ESF Exploratory Workshop (EW05-119):

Challenges in Java Program Verification

Nijmegen, The Netherlands, 16-18 October 2006

Scientific Report

Reiner Hähnle, Wojciech Mostowski, Erik Poll

1. Executive Summary

The workshop managed to gather nearly all research groups world-wide currently working on formal verification of programs written in Java and related languages.

The workshop was described by all participants as an extraordinarily focused event with discussions taking place on a very high technical level.

An explicit aim of the workshop was not only to take stock of the latest advances in formal verification of Java programs, but also to come to an understanding within the research community on the most important future issues and how to cross-fertilise each other's research.

The following trends were identified as likely to have particular influence on the shape of research to take place in verification in the future:

- The definition of *intermediate languages* for specifications and target programs. Ideally, there should be only few such languages and they should have a formal semantics. This will lead to clear tool architectures and inter-operability.
- *Compilation of specifications and proofs* allows to obtain bytecode level proofs while retaining the ability to verify interactively at the source code level.
- *Symbolic execution* of code as a powerful multi-purpose analysis tool.
- *Hybrid methods* that combine technologies from verification, program analysis, and specification to attack complex problems.
- Tight integration of verification methods and tools with those in *conventional software development*. The “customer” of verification technology should be more the software developer, less the formal methods expert. Addition of language-based support for ensuring stronger *modularity* notions than is possible at the moment in Java-like languages.

The following issues were perceived as challenges that the community must respond to in a concerted effort:

- Verification of *concurrent* programs and the definition of a concurrency model that makes verification feasible.
- Verification of larger *frameworks* and of *automatically generated code*.
- The consolidation and integration of *specification languages* for object-oriented programs.

The participants of the workshop clearly see the need for continued and regular collaboration on the issues sketched here. The COST framework is seen as a possible instrument to supply the necessary

infrastructure for that.

2. Scientific Content and Results

The workshop managed to gather nearly all research groups that are currently working on formal verification of programs written in Java and closely related languages. Only two groups were not present (Klaus Havelund from Nasa Research, U.S.A., and Francesco Logozzo from École Polytechnique, France): their representatives had to cancel the initial agreement to come, because of unforeseen circumstances.

The workshop was described by all participants as an extraordinarily focused event with discussions taking place on a very high technical level.

An explicit aim of the workshop was not only to take stock of the latest advances in formal verification of Java programs, but to come to an understanding within the research community on the most important future issues and how to cross-fertilise each others research.

During the talks and discussions a number of important trends were identified that are likely to shape formal verification technology in the next years and will push it forward.

Intermediate Languages Many people stressed the importance of intermediate programming and specification languages for reducing the complexity of verification systems. Many systems work by first compiling Java or C# into a simpler intermediate programming language (Beckert, Leino, Müller) before starting verification condition generation or symbolic execution. Likewise, specification languages such as JML or OCL are desugared and/or compiled to, for example, typed predicate logic (Giese, Marché) instead of reasoning about them directly.

During the panel discussion on “Past Experiences and Lessons for the Future” the importance of high-quality intermediate representations for building inter-operable, maintainable tools was stressed.

Compilation Compilation is a standard principle in Programming Languages and used for run-time efficiency as well as for vertical structuring of complex systems. It is only now beginning to be explored in the context of formal verification. One approach is the *compilation of proofs* of verification tasks at the source code level to corresponding proofs at the bytecode level (Pavlova). The motivation for such compilation is similar as for the usual compilation of high-level programming languages: it allows one to obtain bytecode level proofs while still being able to verify interactively at the source code level. This is important, because the bytecode level is not suitable for human interaction in the context of formal proofs. Second, bytecode is syntactically simpler and has a smaller instruction set than source code. As a consequence, a bytecode proof checker has a smaller code basis and memory footprint than one for source code. This is a serious issue for checking of certificates on small mobile devices in a proof-carrying code architecture.

