Toward Virtual Machine Adaption Rather than Reimplementation

Adapting SOMns for Grace

Richard Roberts
Victoria University of Wellington
richard.andrew.roberts@gmail.com

Stefan Marr
Johannes Kepler University Linz
stefan.marr@jku.at

Michael Homer
James Noble
Victoria University of Wellington
mwh@ecs.vuw.ac.nz
kjx@ecs.vuw.ac.nz

Abstract

We adapt SOMns, a Truffle-based interpreter for Newspeak, to the Grace programming language. We highlight differences between the semantics of these languages and offer preliminary results showing that adaption is possible while retaining performance. The similarities between the languages promote the potential for adaption and code sharing between implementations. Through experimentation we intend to explore how the design of the tailored implementation; the flexibility of the underlying framework; and similarities between languages affect adaptability, and by doing so hope to realize a set of mechanisms that can be easily extended to create optimized virtual machines for object-orientated languages.

1. Introduction

Language implementation frameworks enable us to build simple interpreters with good performance. With Truffle (Würthinger et al. 2013), abstract-syntax tree interpreters are built on top of a host virtual machine that provides garbage collection and just-in-time compilation. With the RPython toolchain, a tailored virtual machine is generated based on a simple interpreter. The RPython toolchain also adds garbage collection and a tracing just-in-time compiler. Developers have already used these frameworks to successfully implement popular object-oriented programming languages (such as Python, Ruby, or JavaScript), with performance results as good as state of the art JIT-ing virtual machines (Marr and Ducasse 2015; Marr et al. 2016).

Despite the multitude of language implementations built using these frameworks, it appears that implementors tend to recreate similar components (e.g., basic control structures, abstract-syntax tree transformations, object storage) *on top of* the framework rather than reuse existing implementations; doing so to gain fine-grained control over later optimizations (at the cost of a significant amount of work).

Recently, the notion of developing a set of components that can be easily extended to create tailored virtual machines have gained traction. For example, Wößet al. (2014) has introduced an *Object Storage Model*, which has now been integrated into the Truffle framework; and Inostroza and Storm (2015) propose reusable components for languages with denotational semantics using *Object Algebras* (Oliveira and Cook 2012). From much the same perspective, we propose to explore creating new language implementations by extending existing ones built using language implementation frameworks. As an initial experiment we adapt *SOMns*, a Newspeak virtual machine, to Grace.

2. Background

Newspeak (Bracha et al. 2010) is a deeply nested object-oriented language designed to support libraries and family polymorphism (as in BETA and gBETA) (Lehrmann Madsen et al. 1993; Ernst 2001). Like Smalltalk, Newspeak is a pure object-oriented language, in that all computation proceeds via messages sent to objects. Additionally, Newspeak adopts Smalltalk's use of lambda-expressions ("blocks") to encode control structures.

Grace is also a deeply nested object-oriented language, designed to support education (Black et al. 2012). Grace mixes static and dynamic types (like Strongtalk or C♯); makes heavy use of lambda expressions (like Smalltalk or Ruby); and supports nested class definitions (like BETA).

Newspeak was a significant influence on Grace's design. Grace shares with Newspeak that static types generally do not affect a program's execution; that control structures are handled with lambda expressions; and that class definitions can be nested. A significant difference in Grace is that classes are defined in terms of objects and they can

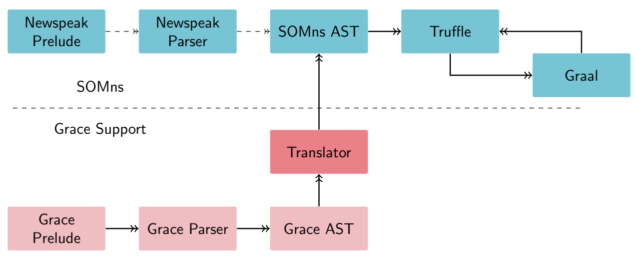

Figure 1. SOMns is built upon Truffle and Graal and implements an abstract-syntax tree for Newspeak. We parse programs written in Grace using an existing parser, and translate the generated AST into the SOMns AST.

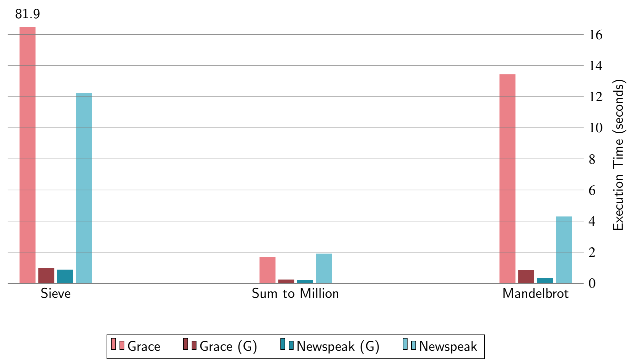

Figure 2. The average running times for three harnessed benchmarks (with (G) and without the Graal optimization layer). Each benchmark was implemented equivalently in both Grace and Newspeak.

be nested freely (inside other objects, classes, methods, or lambda expressions), whereas in Newspeak objects are defined in terms of classes and classes can only be nested inside other classes.

