Basic Nanopass for RPython

Corbin Simpson
Matador Cloud LLC
Portland, Oregon, USA
corbin@matador.cloud

ABSTRACT

The nanopass style of compiler design eases the task of maintaining data-transformation passes in compilers, at the cost of requiring a metaprogramming framework which aids in the generation of the boilerplate for those passes. We present a Python 2 module for generating a nanopass compiler pipeline for the RPython toolchain. Our approach uses standard Python metaprogramming tactics to generate boilerplate RPython classes. The generated passes use features of the RPython type system to enforce a modicum of correctness, preventing certain common programmer errors. Passes also have utility methods for raising error contexts to the end user and handling source span information. Our module is about 200 lines of code and is used in the Typhon compiler to optimize and interpret the Monte programming language.

KEYWORDS

Monte, RPython, nanopass

ACM Reference Format:
Corbin Simpson. 2018. Basic Nanopass for RPython. In *Proceedings of MoreVMs 2018*. ACM, New York, NY, USA, 1 page. https://doi.org/10.1145/nnnnnnn.nnnnnnn

MoreVMs 2018, April 2018, Nice, France

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM.
https://doi.org/10.1145/nnnnnnn.nnnnnnn