Extending JML with Temporal Logic

Kerry Trentelman

A Behavioural Interface Specification Language, or BISL incorporates a behavioural specification, describing what a particular program does, with a interface specification, describing how a module can be used. A particular brand of BISL is the Java Modeling Language, JML [1]. JML, developed by Leavens and Baker at Iowa State University, is used to specify Java program modules of classes and interfaces.

Along with its predecessors Larch and Eiffel, JML is based on the Design by Contract methodology. With Design by Contract, a program fragment is specified with pre and post conditions. These specifications can be thought of as contracts between the client and the implementer. Even though these specifications do not necessarily guarantee what a fragment will do, they give us a precise description of what a fragment is supposed to do. This provides better guidance when testing and debugging.

Two examples of current projects involving JML are the ESC/Java and LOOP projects. The Extended Static Checking project at Compaq Systems Research Center have developed a tool, ESC/Java, for detecting errors in Java programs. The LOOP, (or Logic of Object Oriented Programming) project at the University of Nijmegen is interested in using proof tools such as PVS and Isabelle to specify and verify properties of classes in object-oriented programming languages [2]. Their LOOP compiler translates Java classes into theories of PVS and Isabelle. Originally, the properties that one wished to prove about the Java programs needed to be written in the language of the the theorem provers. However, using JML, program assertions can be written in the same code as the program itself. One of the goals of the project is to extend the compiler so that it is able to translate Java programs annotated with JML [3].

The LOOP project's work is also highly applicable to a language subset of Java, JavaCard. The group have been working on a formal specification in JML of the JavaCard API in order to enable verification that the current reference implementation satisfies these specifications via the LOOP tool [4]. However, this formal specification is lacking in temporal aspects, although temporal statements are present in the API.

My work is an attempt to incorporate temporal statements within the specification in JML of the JavaCard API. Having located areas within the

JavaCard API in which temporal specifications would be beneficial, I have identified a number of constructions needed to specify them. I plan to propose a concrete JML syntax and a semantics for this.

References

[1] *Preliminary design of JML: A Behavioural Interface Specification Language for Java.*

[2] M Huisman. *Reasoning about Java programs in higher order logic with PVS and Isabelle*. PhD thesis, University of Nijmegen, 2000.

[3] B Jacobs and E Poll. A logic for the Java modeling language JML. In *proceedings of Fundamental Approaches to Software Engineering 2001, Volume 2029 of LNCS.*

[4] H Meijer and E Poll. Towards a full formal specification of the JavaCard API. In *proceedings of International Conference on Research in Smart Cards, E-smart 2001.*