ABSTRACT

Hazel is a live programming environment with typed holes that serves as a reference implementation of the Hazelnut Live dynamic semantics [1] and the Hazelnut static semantics [2], both of which tackle the "gap problem." This work attempts to further develop the Hazelnut Live dynamic semantics by implementing the environment model of evaluation (as opposed to the current substitution model) and memoizing several evaluation-related operations to improve performance. Additionally, we provide an implementation-level description and a reference implementation of the fill-and-resume (FAR) performance optimization proposed in Hazelnut Live. We produce a metatheory and reference implementation of the proposed changes. Our implementation is benchmarked against the existing Hazel implementation to show that the results match expectations, although there is room for future improvement with the development with memoization. Finally, we discuss some useful theoretical generalizations that result from this work.