Yann-Gaël Guéhéneuc

Thesis subject proposal

A multi-language compiler framework for Java[1] virtual machines

Technical summary

The goal for this thesis is to explore the feasibility of combining different Java-compatible languages, i.e. languages that can be translated into Java bytecodes and be executed on a standard Java virtual machine.

Languages such as Smalltalk, C/C++, Visual Basic, SQL, NetRexx, Inferno, Scheme, Prolog, Tcl … (and/or subsets of them) would be studied in details to identify their (syntactic and semantic) similarities with the Java object model and the Java execution model.

From this study, a general compiler framework would be defined allowing for a reduction in implementation work when a new compilers for other languages is being developed. The definition of this framework will lead to questions such as how to deal with constructs that do not exist in Java, performance of the derived compilers, and other issues.
Another interesting problem would be to allow the user to *not specify* which language is being used and to let the compiler decide how to actually compile the input. This study would determine under which criteria languages may be combined and with what granularity, i.e. class, methods, statements (or similar constructs). The result of this study would be a set of generic well-defined "compiler components" (code generator, AST nodes, …) which, once assembled together (and possibly specialized), form a compiler. Such a compiler would discover by itself what language the source belongs to and how to process it. This research project can feed back to the MLS project (a Multi-Language Support environment realized inside OTI), in which a "dual" compiler, compiling either Java or Smalltalk, has been defined.

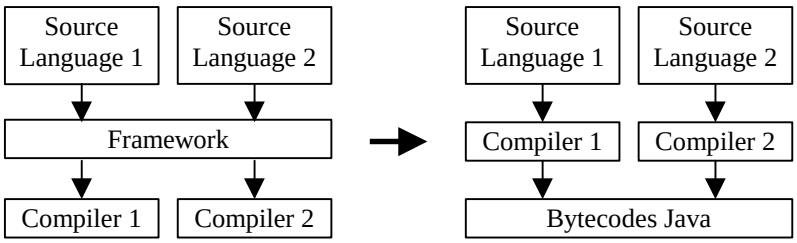
One problem is to define the criteria used to choose one language or another. When the syntax of the languages are different enough (e.g. Java and Scheme), the choice can be done at an early stage (at parse-time). However when the languages are very similar (Java, C, C++), other mechanisms should be studied (definition and calculus of the *distance* between various programming languages). The level of abstraction of the so-called "compiler components" should also be examined to allow a sophisticated abstraction and a wide range of possibilities. The composition of multi-language compilers may also be seen as the practice of developing more generic compilers. Finally, all this will lead to the idea of mixing syntactic and semantic constructs belonging to different languages inside one unique source. How could the compiler react to such situations ?

At last, an interesting feature of such a framework would be the possibility to plug-in tools that may be defined independently of the source languages being used. Automatic error correction, code assistance (display of the completion that may be applied on an piece of text at a particular position in the source code), repository management and name resolution, text search, and support for debugging need to be considered.

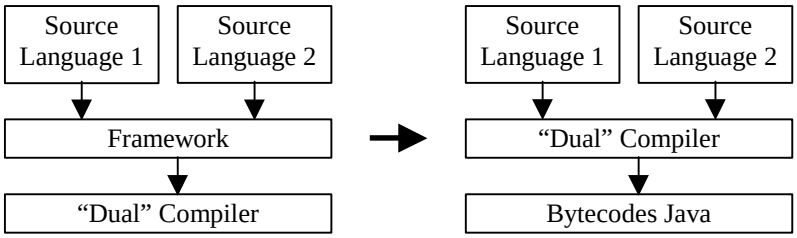
[1] Java is a registered trademark of Sun Microsystems, Inc.

This subject may be decomposed in three main steps.

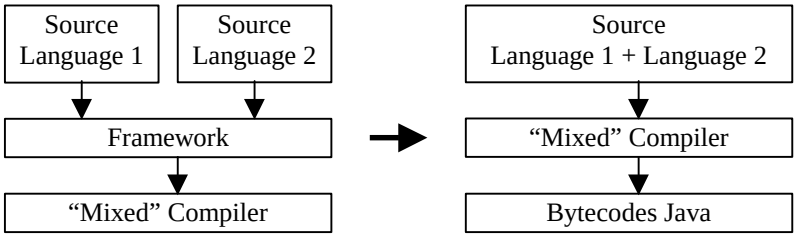
First step:



Second step:



Third step:



Synopsis

Start and duration:

Start : April 1999
Duration: 3 years

Location:

I would like to share my time between Ottawa and Amsterdam since I will work with Chris Laffra and the compiler team is mainly located in Ottawa, but I would mainly stay in Amsterdam.

Requirements:

I need to register to a university (either in Ottawa, Nantes or Amsterdam) as phD student.

Salary:

1,600 EUR (+ health care if needed and university registration)