

Automatic C to O-O Translation with C2Eiffel

Marco Trudel · Carlo A. Furia · Martin Nordio
Chair of Software Engineering, ETH Zurich, Switzerland
Email: firstname.lastname@inf.ethz.ch

Abstract—C2Eiffel is a fully automatic source-to-source translator of C applications into the Eiffel object-oriented programming language. C2Eiffel supports the complete C language, including function pointers, unrestricted pointer arithmetic and jumps, arbitrary native libraries, and inlined assembly code. It produces readable Eiffel code that behaves as the source C application; it takes advantage of some of Eiffel’s object-oriented features to produce translations that are easy to maintain and debug, and often even safer than their sources thanks to stricter correctness checks introduced automatically. Experiments show that C2Eiffel handles C applications of significant size (such as `vim` and `libgsl`); it is a fully automatic tool suitable to reuse C code within a high-level object-oriented programming language.

I. REUSING C CODE WITH A CLICK

This paper presents C2Eiffel (C2Eif for short), a completely automatic tool that translates C source code into the Eiffel object-oriented programming language. With C2Eif, you can reuse C applications in a modern environment, where they can be seamlessly integrated with other native Eiffel code that fully takes advantage of the object-oriented paradigm and of other high-level language features such as contracts and static type safety. Since C2Eif translates source-to-source, the translated C code is not merely made available within a host environment—as is the case with integration solutions based on foreign language interfaces—but becomes a native Eiffel implementation, which developers can refactor and extend as the overall application evolves and undergoes maintenance. The translation provided by C2Eif may even single-handedly *improve* the safety and debuggability of the C implementation, because it automatically introduces safety checks for issues such as out-of-bound array access and null-pointer dereferencing. The checks make understanding and debugging translated applications considerably easier, because faults manifest themselves closer to their source location, and because certain buffer overflow errors are harder to replicate and exploit within the tight Eiffel runtime.

Even if you are not developing applications in Eiffel, the same ideas used in C2Eif underlie the development of fully automatic translators of C into other full-fledged modern object-oriented programming languages such as Java and C#. In fact, the chasm existing between the C and Eiffel languages ensures that the translation of one into the other is not a simple transliteration (as it would have been possible, for example, with C++) but has to provide output that is germane to the object-oriented paradigm.

C2Eif is open-source and available for download at <http://se.inf.ethz.ch/research/c2eif>

This tool demo paper describes the overall architecture of

C2Eif (Section II), presents the main results of an extensive experimental evaluation (Section III), and compares C2Eif against other similar tools (Section IV). This short paper only describes the tool architecture and usage; a detailed presentation of the translation scheme including examples is available in a companion paper [1].

II. OVERVIEW AND ARCHITECTURE

C2Eif is a compiler with a graphical user interface and a command-line interface that translates C programs to Eiffel. The translation produces a complete Eiffel application that is functionally equivalent to the C source application. C2Eif is implemented in Eiffel, and it is available as a standalone tool.

Input and output. Figure 1 shows the overall workflow of using C2Eif. It inputs C projects (applications or libraries) preprocessed with the C Intermediate Language (CIL) framework [2]. CIL simplifies programs written in C into a subset of C amenable to program transformation. Using CIL input ensures complete support of the whole set of C statements, without polluting the translation with variants and special cases only to deal with syntactic sugar. C2Eif translates CIL programs to Eiffel projects consisting of collections of classes.

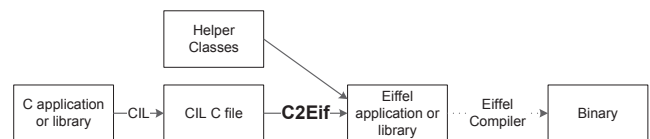


Fig. 1. Overview of how C2Eif works.

Incremental translation. C2Eif builds an initial AST by parsing the input C program, and transforms it into a target AST that can be pretty-printed as Eiffel code. Following a modular design that improves maintainability, C2Eif implements the translation from C to Eiffel as a series of successive incremental transformations on the AST. Every transformation targets exactly one language aspect (for example, loops or inlined assembly code) and produces an AST that combines C features with Eiffel extensions: the code progressively *morphs* from C to Eiffel. C2Eif can output the AST in text form after any of the intermediate transformations; the output is a mixture of C and Eiffel code that is readily understandable by programmers familiar with both languages. The current implementation uses about 40 transformations.

	SIZE (LOCS)		#EIFFEL CLASSES	TRANSLA- TION (S)	BINARY SIZE (MB)
	CIL	EIFFEL			
hello world	8	15	1	1	1.3
micro httpd	565	1,934	16	1	1.5
xeyes	1,463	10,661	78	1	1.8
less	16,955	22,545	75	5	2.6
wget	46,528	57,702	183	25	4.5
links	70,980	100,815	211	33	13.9
vim	276,635	395,094	663	144	24.2
libcurl	37,836	65,070	289	18	–
libgmp	61,442	79,971	370	21	–
libgsl	238,080	344,115	978	85	–
gcc (torture)	147,545	256,246	2,569	79	1,576
TOTAL	898,037	1,334,168	5,433	413	1,626

TABLE I
AUTOMATED TRANSLATION OF OPEN-SOURCE PROGRAMS.

