Chair of programming methodology

Master Thesis

Implementation of Frozen Objects into Spec#

Florian Leu

02-04-2009

Introduction The Spec# programming system [1] is an extended version of C# which adds non-null types, method contracts, object invariants and an ownership model. It consists of a programming language, a compiler and a static verifier called Boogie [3]. Object immutability is a familiar concept that allows safe sharing of objects. So far Spec# support for this concept is based on immutable classes. However, this is restrictive because programmers can neither make instances of arbitrary classes immutable, nor can they control when an instance becomes immutable.

Goal of this master's project is to implement a more flexible technique to enforce immutability not just on the class level but based on individual objects as presented in [4]. Using Spec#'s dynamic ownership model [2], this is achieved by capturing those objects to be owned by a special freezer object, which prevents further modification from this point on. Immutable classes are a special case of this methodology, where each instance is frozen at the end of the constructor. Therefore the implementation of immutable classes will be replaced by this new methodology without affecting previously written Spec# programs. The second part is to extend the Boogie methodology to make use of the newly added immutable objects. It has to be guaranteed that immutable objects satisfy their invariants in all states. Invariants of other objects may then depend on those immutable shared objects which couldn't be allowed before.

The main parts of this project are (with time estimations):

1. making myself familiar with the task and learn to build, understand and work with the source code of Spec# and Boogie in Visual Studio (3 weeks)
2. adding a special freezer object, which cannot be referred to by the rest of the program, and corresponding verification axioms (4 weeks)
3. add a freeze operation which allows transferring ownership of objects to the freezer object (1 weeks)
4. add a frozen modifier to denote objects which are immutable (2 weeks)
5. extend Boogie to guarantee that frozen objects satisfy their invariants in all states (1 week)

6. extend Boogie to allow invariants to depend on frozen objects (3 weeks)
7. replace the previous implementation of immutable classes with (a special case of) the newly implemented methodology (2 weeks)
8. test the correctness of the implementation with a test suite (1 weeks)

Possible Extensions

1. evaluate Boogie for possibilities to use the newly implemented frozen object methodology (e.g. allow objects in a method precondition which had to be peer consistent before to be either peer consistent or frozen now because both guarantees their validity)
2. examine the source code of Spec# and Boogie for classes where making some of their objects frozen would simplify some verification proofs

References

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

[2] K. Rustan M. Leino, and Peter Müller. Object Invariants in Dynamic Contexts. In LNCS, volume 3086. Springer, 2004.

[3] Mike Barnett, Bor-Yuh Evan Chang, Robert DeLine, Bart Jacobs, and K. Rustan M. Leino. Boogie: A modular reusable verifier for object-oriented programs. In LNCS, volume 4111. Springer, 2006.

[4] K. Rustan M. Leino, Peter Müller, and Angela Wallenburg. Flexible Immutability with Frozen Objects. In VSTTE, volume 5295. Springer, 2008.