REFERENCES

[1] [n. d.]. Typhon A virtual machine for Monte. https://github.com/monte-language/typhon/. ([n. d.]). Accessed: March 4, 2018.

[2] [n. d.]. Typhon typhon/nanopass.py. https://github.com/monte-language/typhon/blob/master/typhon/nanopass.py. ([n. d.]). Accessed: March 4, 2018.

[3] Davide Ancona, Massimo Ancona, Antonio Cuni, and Nicholas D. Matsakis. 2007. RPython: A Step Towards Reconciling Dynamically and Statically Typed OO Languages. In *Proceedings of the 2007 Symposium on Dynamic Languages (DLS '07)*. ACM, New York, NY, USA, 53–64. https://doi.org/10.1145/1297081.1297091

[4] Andy Keep. 2012. *A Nanopass Framework For Commercial Compiler Development.* Ph.D. Dissertation. Indiana University.

[5] Corbin Simpson. 2017. Monte: A Spiritual Successor to E. https://github.com/monte-language/monte-talks/blob/master/corbin-ocap17-monte.pdf. (2017).

Basic Nanopass for RPython

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

ABSTRACT

The nanopass [4] style of compiler design eases the task of maintaining data-transformation passes in compilers. However, nanopass compilers require a metaprogramming framework which aids in the generation of the boilerplate for those passes. The original nanopass toolchain is a Racket module of over 2000 lines of code containing many Racket-specific features.

We present a Python 2 module for generating a nanopass compiler pipeline for the RPython toolchain [3]. Our approach uses standard Python metaprogramming tactics to generate boilerplate RPython classes. Users of our module write standard RPython, inheriting from our classes and providing plain visitor methods.

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. We developed and use our nanopass module in the Typhon [1] compiler to optimize and interpret the Monte programming language [5]. Our source code is publically available under a Free Software license at [2].

KEYWORDS

Monte, RPython, nanopass

ACM Reference Format:
Corbin Simpson. 2018. Basic Nanopass for RPython. In *Proceedings of MoreVMs 2018.* ACM, New York, NY, USA, 2 pages. 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