dyer and J. Baugh, "Sterling: A web-based visualizer for relational modeling languages," in *Rigorous State-Based Methods*, A. Raschke and D. Méry, Eds. Cham: Springer International Publishing, 2021, pp. 99–104.

[9] K. Ye, W. Ni, M. Krieger, D. Ma'ayan, J. Wise, J. Aldrich, J. Sunshine, and K. Crane, "Penrose: from mathematical notation to beautiful diagrams," *ACM Trans. Graph.*, vol. 39, no. 4, Aug. 2020. [Online]. Available: https://doi.org/10.1145/3386569.3392375

[10] E. W. Dijkstra, "Hierarchical ordering of sequential processes," *Acta Informatica*, vol. 1, no. 2, pp. 115–138, Jun 1971. [Online]. Available: https://doi.org/10.1007/BF00289519