Pragmatic Extensions for Language Embedding Using Load-Time Metaprogramming

論 文 の 内 容 の 要 旨

論文題目　Pragmatic Extensions for Language Embedding Using Load-Time Metaprogramming
（言語埋込みのためのロード時メタプログラミングを用いた実用的拡張）

氏　名　シェア　マクシミリアン　パスカル

This dissertation details my research on the improvement of techniques for embedding domain-specific languages (DSLs) in a general-purpose host language. Standalone, dedicated DSLs provide a high-level abstraction to problem solving. Their embedding by using existing host-language constructs and language features not only reduces the initial implementation effort but also opens avenues to enrich traditional programming-library interfaces.

Historically this idea evolved from compile-time metaprogramming techniques such as syntactic macros. In modern high-level languages one can find a tendency of avoiding such so-called impure techniques. Instead, language constructs that have a fixed run-time behavior are preferred and often sufficient. This pure embedding is arguably easier for authors as well as (and maybe more importantly) users of embedded DSLs (EDSLs) due to the predictable, uniform behavior of the employed constructs or abstraction mechanisms.

Still, existing manual implementation approaches suffer from tradeoffs in terms of reliability, performance, and usability. The major cause for this is found in the low-level emulation of high-level embedded-language aspects. Ultimately, EDSLs are non-citizens in their general-purpose host-language, which leads to missed opportunities for eliminating the aforementioned tradeoffs.

My proposal is to raise the status of EDSLs to almost first-class citizens by means of meta-level, semi-linguistic support. It lies in the nature of doing so to tamper with the workings of the host language, essentially extending it. This is prone to lead to heavyweight designs unlikely to appeal to mainstream users.

Also, many of the strengths of language embedding may be weakened when interfering with the host-language front end. As an alternative, load-time metaprogramming offers a good compromise for realizing a pragmatic form of language extension in our context.

The initial milestone is to turn EDSLs into explicit entities with ownership of associated host-language constructs such as methods. I address this with the so-called implicit staging approach, which is rather abstract in nature. Based on a prototype framework and closer investigation I conclude that further restrictions and documentation mechanisms are necessary to render EDSLs more reliable and useful. This is explored with a concrete framework that, in addition to load-time metaprogramming, relies on Java's annotation feature to enable control and communication of staged-EDSL behavior.