Software Component Technology Group

Master Thesis

Verifying Spec# delegates

Samuele Gantner

31-03-2008

Supervisors: Prof. Peter Müller, Joseph N. Ruskiewicz

Introduction The Spec# programming language is an extension of the .NET programming language C#, consisting of specification constructs like pre- and postconditions, non-null types and some facilities for higher-level data abstractions [1]. Function objects are used to express higher-order features in object-oriented programs [3]; the C# *delegate* construct simplifies the implementation of such objects. A delegate instance represents a method together with a possible target object or a list of methods and possible target objects in case of multicast delegates. In [7] P. Müller and J. Ruskiewicz propose a verification methodology for C# delegates which allows sound static reasoning about precondition, postcondition and frame-condition of the underlying method associated with a delegate instance. The methodology is restricted to single cast delegates with exactly one underlying method and target.

Goal of the project This proposed thesis will extend the verification methodology described in [7] in order to handle the specific Spec# needs; specifically the methodology will be extended to delegates instantiated with static methods, multicast delegates and events.

Once these extensions have been shown sound, the results will be implemented in the Spec# programming system. In order to show the usefulness of the idea and the practicality of the proposed solution, some case studies will be performed on common delegate usage scenarios and on design patterns based on delegates.

Main parts and Timeline In the first part of the thesis the methodology will be extended and modified to fit specific programming needs. In the second part the methodology will be implemented in the Spec# programming system.

2 weeks	Get to know Spec#
7 weeks	Extension of the methodology
10 weeks	Implementation
2 weeks	Case studies
3 weeks	Write the report

Possible extensions Possible extensions to this thesis include:

- Extension of the methodology and the implementation to include the concepts of immutability and purity [4].
- Extension of the methodology to handle static fields in delegate invariants. This extension is subject to the progress of the implementation of static class invariants [5] in Spec#.
- Extension of the methodology for using history invariants with delegates [6].
- Extension of the methodology to the anonymous delegates of .NET 3.0 [2].

References

[1] Mike Barnett, K. Rustan M. Leino, and Wolfram Schulte. The Spec# programming system: An overview. In *LNCS*, volume 3362, 2004.

[2] ECMA International. Ecma-334: C# language specification, 2006.

[3] G. T. Leavens, K. R. M. Leino, and P. Müller. Specification and verification challenges for sequential object-oriented programs. *Formal Aspects of Computing*, 19(2):159–189, 2007.

[4] K. R. M. Leino. Spec sharp object primer. http://channel9.msdn.com/wiki/default.aspx/SpecSharp.SpecSharpObjectPrimer.

[5] K. R. M. Leino and P. Müller. Modular verification of static class invariants. In J. Fitzgerald, I. Hayes, and A. Tarlecki, editors, *Formal Methods (FM)*, volume 3582 of *Lecture Notes in Computer Science*, pages 26–42. Springer-Verlag, 2005.

[6] K. Rustan M. Leino and Wolfram Schulte. Using history invariants to verify observers. In *ESOP*, pages 80–94, 2007.

[7] P. Müller and J. N. Ruskiewicz. A modular verification methodology for C# delegates. In U. Glässer and J.-R. Abrial, editors, *Rigorous Methods for Software Construction and Analysis*, 2007. To appear.