HyperAST: Enabling Efficient Analysis of Software Histories at Scale

Quentin Le Dilavrec*[1]

[1]Institut de Recherche en Informatique et Systèmes Aléatoires (IRISA) – Université de Rennes, Institut National des Sciences Appliquées - Rennes, Université de Bretagne Sud, École normale supérieure - Rennes, Institut National de Recherche en Informatique et en Automatique, CentraleSupélec, Centre National de la Recherche Scientifique, IMT Atlantique – Avenue du général LeclercCampus de Beaulieu 35042 RENNES CEDEX, France

Résumé

Abstract Syntax Trees (ASTs) are widely used beyond compilers in many tools that measure and improve code quality, such as code analysis, bug detection, mining code metrics, refactoring. With the advent of fast software evolution and multistage releases, the temporal analysis of an AST history is becoming useful to understand and maintain code.

However, jointly analyzing thousands versions of ASTs independently faces scalability issues, mostly combinatorial, both in terms of memory and CPU usage. In this paper, we propose a novel type of AST, called HyperAST, that enables efficient temporal code analysis on a given software history by: 1/ leveraging code redundancy through space (between code elements) and time (between versions); 2/ reusing intermediate computation results. We show how the HyperAST can be built incrementally on a set of commits to capture all multiple ASTs at once in an optimized way. We evaluated the HyperAST on a curated list of large software projects. Compared to Spoon, a state-of-the-art technique, we observed that the HyperAST outperforms it with an order-of-magnitude difference from $\times$ 6 up to $\times$ 8076 in CPU construction time and from $\times$ 12 up to $\times$ 1159 in memory footprint. While the HyperAST requires up to 2h 22min and 7.2GB for the biggest project, Spoon requires up to 93h and 31min and 2.2TB. The gains in construction time varied from to and the gains in memory footprint varied from to . We further compared the task of finding references of declarations with the HyperAST and Spoon. We observed on average precision and recall without a significant difference in search time.

*Intervenant