Yin-Yang: Concealing the Deep Embedding of DSLs

Vojin Jovanovic and Amir Shaikhha

Ecole Polythechnique Federale de Lausanne, EPFL
{first}.{last}@epfl.ch

Abstract. Deeply embedded domain-specific languages (EDSLs) inherently compromise programmer experience for improved program performance. Shallow EDSLs complement them by trading program performance for good programmer experience. We present Yin-Yang, a framework for DSL embedding that uses Scala reflection to reliably translate shallow EDSL programs to the corresponding deep EDSL programs. The translation allows program prototyping and debugging in the user friendly shallow embedding, while the corresponding deep embedding is used in production—where performance is important. The reliability of the translation completely conceals the deep embedding from the DSL user. For the DSL author, Yin-Yang generates the deep DSL embeddings from their shallow counterparts by reusing the core translation. This obviates the need for code duplication and assures that the implementations of the two embeddings are always synchronised.

Keywords: Embedded Domain-Specific Languages, Macros, Deep Embedding, Shallow Embedding, Compile-Time Meta-Programming

1 Introduction

External Domain-specific languages (DSLs) are languages with a custom compiler specialized to a particular application domain. Knowledge about the domain and language specialization give the compiler possibilities for optimization that do not exist in general purpose languages. The restricted language makes it easy for *DSL users* to learn the language while the optimizations can yield execution times close to hand-optimized programs [3].

Development of external DSLs includes building a parser and type checker as well as a large tool-chain (i.e., debuggers, IDEs, and documentation tools). An appealing alternative are *embedded DSLs* (EDSLs) [2] that reuse the parser, type checker and the tool-chain of the general-purpose *host language* to minimize required development effort. We can classify DSL embeddings into two categories:

– *Shallow embeddings* where values in the embedded language are *directly* represented by the values in the host language. Special category of shallow EDSLs are *direct EDSLs* where language constructs and term types are the same as in the host language.

– *Deep embeddings* where values in the embedded language are symbolically (with data structures) represented in the host language.

On the EDSL author side, Yin-Yang can automatically generate deep EDSL embeddings from their shallow counterparts. This leads to reliability by construction, since the deep and shallow embeddings are provably equivalent. Generating deep embeddings shields the EDSL author from complicated compiler internals and shifts focus on domain-specific optimizations and error-reporting [1].

On the EDSL author side, Yin-Yang can automatically generate deep EDSL embeddings from their shallow counterparts. This leads to reliability by construction, since the deep and shallow embeddings are provably equivalent. Generating deep embeddings shields the EDSL author from complicated compiler internals and shifts focus on domain-specific optimizations and error-reporting [1].

On the EDSL author side, Yin-Yang can automatically generate deep EDSL embeddings from their shallow counterparts. This leads to reliability by construction, since the deep and shallow embeddings are provably equivalent. Generating deep embeddings shields the EDSL author from complicated compiler internals and shifts focus on domain-specific optimizations and error-reporting [1].

2 Yin-Yang Demonstration

On the EDSL author side, Yin-Yang can automatically generate deep EDSL embeddings from their shallow counterparts. This leads to reliability by construction, since the deep and shallow embeddings are provably equivalent. Generating deep embeddings shields the EDSL author from complicated compiler internals and shifts focus on domain-specific optimizations and error-reporting [1].

On the EDSL author side, Yin-Yang can automatically generate deep EDSL embeddings from their shallow counterparts. This leads to reliability by construction, since the deep and shallow embeddings are provably equivalent. Generating deep embeddings shields the EDSL author from complicated compiler internals and shifts focus on domain-specific optimizations and error-reporting [1].

On the EDSL author side, Yin-Yang can automatically generate deep EDSL embeddings from their shallow counterparts. This leads to reliability by construction, since the deep and shallow embeddings are provably equivalent. Generating deep embeddings shields the EDSL author from complicated compiler internals and shifts focus on domain-specific optimizations and error-reporting [1].

On the EDSL author side, Yin-Yang can automatically generate deep EDSL embeddings from their shallow counterparts. This leads to reliability by construction, since the deep and shallow embeddings are provably equivalent. Generating deep embeddings shields the EDSL author from complicated compiler internals and shifts focus on domain-specific optimizations and error-reporting [1].

On the EDSL author side, Yin-Yang can automatically generate deep EDSL embeddings from their shallow counterparts. This leads to reliability by construction, since the deep and shallow embeddings are provably equivalent. Generating deep embeddings shields the EDSL author from complicated compiler internals and shifts focus on domain-specific optimizations and error-reporting [1].

References

1. Slick, http://slick.typesafe.com/.
2. Paul Hudak. Building domain-specific embedded languages. *ACM Comput. Surv.*, 28(4es):196, 1996.
3. Tiark Rompf, Arvind K. Sujeeth, Nada Amin, Kevin J. Brown, Vojin Jovanovic, HyoukJoong Lee, Manohar Jonnalagedda, Kunle Olukotun, and Martin Odersky. Optimizing data structures in high-level programs: New directions for extensible compilers based on staging. In *Proceedings of the 40th annual ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, page 497–510, 2013.