Software Component Technology Group

Master Thesis

Translating invariant proofs between Spec# and JML

Benjamin Morandi

1.11.2007

Introduction Spec# [1] and JML [2] are steps towards industry applicable specification and verification solutions. Spec# is a solution targeted to the .NET platform on top of C#. Its methodology [3, 4] combines ownership based invariants with visiblity based invariants. JML uses a similar methodology [5] to provide the Java world with specification and verification.

Although, from an abstract point of view Spec# and JML share a goal and have common elements, there are some differences in the underlying methodologies. Spec# and JML use somehow similar ownership models. However, while Spec# encodes ownership dynamically by means of ghost fields, invariants, and new statements (e.g., pack and unpack), JML encodes ownership statically, via the Universe Type System (UTS) [5]. This fundamental difference enables Spec# to handle programming patterns which cannot be typed in UTS. With respect to the semantics of invariants, JML is underpinned with the relevant invariant semantic, which guarantees that all the invariants of the objects within the same ownership context as the executed method's receiver object hold upon the entry and exit of the method. Concerning the semantics of invariants, Spec# makes use of ghost fields to encode when an invariant of an object can be assumed to hold. In addition, a Spec# object invariant can hold at different levels within the inheritance hierarchy. This allows, in particular, the invariant in a Spec# class T to depend on fields declared in T's proper superclasses. Such an invariant would not be admissible in JML.

Despite the above differences, it is possible to formalize a bidirectional translation between Spec# and JML programs for a limited subset of their specification. The existence of such a translation scheme is useful, since it enables one to carry over research results from Spec# to JML and vice versa.

The goal of this master's project is to develop a bidirectional translator between Spec# and JML programs based on the already existing formalization of the translation. The translator will either take a Spec# program and produce a JML program, or take a Spec# program and produce a JML program.

The main parts of this project are:

1. design intermediate representations of Spec# and JML programs
2. develop parsers for Spec# and JML programs

3. develop the bidirectional translator between Spec# and JML programs based on the intermediate representations
4. develop code generators for Spec# and JML programs
5. develop test cases

Since the current bidirectional translation scheme only covers a limited subset of the Spec# and JML specification, a possible extension is to enable the translation to be able to deal with an extended subset of the specifications.

This project is supervised by Dr. Nicu Georgian Fruja and Prof. Dr. Peter Müller

Literatur

[1] *Spec# Website*, http://research.microsoft.com/specsharp/

[2] *JML Website*, http://www.eecs.ucf.edu/∼leavens/JML/

[3] Mike Barnett, Robert DeLine, Manuel Fähndrich, K. Rustan M. Leino, Wolfram Schulte, *Verification of object-oriented programs with invariants*. Journal of Object Technology, Volume 3, Number 6

[4] K. Rustan M. Leino, Peter Müller, *Object Invariants in Dynamic Contexts*. European Conference on Object-Oriented Programming, 2004

[5] Peter Müller, Arnd Poetzsch-Heffter, Gary T. Leavens, *Modular Invariants for Layered Object Structures*. Science of Computer Programming, 2006