SOMns (Marr and Mössenböck 2015) is a high-performance virtual machine built using Truffle and the Graal JITing compiler (Würthinger et al. 2013) and is a *mostly* compliant implementation of the Newspeak specification.[1] Compared to Newspeak, it forgoes support for image-based development, many reflective features, and a foreign function interface. Instead, it is meant as a research platform for investigating concurrency models.

3. Current Status

For the initial proof of concept, we have implemented expression nodes, numerals and string nodes, as well as nodes to maintain lexical scoping for blocks and methods.

To date SOMns has been under development for 22 months, and is as fast as other JITing virtual machines. In contrast Grace has been in development for around five years, without consideration of performance thus far. Yet, in only three weeks, we have adapted a Truffle-based virtual machine to realize a new implementation that supports Grace's primitive operations; compound expressions; method and field declarations, and object instantiation. We do not yet support full inheritance nor type systems.

Our implementation adheres correctly to the expected semantics and Grace programs can be executed with comparable speed to Newspeak. As indicated by the benchmarks in Figure 2, the loss of performance due to the additional translation step is marginal. The implementation retains much of the optimizations that were carefully tailored for Newspeak but required only a trivial amount of work by comparison.

[1] http://newspeaklanguage.org/spec/newspeak-spec.pdf

4. Conclusion

For a fully compliant implementation, we need to address the inheritance model and also differences in lexical scoping between Grace (classes can be nested anywhere) and Newspeak (classes can only be nested inside of other classes). As Truffle does not support higher-level abstract-syntax tree manipulation or lexical scoping, we are left with the choice of reimplementing new abstract-syntax tree nodes using Truffle (work that requires careful planning) or otherwise changing the SOMns abstract-syntax tree nodes (which may conflict with Newspeak semantics).

While language implementation frameworks provide significant advantages, we believe that a set of simple-but-extensible mechanisms for such functionalities as higher-level abstract-syntax tree manipulation, lexical scoping, and storage strategies would encourage adaption over reimplementation. Through continuing to adapt SOMns for Grace we will pursue the development of such mechanisms. If successful, we can begin to reconceptualize the process of language implementation, where developers may choose to reuse small but valid virtual machines that require only minor adaption to implement semantically correct and fast interpreters for their languages.

Acknowledgments

This research was partially supported by a James Cook Fellowship and a Royal Society of New Zealand Marsden Grant. Stefan Marr was funded by a grant of the Austrian Science Fund (FWF), project number I2491-N31.

References

A. P. Black, K. B. Bruce, M. Homer, and J. Noble. Grace: the absence of (inessential) difficulty. Onward!, 2012.

G. Bracha, P. von der Ahé, V. Bykov, Y. Kashai, W. Maddox, and E. Miranda. Modules as objects in newspeak. ECOOP, 2010.

E. Ernst. Family polymorphism. In *ECOOP*, 2001.

[1] http://gracelang.org/documents/grace-spec-0.7.0.pdf

P. Inostroza and T. v. d. Storm. Modular interpreters for the masses: Implicit context propagation using object algebras. *GPCE*, 2015.

O. Lehrmann Madsen, B. Møller-Pedersen, and K. Nygaard. *Object-Oriented Programming in the BETA Programming Language*. Addison-Wesley, 1993.

S. Marr and S. Ducasse. Tracing vs. partial evaluation: Comparing meta-compilation approaches for self-optimizing interpreters. OOPSLA, 2015.

S. Marr and H. Mössenböck. Optimizing communicating event-loop languages with truffle, AGERE, 2015.

S. Marr, B. Daloze, and H. Mössenböck. Cross-Language Compiler Benchmarking—Are We Fast Yet? DLS, 2016.

B. C. d. S. Oliveira and W. R. Cook. Extensibility for the masses: Practical extensibility with object algebras. ECOOP, 2012.

A. Wöß, C. Wirth, D. Bonetta, C. Seaton, C. Humer, and H. Mössenböck. An object storage model for the truffle language implementation framework. PPPJ '14, 2014.

T. Würthinger, C. Wimmer, A. Wöß, L. Stadler, G. Duboscq, C. Humer, G. Richards, D. Simon, and M. Wolczko. One VM to Rule Them All. Onward!, 2013.