Live Feedback through Incremental Program Analysis (Keynote)

Sebastian Erdweg
Johannes Gutenberg University Mainz

ABSTRACT Static analyses are vital in modern software development. Static analyses are best known for detecting errors at compile time and thus helping developers to write correct code. That's an honorable quest and not for nothing static analysis is sometimes referred to as lightweight verification. But static analysis has a second quest of equal importance: to aid program understanding. Development tools such as IDEs use information computed by static analyses to explain the program to the developer (variable v has type int, variables v1 and v2 point to the same object), to provide exploration functionality (jump to declaration, find call sites), to guide developers (uninitialized read, dead code), and to provide sophisticated edit actions (refactorings). IDEs must provide all these features without interrupting the workflow of the developer. In particular, IDEs must react quickly to code changes. That is, the underlying static analyses need to be incremental.

We present IncA, a language-independent framework for incremental static analysis that does not compromise on precision. IncA provides a DSL for the definition of static analyses and incrementalizes them automatically. We explain how the IncA compiler translates analysis specifications into Datalog-style rules and how the IncA runtime solves these rules incrementally. We particularly discuss how IncA incrementalizes fixpoint computations, which are ubiquitous in data-flow analysis. IncA has been used to incrementalize control flow and points-to analysis for C, string analysis and strong points-to analysis for Java, and type analysis for Rust.

JOT reference format:
Sebastian Erdweg. *Live Feedback through Incremental Program Analysis (Keynote)*. Journal of Object Technology. Vol. 25, No. 1, 2026. Licensed under Attribution 4.0 International (CC BY 4.0)
http://dx.doi.org/10.5381/jot.2026.25.1.a6