TJHSST Computer Systems Lab Senior Research Project Proposal C to Java Translator 2007-2008

Jamie McAtamney

October 31, 2007

Abstract

The purpose of this project is to create a fully functional, fast, and efficient translator to convert programs in C code to Java code. While C is a language which is broad in scope and fast in execution, its implementation is platform-specific; this can be a disadvantage for programs expected to be used on multiple platforms and operating systems or run over the Internet. In addition, while C gives a programmer greater control over computer architecture and program structure, a higher-level language such as Java can be easier for amateurs to learn and understand; also, Java implements variable types such as Strings which make programming easier even for experienced programmers, as some algorithms and data structures are already defined for them. Programming language translation is a sub-field of computational linguistics that has not been widely researched, and while there are a few companies or private individuals who have written programs to translate between a limited number of languages, the field remains largely unexplored. The number of companies or private individuals who would like or require the translation of older programs into a different language vastly outnumbers the translation algorithms available, and of these groups, those who require old C programs translated into Java are particularly numerous while C-to-Java translation algorithms are particularly rare.

1 Introduction

1.1 Scope of Study

Conversion of basic syntax, input/output methods, and control structures will be included, and translation of included C modules to imported Java modules will be included if time permits. While little research has been done pertaining to programming language translation, more practical information is needed than theoretical information, namely, the different syntax structures, grammar, and abilities of C and Java. The area of overlap between C and Java is significant, but there are still many differences between the two languages which would make a single perfectly-comprehensive translator almost impossible; for this reason, translation will begin with more basic concerns, such as method headers, input/output routines, and variables, before gradually moving on to more complex tasks, concluding eventually with translation of C “include” module methods to their Java “import” equivalents.

As a comprehensive translator could be difficult to achieve in a single year, development will progress in stages. The first stage will deal with the basics of each language: input/output, flow control (switch statements, for loops, etc.), variable types and arrays, and method headers, not necessarily in that order. The second stage will move onto translation of structs into classes, as well as minor topics that were not covered in the first stage but that are extremely syntactically similar in both languages, such as bit shifting and packages such as ¡math.h¿. The final stage would involve translating as much of the other packages as time permits, as well as other differences that might have been missed in prior stages. Stage 1 is expected to be finished quickly in relation to the others, with Stages 2 and 3 each increasing in time required.

1.2 Expected results

In the process of translating between C and Java, the similarities and differences between the two will become more apparent, which will both assist later attempts at translation and highlight the strengths of each language. Insights gained from this translation process may subsequently be applied to translation between other languages, eventually lessening the barriers between operating systems and computer architecture and allowing the best language for a given task to be used regardless of system requirements.

With this project, I would like to learn what the differences between C and Java are, aside from the obvious fact that C is a lower-level language than Java, and why the differences are what they are. I would also like to add to general knowledge in the area of programming language translation, as it is neglected by the majority of computer scientists, and fill in the gaps in that knowledge with my own experiences with the project. Finally, I enjoy a challenge, and I would like to attempt a task which the majority of programmers believe is too complex.

1.3 Type of research

The general area of research into which my project is classified is "use-inspired basic research," research that attempts to pursue fundamental understanding and which is motivated by a question of practical application.

2 Background and Review of Current Research

Little research has been done in the area of programming language translation, as it is widely regarded as too difficult to deal with different syntax and different included models, but some headway has been made into the field. In particular, a company named Jazillion provides a pay-per-program approach to C-to-Java translation and claims to generate "natural" code similar to that a human programmer would write. Even this program, however, does not guarantee perfect translation and requires significant client involvement to customize the algorithm to the client's needs; language translation is still a field on the frontiers of computer science. There are also many free (but more limited) programs on the Internet which can handle particular aspects of a program; for example, one might be able to translate the C printf() functions to Java's System.out.print(), while another might be able to translate method headers.

No research papers or projects specifically written about the translation of C to Java (or even Java to C) have been made available on the Internet. There are a few papers dealing with translating COBOL to C and a few more on translating COBOL to Java, which could provide a few pointers; however, all of the papers deal with translation by hand, so the vast majority are inapplicable to automated translation processes. Jazillion's translator would probably provide the most inspiration for my project, as the company has made their design philosophy and goals available on its website, but my project will differ in terms of the language used for the translator as well as the design scheme, as perfection will necessarily be sacrificed for modularity. There is one paper dealing with automated language translation in general, however; "The Realities of Language Conversion" by Terekhov and Verhoef examines the difficulties inherent in any automated translation processes and how to fix them, which will be very useful for my project

3 Procedures and Methodology

The translation program itself will be written in Java; though I have programmed in C more recently than in Java, and C runs more quickly than Java, which can be a consideration when attempting translation of a long or complex program, Java's previously-implemented tokenizer and String structures are necessary for the translator to function optimally. I have a repertoire of C programs from Computer Architecture last year, with varied methods, constructs, and structures among them, which should make a diverse and strong set of test cases for the translator. I will also search for programs incorporating structures and algorithms which my programs do not incorporate, though it will not be possible to incorporate all possible permutations of C code for the translator. In terms of a visual display for my project, aside from generating a percentage of programs correctly translated, it does not lend itself to numerical judgment of progress, instead relying on a comparison of code to determine accuracy.

Owing to the nature of language translation, testing will necessarily involve more methodical than stochastic testing; though a given program may contain seemingly random inputs, the test cases as a whole will contain a broad and comprehensive spectrum of test cases that will test every part of the program. Testing a translation program is simple–if the program translates the cases for which you have created algorithms and passes over untranslatable sections without changing them or causing errors, the program works as intended. More generally, if the translated portions of the program compile and run in the same way as the original program, the translator is a success. A binary system such as this–either an approach works or it doesn't–makes error checking and correction relatively simple; as well, because the exact output for a given input is known precisely (in other words, the process rather than the destination is the object), there is no complex model necessary to deduce or learn the correct output and therefore performance validation will not be an issue. As my project will involve development in stages, it will require a large degree of modularity so as to permit the addition of new cases as translations are implemented. Otherwise, my project is fairly structure-independent, not requiring a single approach or algorithm to be successful.

4 Expected Results

By the end of the year, I expect a functional and efficient, if not completely comprehensive, translator able to convert the majority of C code into Java with high accuracy. When graphical representations of progress are required, side-by-side comparisons of code will by most useful to judge progress, as well as an analysis of the program's accuracy over the course of the year. The principles of my project should be widely applicable to translation between any two languages, as well as any researchers interested in computational linguistics. While the contributions of my project to the public at large might not be particularly memorable, I would hope it would be significant.