Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science
ELSEVIER
Electronic Notes in Theoretical Computer Science 203 (2008) 21–36
www.elsevier.com/locate/entcs

Fusing a Transformation Language with an Open Compiler

Karl Trygve Kalleberg [1]

Department of Informatics, University of Bergen, P.O. Box 7800, N-5020 BERGEN, Norway

Eelco Visser [2]

Department of Software Technology, Faculty of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology, The Netherlands

Abstract

Program transformation systems provide powerful analysis and transformation frameworks as well as concise languages for language processing, but instantiating them for every subject language is an arduous task, most often resulting in half-completed frontends. Compilers provide mature frontends with robust parsers and type checkers, but solving language processing problems in general-purpose languages without transformation libraries is tedious. Reusing these frontends with existing transformation systems is therefore attractive. However, for this reuse to be optimal, the functional logic found in the frontend should be exposed to the transformation system – simple data serialization of the abstract syntax tree is not enough, since this fails to expose important compiler functionality, such as import graphs, symbol tables and the type checker.
In this paper, we introduce a novel and general technique for combining term-based transformation systems with existing language frontends. The technique is presented in the context of a scriptable analysis and transformation framework for Java built on top of the Eclipse Java compiler. The framework consists of an adapter automatically extracted from the abstract syntax tree of the compiler and an interpreter for the Stratego program transformation language. The adapter allows the Stratego interpreter to rewrite directly on the compiler AST. We illustrate the applicability of our system with scripts written in Stratego that perform framework and library-specific analyses and transformations.

Keywords: compiler scripting; strategic programming; program transformation

1 Introduction

Developing and maintaining frameworks and libraries is at the core of software development: all domain abstractions of software applications are invariably encoded into libraries of a given programming language. Maintenance of this code involves various language processing tools such as compilers, editors, source code navigators,

[1] Email: karltk@ii.uib.no
[2] Email: visser@acm.org

doi:10.1016/j.entcs.2008.03.042

documentation generators, style checkers and static analysis tools. Unfortunately, most of these tools only have a fixed repertoire of functionality which seldom covers all the needs of the developer of a given library or framework. Relatively few processing tools can quickly and easily be programmed, extended or adapted by the library developer. This often drives developers to implement many additional, text-based tools from scratch. A preferable solution would be for library developers to quickly write custom scripts in a suitable scripting language and thus implement analyses and transformations specific to their own code bases, such as style checking and library-specific optimizations. Domain-specific languages (DSLs) for program analysis and transformations are attractive candidates for expressing these scripts, since DSLs allow precise and concise formulations. However, the DSLs are rarely coupled with robust and mature parsers and type analyzers. Open compilers are also attractive because they provide solid parsers and type analyses, but implementing analyses and transformation in their general-purpose languages is often very time-consuming.

In this paper, we obtain the best of both worlds by combining Stratego, a DSL for program transformation and the open Eclipse Compiler for Java (ECJ), using a program object model (POM) adapter. The POM adapter welds together the Stratego runtime and the ECJ abstract syntax tree (AST) by translating Stratego rewriting operations on-the-fly to suitable method calls on the AST API. This obviates the need for data serialization. The technique can be applied to many tree-like APIs, and is reusable for other rewriting systems. Using the POM adapter, Stratego becomes a compiler scripting language, offering its powerful features for analysis and transformation such as pattern matching, rewrite rules, generic tree traversals, and a reusable library of generic transformation functions and data-flow analysis. This combination is a powerful platform for programming domain-specific analyses and transformations. We argue that the system can be wielded by advanced developers and framework providers because large and interesting classes of domain-specific analyses and transformations can be expressed by reusing the transformation libraries provided with Stratego.

The contributions of this paper include the fusing of a DSL for language processing with an open compiler without resorting to data serialization. This brings the analysis and transformation capabilities of modern compiler infrastructure into the hands of advanced developers through a convenient and feature-rich transformation language. The technique is reusable for other transformation languages. It may help make transformation tools and techniques practical and reusable both by compiler designers and by framework developers, since it directly integrates them with stable tools like the Java compiler – developers can write interesting classes of analyses and transformations easily and compiler designers can experiment with prototypes of analyses and transformations before committing to a final implementation. We validate the system's applicability through a series of examples taken from mature and well-designed applications and frameworks.

The remainder of this paper is organized as follows: In Sec. 2, we discuss the POM adapter and how it connects Stratego with ECJ. In Sec. 3, we show the