Totally Live Programming in Hazel (Progress Report)

CYRUS OMAR, University of Michigan, USA
ANDREW BLINN, University of Michigan, USA
DAVID MOON, University of Michigan, USA

Modern programming environments provide developers with an arsenal of semantic services—for example, type hints, semantic navigation, code completion, automated refactorings, debugging, run-time instrumentation, and so on—that require syntactic, static, and dynamic reasoning about a program as it is being developed. The problem is that when the program being edited is erroneous, these semantic services can become degraded or unavailable [Omar et al. 2017b]. These gaps in service are not always transient. For example, a change to a type definition might result in type errors at dozens of use sites in a large program, which might take hours to resolve, all without the full aid of these services. An error *anywhere* renders the program formally meaningless *everywhere*.

This gap problem has prompted considerable interest in (1) **error localization**: mechanisms for identifying the location(s) in a program that explain an error, and (2) **error recovery**: mechanisms that allow the system to optimistically recover from a localized error and continue on to locate other errors and provide downstream semantic services, ideally at every location in the program and with minimal gaps in service. Essentially all widely-used programming systems have some support for error localization, e.g. in compiler or run-time error messages. Developers are known to attend to reported error locations when debugging errors [Joosten et al. 1993]. Many systems also attempt recovery in certain situations, e.g. by ignoring the erroneous definition and attempting to continue on. However, error localization and recovery mechanisms have developed idiosyncratically, in part as folklore amongst language and tool implementors. Different compilers or language servers [Barros et al. 2022; Bour et al. 2018], even for the same language, localize and recover from type errors in different ways, with little in the way of unifying theory of the sort that grounds the design of modern type systems themselves.

The Hazel project (https://hazel.org/) has sought to address this problem by developing principled language-theoretic foundations for syntax, type, and run-time error localization and recovery. In particular, the Hazel project has developed *total* error localization and recovery methods: there are no malformed or meaningless editor states in Hazel. Instead, Hazel automatically inserts empty and non-empty holes into the program to ensure that each and every editor state is well-structured [Moon et al. 2022, 2023] and statically [Omar et al. 2017a] and dynamically [Omar et al. 2019] meaningful. In short, Hazel is the first *totally live* typed general-purpose programming environment.

The proposed talk will review the underlying theoretical developments and include live demos of the Hazel programming environment, which scales up these theoretical foundations to a full-scale live functional programming environment, with support for features like pattern matching (with pattern holes [Yuan et al. 2023]) and modules. Hazel has been used to introduce functional programming to students in an undergraduate programming languages course at the University of Michigan, and the talk will also demonstrate the educational technology underlying this effort [Potter et al. 2022; Potter and Omar 2020] as well as the results of our initial assessments of its usability and suitability in an educational setting. Finally, we will discuss ongoing and future efforts, with a particular emphasis on using these underlying semantic mechanisms to develop useful editor services (including semantically contextualized programming assistants that integrate a variety of techniques, including generative AI models [Blinn et al. 2022]) and evaluate them with humans.

Authors' addresses: Cyrus Omar, comar@umich.edu, University of Michigan, USA; Andrew Blinn, blinnand@umich.edu, University of Michigan, USA; David Moon, dmoo@umich.edu, University of Michigan, USA.

REFERENCES

Djonathan Barros, Sven Peldszus, Wesley KG Assunção, and Thorsten Berger. 2022. Editing support for software languages: implementation practices in language server protocols. In *Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems*. 232–243.

Andrew Blinn, David Moon, Eric Griffis, and Cyrus Omar. 2022. An Integrative Human-Centered Architecture for Interactive Programming Assistants. In *2022 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, 1–5.

Frédéric Bour, Thomas Refis, and Gabriel Scherer. 2018. Merlin: a language server for OCaml (experience report). *Proceedings of the ACM on Programming Languages* 2, ICFP (2018), 1–15.

Stef Joosten, Klaas van den Berg, and Gerrit van Der Hoeven. 1993. Teaching Functional Programming to First-Year Students. *J. Funct. Program.* 3, 1 (1993), 49–65. https://doi.org/10.1017/S0956796800000599

David Moon, Andrew Blinn, and Cyrus Omar. 2022. tylr: a tiny tile-based structure editor. In *Proceedings of the 7th ACM SIGPLAN International Workshop on Type-Driven Development*. 28–37.

David Moon, Andrew Blinn, and Cyrus Omar. 2023. Gradual Structure Editing with Obligations. In *To appear, IEEE Symposium on Visual Languages and Human-Centric Computing, VL/HCC 2023*.

Cyrus Omar, Ian Voysey, Ravi Chugh, and Matthew A. Hammer. 2019. Live Functional Programming with Typed Holes. *Proc. ACM Program. Lang.* 3, POPL, Article 14 (Jan. 2019), 32 pages. https://doi.org/10.1145/3290327

Cyrus Omar, Ian Voysey, Michael Hilton, Jonathan Aldrich, and Matthew A. Hammer. 2017a. Hazelnut: A Bidirectionally Typed Structure Editor Calculus. In *ACM SIGPLAN Symposium on Principles of Programming Languages (POPL)*.

Cyrus Omar, Ian Voysey, Michael Hilton, Joshua Sunshine, Claire Le Goues , Jonathan Aldrich, and Matthew A. Hammer. 2017b. Toward Semantic Foundations for Program Editors. In *Summit on Advances in Programming Languages (SNAPL)*.

Hannah Potter, Ardi Madadi, René Just, and Cyrus Omar. 2022. Contextualized Programming Language Documentation. In *Proceedings of the 2022 ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software*. 1–15.

Hannah Potter and Cyrus Omar. 2020. Hazel Tutor: Guiding Novices Through Type-Driven Development Strategies. *Human Aspects of Types and Reasoning Assistants (HATRA)* (2020).

Yongwei Yuan, Scott Guest, Eric Griffis, Hannah Potter, David Moon, and Cyrus Omar. 2023. Live Pattern Matching with Typed Holes. *Proc. ACM Program. Lang.* 7, OOPSLA1, Article 96 (apr 2023), 27 pages. https://doi.org/10.1145/3586048