Symbolic Execution Several tools feature complete symbolic execution engines. One of the advantages is that symbolic execution can be used not only for verification, but also for code-based test generation, visualisation, and automated bug finding (Robby, Rümmer). In addition, symbolic execution is used as a powerful dynamic analysis tool in order to increase automation of verification proofs that otherwise require induction (Hähnle, Wallenburg).

Hybrid Methods Several of the most advanced verification activities involve not merely formal verification, but, for example, refinement and verification (Stenzel), model checking and deduction (Mostowski), program analysis and verification (Hähnle). Clearly, there is tendency to integrate verification based on logical inference with methods developed in

program analysis and in specification.

Close Interaction with Conventional Software Development In several contributions it was stressed that it is necessary to deeply integrate verification with more conventional software development tools. Verification tools must be readily usable, pluggable, robust.They must be designed for users, not for other researchers. This is an insight that the community started to address seriously (talks by Kiniry, Robby, Leino).

In addition, in the creation of verification tools it is essential to use state-of-art software engineering (SWE) techniques for development and documentation.

The verification community has detailed insights in specifications that are not well understood or incomplete or simply too complex to be understood. There should be more dialogue between the authors of Java JSRs and the formal verification community. The analyses obtained from attempts to formally treat such concepts as the Java Memory Model (Huisman) or Java Card Atomic Transactions (Mostowski) suggest that these aspects of the Java/Java Card language are in serious need to becoming more precise and less complex.

Language-Based Modularity It has been long known in the formal verification community that the design and complexity of target programming languages in which the verification objects are written has a strong influence on the difficulty of the ensuing verification tasks. In the past, a number of "clean" languages that are particularly suitable for verification have been designed (for example, guarded commands, Pascal subsets, mutually recursive functions), but they are too far away from what is required in an industrial context. The community represented in this workshop, on the other hand, starts from an industry standard (Java, Java Card, C#) and tries to handle as many features as possible.

One of the more serious shortcomings of object-based languages such as Java is lacking support for writing modular programs: change or extension of existing classes can, in principle, break numerous invariants and contracts. Worse, there is no obvious way to determine which invariants and contracts are preserved. This situation extends to verification proofs: classes and methods cannot be verified in isolation, that is, modularly, because their invariants may depend on other classes. The solution that the Java verification community (talks by Darvas, Jacobs, Leino, Middelkoop, Müller, Poetzsch-Heffter, Poll) has to offer here is to *extend* the target language with concepts that allow the developer to specify modularity assumptions more explicitly. In several talks solutions based on ownership type systems and on directives for controlled representation exposure were discussed. The current challenge is to keep the programming model of these new constructs simple enough in order for the average developer to benefit from it.

This is an area where verification and programming language design can fruitfully interact.

Each of the issues discussed so far contains several research challenges. However, during the talks and discussions a number of additional topics emerged that need to be addressed systematically in the future.

Concurrency The first Java and C# verification tools now support concurrent programs (Klebanov, Jacobs), but current restrictions are serious. In addition, the concurrency model of Java is considered to be too low-level by the verification community (and by many Java developers).

Frameworks, Code Generation Traditionally, formal verification centred on hand-written code on the module level. In practice, however, large frameworks such as EJB or code generators from more abstract models account for more and more executed code. Modularity and Compilation will be essential technologies, but overarching strategies for verification of

large systems need yet to be developed.

Specification Languages The de-facto standard for specification of the formal requirements to be verified is the Java Modeling Language (JML). In addition, the C# group uses Spec# (which shares many features with JML) and one group also supports the UML standard Object Constraint Language (OCL). While JML is sufficient in many situations, sometimes it lacks expressivity. Several people use "low-level" formalisms such as first-order logic with abstract data types in addition to JML (Marché, Mostowski). In particular, concurrent programs cannot yet be specified. Another problem of JML is the lack of a formal semantics that could guarantee inter-operability of all JML-based tools. A serious issue is the lack of integration of JML with UML-based CASE tools and IDEs.

Clearly this challenge cannot be answered by the Java verification community alone. A concerted effort is necessary.

