Automatic Programming Language Translation

Marco Trudel

Chair of Software Engineering, ETH Zurich, Switzerland

Abstract. Different programming languages have different advantages. Can we automatically have the best of multiple worlds? I will present two translators we developed to achieve that goal: (1) C2Eif; translates C source code to Eiffel source code, allowing to move legacy code to a modern object-oriented programming language. (2) J2Eif; translates Java source code to Eiffel source code, allowing to reuse the numerous libraries available for Java.

1 C2Eif

Can we reuse some of the huge code-base developed in C to take advantage of modern programming language features such as type safety, object-orientation, and contracts? I will present a source-to-source translation of C code into Eiffel, a modern object-oriented programming language, the applied object-oriented reengineering, and the supporting tool C2Eif[1]. The translation is completely automatic and supports the entire C language (ANSI, as well as many GNU C Compiler extensions, through CIL) as used in practice, including its usage of native system libraries and inlined assembly code. Our experiments show that C2Eif can handle C applications and libraries of significant size (such as `vim` and `libgsl`), as well as challenging benchmarks such as the GCC torture tests. The produced Eiffel code is functionally equivalent to the original C code, and takes advantage of some of Eiffel's features to produce safe and easy-to-debug translations.

2 J2Eif

Reusability is an important software engineering concept actively advocated for the last forty years. While reusability has been addressed for systems implemented using the same programming language, it does not usually handle interoperability with different programming languages. I will present a solution for the reuse of Java code within Eiffel programs based on a source-to-source translation from Java to Eiffel. The translation is implemented in the freely available tool J2Eif[2]; it provides Eiffel replacements for the components of the Java runtime environment, including Java Native Interface services and reflection mechanisms. Our experiments demonstrate the practical usability of the translation

[1] `http://se.inf.ethz.ch/research/c2eif`
[2] `http://se.inf.ethz.ch/research/j2eif`

scheme and its implementation, and record the performance slow-down compared to custom-made Eiffel applications: automatic translations of `java.util` data structures, `java.io` services, and SWT applications can be re-used as Eiffel programs, with the same functionalities as their original Java implementations.