

# Computer Science Tripos Project Progress Report

## Decompilation of .NET bytecode

Stephen Horne (srh38), Trinity Hall College

Originator: Jeremy Singer

29 January 2004

**Project Supervisor:** Eben Upton

**Director of Studies:** Dr S. W. Moore

**Project Overseers:** Dr J. G. Daugman & Dr J. M. Bacon

## Summary of progress

The project is proceeding well, and whilst I am technically behind schedule I am confident that it will be finished to my satisfaction well before the deadline. The following sections describe the progress I have made against the first half of the work blocks laid out in my proposal.

### Block 1 - Learning C# and .NET

In order to aid the task of learning C# and the platform it is based around I bought *C# and the .NET Framework* by Andrew Troelson. Having read the first half of this I gained a good understanding of the differences between C# and Java, and the way in which .NET programs work. I also conducted a significant amount of research into the Common Intermediate Language (CIL) that .NET programs are compiled into.

### Block 2 - Researching decompilation methods

As stated in my progress report I intended to base my decompiler on Cristina Cifuentes's PhD thesis "Reverse Compilation Techniques". I read some of this before starting work on the project, and have since gone over it in greater detail in order to see how the techniques described might be applied to my project.

### Blocks 3 to 5 - Implementing the decompiler

Initially I planned to create the front-end (CIL reader and abstract syntax tree generator), Universal Decompilation Machine (data and control-flow analysis) and back-end (target code generator) sequentially and independently. However, as I progressed with the front-end I decided it was probably a good idea to create skeleton code for the other two sections in order to ensure they could interact correctly. This has led to the three sections being developed in tandem, and as such none of them are fully completed. However, the code has been developed to the stage where a simple class containing no branches decompiles to recognisable source code. The features implemented by each section are documented as follows:

**Front-end:** Almost half of the CIL instruction set is now handled by the front-end of the decompiler. An executable .NET program can be read in and each method converted to a graph representing the basic blocks of the procedure. Each node of the graph contains a list of its instructions, converted from CIL into an intermediate form. Arguments read from the stack for various operations are turned into named registers.

**UDM:** There doesn't need to be as much dataflow analysis as I originally thought there could be, since little optimisation of the code occurs. The data-flow analysis phase currently just performs register copy propagation, where defined registers are replaced by their assigned expression in subsequent references to them. The control-flow analyser now successfully divides each graph into intervals, collapses each interval into a new node, and repeats the

process until the same graph is generated each time. This creates the derived sequence of graphs, which can be used to extract control structure information such as the nesting level of loops.

**Back-end:** The back-end makes use of the Code Document Object Model (CodeDOM); this is a mechanism included in the .NET framework for programmatically generating source code for various .NET-enabled languages. Instructions from the processed graph are converted into the form of abstract syntax tree required by CodeDOM. This can then be used to automatically generate code for any language supporting CodeDOM; currently only C# and Visual Basic .NET but support for others can be added.

## Conclusion

This then describes what progress has been made so far with regard to the original schedule. I should now be in the stage of testing the core routines, but given that I am testing as I go along, and allowed over a month between the last work block and the final deadline I am not overly concerned. A web-site for the project detailing current progress and containing source code for download can be found at <http://hal.trinhall.cam.ac.uk/~srh38/project>.