3. Outcomes

The workshop was highly interactive and, in addition to individual presentations, featured also a number of (panel) discussions. Of these, we would like to highlight two in particular. The first was about "Past Experiences and Lessons for the Future" and it featured a representative of each of the verification systems presented at the workshop. Each panelist was asked to name two positive and two negative things they learned from their tool experiences. The emerging picture was remarkably coherent and is partly taken up in the list of issues above. A summary of the discussion has been made available in the form of a blog[1] by the panel discussion leader, Joe Kiniry.

We also wish to report briefly on the discussion about follow-up activities to the workshop: various research funding programmes within ESF and FP7 were discussed. As this summary of the scientific results shows, there is serious need for the research groups in Java verification to collaborate even more closely and on a regular basis. It was felt that it is more important to establish a research network rather than apply for funding of a project. Most people thought that a COST action would provide an adequate infrastructure, because it is a relatively flexible and open instrument. An informal inquiry among the participants produced a sufficient number of teams that would be seriously interested. Within an appropriate COST action one could also initiate the necessary dialogue with the formal specification community.

4. Worshop Webpage

At the Workshop webpage[2] all post-workshop information and notes provided by the participants and organisers are published.

5. Final Program

Monday October 16th

9:00-9:30 *Introduction*
Reiner Hähnle - Welcome
Kaisa Sere - Presentation of the European Science Foundation

1 http://kindsoftware.blogspot.com/2006/11/esf-workshop-on-challenges-in-java.html

2 http://www.cs.ru.nl/~woj/esfws06

9:30-10:30	*Session 1a: Verification Frameworks* Bernhard Beckert - Dynamic Logic with Non-rigid Functions Kurt Stenzel - Refinement of Abstract Specifications to Java Code
10:30-11:00	*Coffee Break*
11:00-12:00	*Session 1b: Verification Frameworks* Mariela Pavlova - Java Bytecode Verification Martin Giese - A Logic with Subtypes to talk about Java Objects
12:00-13:30	*Lunch*
13:30-15:00	*Session 2: Concurrency* Vladimir Klebanov - A Dynamic Logic for Verification of Concurrent Programs Bart Jacobs - A Statically Verifiable Programming Model for Concurrent Object-Oriented Programs Marieke Huisman - Formalising the Java Memory Model
15:00-15:30	*Coffee Break*
15:30-17:00	*Discussion 1: Unsolved problems in Java Verification* See Discussions below

Tuesday October 17th

9:00-10:00	*Session 3: Tools* Joe Kiniry - Usable Formal Tools Robby - Bogor/Kiasan - Combining Symbolic Execution, Model Checking, and Theorem Proving
10:00-10:30	*Coffee Break*
10:30-12:00	*Session 4: Applications* Jean-Louis Lanet/Pierre Girard - New Challenges in Java Verification for Tiny Devices Wojciech Mostowski - Verifying Real Java Programs - API Calls vs. Language Semantics, Official Specs vs. Implementations Peter Müller - Specification and Verification Challenges
12:00-13:30	*Lunch*
13:30-15:00	*Session 5: Loops* Reiner Hähnle - Verification of Loops by Parallelization Steffen Schlager - A Method for Loop Verification Based on Fixed Points Angela Wallenburg - Using Induction in Interactive Verification

15:00-15:30	*Coffee Break*
15:30-16:30	*Discussion 2: Challenges*
	See Discussions below
16:30-17:00	*Discussion 3: Follow-up Activities*
	See Discussions below

Wednesday October 18th

9:00-10:00	*Session 6: Encapsulation*
	Arnd Poetzsch-Heffter - The Box-Model: A Modular Semantics for Modular Verification
	Rustan Leino - Going Beyond Simple Ownership System in Spec#
10:00-10:30	*Coffee Break*
10:30-12:00	*Session 7a: Specification and Verification Techniques*
	Adam Darvas - Verification Technique for Method Calls in Specifications
	Erik Poll/Christian Haack - Immutable Objects in Java
	Philipp Rümmer - Disproving in Dynamic Logic for Java
12:00-13:30	*Lunch*
13:30-14:30	*Session 7b: Specification and Verification Techniques*
	Ronald Middelkoop - Invariants for Non-Hierarchical Object Structures
	Claude Marché - Algebraic Specifications for Modeling Java Programs (Example)
14:30-15:00	*Coffee Break*
15:00-16:30	*Discussion 4: Past Experience and Lessons for the Future*
	See Discussions below
16:30	*Closing*

Panel Discussions

1. Unsolved Problems in Java Program Verification

Abstract: What are the most important problems in Java program verification for which we don't yet have good solutions? What are the current means of attack? Can we identify areas where we can collaborate?