III. EVALUATION

Table I shows data about 10 open-source C programs and one testsuite (for `gcc`) translated to Eiffel with C2Eif. The 10 programs include 7 applications and 3 libraries; all of them are widely-used in Linux and other *nixes. For each entry Table I reports: the size (in lines of code) of the CIL version of the C code and of the translated Eiffel code; the number of Eiffel classes created; the time (in seconds) spent by C2Eif to perform the source-to-source translation (not including compilation from Eiffel source to binary); the size of the binaries (in MBytes) generated by EiffelStudio¹.

We ran extensive trials on the translated programs to verify that they behave as in their original C version. In addition to informal usage, we ran standard testsuites on the 3 libraries totalling over 770 tests. All tests execute and pass on both the C and the translated Eiffel versions of the libraries, with the same logged output. The `gcc` torture testsuite includes 1002 tests that pass after being processed by CIL; C2Eif (which depends on CIL) passes 989 (nearly 99%) and fails 13 of these tests (see [1] for a detailed discussion of the reasons for the 13 failures). Given the variety of the programs considered (and the challenging nature of the torture testsuite), the experiments are strong evidence that C2Eif handles the complete C language used in practice, and produces correct translations.

Performance. We performed systematic performance tests for some of the applications; the results of the tests are described in detail in [1]. In summary, the performance overhead in switching from C to Eiffel significantly varies—from negligible to noticeable—with the program type. Even in the cases where the slowdown is noticeable, it does not preclude the usability of the translated application or library in normal conditions.

Safety. While running the Eiffel translation of `libgmp` we detected 3 bugs (1 in the library itself and 2 in the accompanying testsuite) thanks to the dynamic checks introduced automatically by C2Eif. More generally, C2Eif translations are often more robust and easier to debug than the original C versions thanks to the contracts introduced by C2Eif and by the tighter Eiffel runtime.

¹We do not give a binary size for libraries, because EiffelStudio cannot compile them without a client.

	target language	completely automatic	available	readability	external libraries	pointer arithmetic	gotos	inlined assembly
C2Eif	Eiffel	yes	yes	+	yes	yes	yes	yes
Ephedra [3]	Java	no	no	+	no	no	no	no
C2J++ [4]	Java	no	no	+	no	no	no	no
C2J [5]	Java	no	yes	–	no	yes	no	no
C++2Java [6]	Java	no	yes	+	no	no	no	no
C++2C# [6]	C#	no	yes	+	no	no	no	no

TABLE II
TOOLS TRANSLATING C TO O-O LANGUAGES.

IV. RELATED WORK

A number of tools are available that translate C to object-oriented languages. Table II shows a summary feature comparison among the currently available tools. For each tool, Table II reports:

- The *target language*: Eiffel, Java, or C#.
- Whether the tool is *completely automatic*, that is whether it generates translations that are ready for compilation without need for any manual rewrite or adaptation.
- Whether the tool is *available* for download and usable. In a couple of cases we could only find papers describing the tool but not a version of the implementation working on standard machines.
- An assessment of the *readability* of the code produced. The evaluation of this aspect—subjective to a certain extent—is based on the analysis of sample programs translated using the various tools; when the tool was not available, we analyzed translation examples discussed in the tool documentation. In each case, we tried to evaluate if the translated code is sufficiently similar to the C source to be readily understandable by a programmer familiar with the latter.
- Whether the tool supports unrestricted calls to *external libraries*, unrestricted *pointer arithmetic*, unrestricted *gotos*, and inlined *assembly code*.

See [1] for a discussion of other work related to C2Eif; and [3] for a comparison among Ephedra, C2J++ and C2J.

Acknowledgements. This work was partially supported by ETH grant “Object-oriented reengineering environment”.

REFERENCES

- [1] M. Trudel, C. A. Furia, M. Nordio, B. Meyer, and M. Oriol, “C to O-O Translation: Beyond the Easy Stuff,” in *WCRE*, 2012.
- [2] G. C. Necula, S. McPeak, S. Rahul, and W. Weimer, “CIL: Intermediate Language and Tools for Analysis and Transformation of C Programs,” in *Conference on Compiler Construction*, 2002, pp. 213–228.
- [3] J. Martin and H. A. Müller, “Strategies for migration from C to Java,” in *CSMR*. IEEE Computer Society, 2001, pp. 200–210.
- [4] E. Tilevich, “Translating C++ to Java,” in *First German Java Developers’ Conference Journal*, Sun Microsystems Press. IEEE Computer Society, 1997.
- [5] Novosoft, “C2J: a C to Java translator,” http://www.novosoft-us.com/solutions/product_c2j.shtml, 2001.
- [6] Tangible Software Solutions, “C++ to C# and C++ to Java,” <http://www.tangiblesoftwaresolutions.com/>.