Benchmarking Partial Evaluation in Truffle

Truffle [1] is a framework which allows implementing programming languages on top of the GraalVM. Language implementers write an abstract syntax tree (AST) interpreter which can be registered in the Truffle ecosystem to be part of the polyglot GraalVM world.

One primary goal of Truffle is to deliver automatic near-native performance for AST interpreters by performing partial evaluation based on the 1. Futamura projection [2]. This is a process of specializing an interpreter with compilation-constant source code to produce an executable. Truffle's implementation of partial evaluation is compile-time intensive, because it involves parsing the bytecode of corresponding AST nodes and inlining them into the root compilation unit.

This presentation shows a framework which has been developed for benchmarking the process of partial evaluation in Truffle, as well as corresponding benchmarking results. Finally, it gives an outline for partial evaluation based on the 2. Futamura projection [2], which is going to be investigated and implemented for Truffle in a future PhD studies. This would allow specializing Truffle‘s partial evaluator itself with respect to a given AST interpreter. The result would be a partial evaluator tailored to a specific language, i.e, the resulting partial evaluator could carry out the compile-time intensive task of the 1. Futamura projection for any source program of the given language, and it could do this faster than the previous one.

[1] Thomas Würthinger, Christian Wimmer, Andreas Wöß, Lukas Stadler, Gilles Duboscq, Christian Humer, Gregor Richards, Doug Simon, and Mario Wolczko. 2013. One VM to rule them all. In Proceedings of the 2013 ACM international symposium on New ideas, new paradigms, and reflections on programming & software (Onward! 2013). ACM, New York, NY, USA, 187-204. DOI=http://dx.doi.org/10.1145/2509578.2509581

[2] Yoshihiko Futamura. 1999. Partial Evaluation of Computation Process – An Approach to a Compiler-Compiler. Higher-Order and Symbolic Computation, 12(4), 381-391. DOI=https://doi.org/10.1023/A:1010095604496

Time slot: 30 minutes