Moderator: Arnd Poetzsch-Heffter

- Mobile Java bytecode and its verification (Mariela Pavlova)
- Java library and software frameworks such as EJB, CORBA (Robby)
- Concurrent Java (Vladimir Klebanov)
- Read and effects systems and ownership systems (Rustan Leino)

2. Challenges

Abstract: What are the strategic directions our field should be headed to? How can we make most impact with our limited resources? Should we try to organize ourselfves more systematically

(COST action, etc.)?

Moderator: Peter Schmitt

- Which problems would industry like to have solved? (Jean-Louis Lanet)
- How to align our research with an overarching security concept for Java applications? (Wojciech Mostowski)
- Our position wrt Hoare's Grand Challenge (Bernhard Beckert)

3. ESF Follow-up Activities

Abstract: How to proceed with the cooperation within the group of participants to support ESF efforts?

Moderator: Reiner Hähnle

4. Past experience and lessons for the future

Abstract: Many of us actively took part in the development of a verification tool for OO languages. As in all research tools one has to make design and technology decisions that in the end may turn out to be suboptimal. Are there things that we can learn from each other? Which mistakes did we make in designing our verification tool? Which technologies worked and which didn't? Which lessons did we learn for the future?

Moderator: Joe Kiniry

Brief statements by:

- Erik Poll on LOOP
- Claude Marché on Krakatoa
- Peter Müller on Jive
- Joe Kiniry on ESC/Java2
- Steffen Schlager on KeY
- Mariela Pavlova on JACK
- Kurt Stenzel on KIV
- Rustan Leino on Boogie
- Robby on Bogor/Bandera

6. Detailed Participant List

Convenors:

1. Reiner Hähnle
Chalmers University of Technology
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
tel: +46-31-772-1061
fax: +46-31-772-3663
e-mail: reiner@cs.chalmers.se
www: http://www.cs.chalmers.se/~reiner/

2. Wojciech Mostowski (local)
Computing Science Department
Radboud University Nijmegen
P.O. Box 9010
6500 GL Nijmegen
The Netherlands
tel: +31-24-365-2076
fax: +31-24-365-3137
e-mail: woj@cs.ru.nl
www: http://www.cs.ru.nl/~woj/

3. Erik Poll (local)
   Computing Science Department
   Radboud University Nijmegen
   P.O. Box 9010
   6500 GL Nijmegen
   The Netherlands
   tel: +31-24-365-2710
   fax: +31-24-365-3137
   e-mail: erikpoll@cs.ru.nl
   www: http://www.cs.ru.nl/~erikpoll/

ESF representative:

4. Kaisa Sere
   Department of Information Technology
   Åbo Akademi University, TUCS -
   Turku Center for Computer Science
   Joukahaisenkatu 3-5
   FIN-20520 Turku
   Finland
   tel: +358-2-215-4537
   e-mail: kaisa.sere@abo.fi
   www: http://www.abo.fi/~kaisa/

Local organiser:

5. Maria van Kuppeveld
   Computing Science Department
   Radboud University Nijmegen
   P.O. Box 9010
   6500 GL Nijmegen
   The Netherlands
   tel: +31-24-365-3132
   fax: +31-24-365-3137
   e-mail: M.Kuppeveld@cs.ru.nl
   www:
   http://www.cs.ru.nl/staff/Maria.van.Kuppeveld

Participants:

6. Philipp Rümmer
   Chalmers University of Technology
   Department of Computer Science and
   Engineering
   SE-412 96 Göteborg
   Sweden
   tel: +46-31-772-1072
   fax: +46-31-772-3663
   e-mail: philipp@cs.chalmers.se
   www:
   http://www.cs.chalmers.se/~philipp/

7. Adam Darvas
   Software Engineering
   ETH Zentrum, RZ F3
   8092 Zurich
   Switzerland
   tel: +41-1-632-7951
   fax: +41-1-632-1435
   e-mail: Adam.Darvas@inf.ethz.ch
   www:
   http://sct.inf.ethz.ch/people/darvas/

8. Peter Müller
   Software Engineering
   ETH Zentrum, RZ F3
   8092 Zurich
   Switzerland
   tel: +41-1-632-2868
   fax: +41-1-632-1435
   e-mail: peter.mueller@inf.ethz.ch
   www:
   http://sct.inf.ethz.ch/people/mueller/index.html

9. Peter Schmitt
   University of Karlsruhe
   Institute for Theoretical Computer
   Science
   Am Fasanengarten 5
   D-76131 Karlsruhe
   Germany
   tel: +49-721-608-4000
   fax: +49-721-608-4211
   e-mail: pschmitt@ira.uka.de
   www:
   http://i12www.ira.uka.de/english/schmitt-engl.htm

10. Steffen Schlager
    University of Karlsruhe
    Institute for Theoretical Computer
    Science
    Am Fasanengarten 5
    D-76131 Karlsruhe
    Germany
    tel: +49-721-608-4338
    fax: +49-721-608-4211
    e-mail: schlager@ira.uka.de
    www:
    http://i12www.ira.uka.de/~schlager/

11. Bart Jacobs
Katholieke Universiteit Leuven,
Department of Computer Science
Celestijnenlaan 200A
3001 Leuven
tel: +32-16-32-7823
fax: +32-16-32-7996
e-mail: bart.jacobs@cs.kuleuven.be
www:
http://www.cs.kuleuven.ac.be/~bartj/

12. Martin Giese
RICAM
Austrian Academy of Sciences
Altenbergerstr. 69
A-4040 Linz
Austria
tel: +43-732-2468-5254
fax: +43-732-2468-5212
e-mail: martin.giese@oeaw.ac.at
www:
http://www.ricam.oeaw.ac.at/people/page/giese/

13. Mariela Pavlova
INRIA Sophia Antipolis
Everest Project
2004, route des Lucioles - BP 93
FR-06902 Sophia Antipolis
France
tel: +33-4-92-38-75-65
e-mail:
Mariela.Pavlova@sophia.inria.fr
www: http://www-sop.inria.fr/everest/personnel/Mariela.Pavlova/

14. Marieke Huisman
INRIA Sophia Antipolis
Everest Project
2004, route des Lucioles - BP 93
FR-06902 Sophia Antipolis
France
tel: +33-4-92-38-79-45
fax: +33-4-92-38-50-29
e-mail:
Marieke.Huisman@sophia.inria.fr
www: http://www-sop.inria.fr/lemme/Marieke.Huisman/

15. Bernhard Beckert
University of Koblenz
Department of Computer Science
AI Research Group
Universitätsstraße 1
D-56070 Koblenz
Germany
tel: +49-261-287-2775
e-mail: beckert@uni-koblenz.de
www: http://www.uni-koblenz.de/~beckert/

16. Vladimir Klebanov
University of Koblenz
Department of Computer Science
AI Research Group
Universitätsstraße 1
D-56070 Koblenz
Germany
tel: +49-261-287-2781
e-mail: vladimir@uni-koblenz.de
www: http://www.uni-koblenz.de/~vladimir/

17. Jing Pan
Technische Universiteit Eindhoven
Department of Mathematics and
Computer Science
Formal Methods Group
Postbus 513
5600 MB Eindhoven
The Netherlands
tel: +31-40-247-4628
e-mail: j.pan@tue.nl
www: http://www.win.tue.nl/~jpan/

