

Scalify.NET and joint compilation of Scala and C#

© Miguel Garcia, STS, Hamburg University of Technology
<http://www.sts.tu-harburg.de/people/mi.garcia>

March 10, 2010

Contents

1 Background	1
1.1 Resources	2

Abstract

The adoption of Scala on .NET can go faster if synergies exist with other tools, to increase the value of developers' investments in Scala. One such accompanying tool is put forward in these notes: a converter from C# into Scala, Previous work is discussed to jumpstart the implementation effort.

1 Background

Right now, not many developers are using the Scala compiler on .NET or Mono, but that is about to change as the current cross-compiler is replaced by an improved (self-hosting) native compiler. Once that happens, some developers might feel fazed by the lack of Scala programs using .NET libraries. To give them some code to play with, the output of a C# frontend (i.e., desugared Abstract Syntax Trees) can be unparsed (ok, after some rewriting) into Scala sources.

It's clear that, in general, the resulting Scala code will *not* exhibit best-practices, as that kind of translation requires human cognition, being thus AI-complete. Rather, Scalify.NET is an interesting project for compiler hackers :-)

As a proof of concept, the Scalify tool performs a similar conversion, from Java to Scala (granted, C# has `goto`). The proposed tool (Scalify.NET) would also contain functionality re-usable to support mixed-source compilation of Scala and C# sources on .NET. Much like joint compilation works nowadays for projects comprising both Java and Scala sources, also in case mutual dependencies exist among them.

“Mixed-source compilation” is kind of an oxymoron, as each compiler usually knows only about one language. Because of this, mixed-source projects are routinely compiled in stages, one for each language (assuming no cyclic dependencies exist across sources in different languages). Two overviews of the issues involved:

- Scala and Java, where cyclic dependencies are handled by `scalac`,
<http://www.codecommit.com/blog/scala/joint-compilation-of-scala-and-java-sources>
- F# and C#, where staged compilation is required.
<http://cs.hubfs.net/forums/thread/4848.aspx>

1.1 Resources

There is previous work on bidirectional translations between VisualBasic and C#, however targeting Scala from any of them would be a first (there's also no translator from C# to F#, for that matter). When targeting Scala, the closest previous work is Scalify:

- <http://wiki.jvmlangsummit.com/Scalify>
- <http://github.com/paulp/scalify>
- <http://video.google.com/videoplay?docid=-3493190786110154189#>

The resources listed next focus on the front-end part of the problem. A compiler-writer's guide to C#

- <http://cartesianclosed.com/pub/csharp/talk.pdf>

There appears to be no single best C# frontend, but one strong candidate is the Mono C# compiler (`mcs` for short). Summary: feature complete for version 3.0 of the language, 4.0 preview available.

- Introductory info, http://www.mono-project.com/CSharp_Compiler
- Internals of `mcs`, <http://anonsvn.mono-project.com/viewvc/trunk/mcs/docs/compiler.txt>
- The codebase can be explored in Visual Studio, check the `.sln` file at <http://anonsvn.mono-project.com/viewvc/trunk/mcs/mcs/>
- Subversion Client Access, http protocol = <http://anonsvn.mono-project.com/source/trunk>

Tapping the Mono compiler

<http://www.arlt.eu/blog/2007/03/12/tapping-the-mono-cs-compiler-mcs/>

For those cases where the input C# programs contain usages of `goto`, a subset of the techniques to recover structured control-flow constructs used by decompilers [1] can be used. The C# community will be thankful for those refactoring recipes, too.

References

- [1] Jerome Miecznikowski and Laurie J. Hendren. Decompiling Java bytecode: Problems, traps and pitfalls. In *CC'02: Proceedings of the 11th International Conference on Compiler Construction*, pages 111–127, London, UK, 2002. Springer-Verlag. <http://www.sable.mcgill.ca/publications/papers/2002-2/sable-paper-2002-2.ps.gz>.