18. Kurt Stenzel
Lehrstuhl für Softwaretechnik und
Programmiersprachen
Institut für Informatik
Universität Augsburg
D-86135 Augsburg
Germany
tel: +49-821-598-2178
e-mail: stenzel@informatik.uni-augsburg.de
www: http://www.informatik.uni-augsburg.de/lehrstuehle/swt/se/staff/stenzel/

19. Joe Kiniry
Computer Science and Informatics Centre
UCD Dublin
Belfield
Dublin 4
Ireland
tel: +353-1-716 2929
fax: +353-1-269 7262
e-mail: kiniry@acm.org
www: http://secure.ucd.ie/~kiniry/

20. Robby
SAnToS Laboratory
Department of Computing and Information Sciences
Kansas State University
212 Nichols Hall
Manhattan, KS 66506
U.S.A.
tel: +1-785-532-6350 (Ext. 30)
fax: +1-785-532-7353
e-mail: robby@cis.ksu
www: http://www.cis.ksu.edu/~robby/

21. Jean-Louis Lanet
Gemplus La Ciotat
ZI Athélia III - Voie Antiope
13705 La Ciotat
France
tel: +33-442-36-3266
fax: +33-442-36-5555
e-mail: Jean-Louis.Lanet@gemplus.com

22. Pierre Girard
Gemplus La Ciotat
ZI Athélia III - Voie Antiope
13705 La Ciotat
France
tel: +33-442-36-5791
e-mail: Pierre.Girard@gemplus.com

23. Jean-Marie Gaillourdet
Universität Kaiserslautern
Fachbereich Informatik, Gebäude 34
Postfach 30 49
D-67653 Kaiserslautern
Germany
tel: +49-631-205-2625
fax: +49-631-205-3420
e-mail: jmg@informatik.uni-kl.de
www: http://softech.informatik.uni-kl.de/twiki/bin/view/Homepage/Jean-MarieGaillourdet

24. Arnd Poetzsch-Heffter
Universität Kaiserslautern
Fachbereich Informatik, Gebäude 48
Postfach 30 49
D-67653 Kaiserslautern
Germany
tel: +49-631-205-3536
fax: +49-631-205-3420
e-mail: poetzsch@informatik.uni-kl.de
www: http://www.informatik.fernuni-hagen.de/pi5/mitarbeiter/poetzsch.htm

25. Rustan Leino
Microsoft Research
One Microsoft Way
Redmond, WA 98052
U.S.A.
tel: +1-425-707-8045
fax: +1-425-936-7329
e-mail: leino@microsoft.com
www: http://research.microsoft.com/~leino/

26. Claude Marché
Université de Paris-Sud, LRI
Bâtiment 490
F-91405 Orsay Cedex
France
tel: +33-01-69-15-64-85
fax: +33-01-69-15-65-86
e-mail: Claude.Marche@lri.fr
www: http://www.lri.fr/~marche/

7. Statistics on Participation

The workshop was attended by 25 researchers from 17 institutions/departments/companies:

- Chalmers University of Technology, Computing Science Department, Sweden (2 participants),
- Radboud University Nijmegen, Computing Science Department, The Netherlands (2 participants),
- ETH Zürich, Software Component Technology Group, Switzerland (2 participants),
- Karlsruhe University, Institute for Logic, Complexity, and Deduction Systems, Germany (2 participants)
- Katholieke Universiteit Leuven, Department of Computer Science, Belgium (1 participant)
- Austrian Academy of Sciences, RICAM, Symbolic Computation Group, Austria (1 participant)
- INRIA, Sophia-Antipolis, France (2 participants),
- Koblenz University, AI Research Group, Germany (2 participants),
- Technical University of Eindhoven, Formal Methods Group, The Netherlands (1 participant),
- Augsburg University, Software Technology and Programming Languages Group, Germany (1 participant),
- University College Dublin, School of Computer Science and Informatics, Ireland (1 participant),
- Kansas State University, Department of Computing and Information Sciences, U.S.A., (1 participant),
- Microsoft Research, Redmond, U.S.A. (1 participant),
- Gemalto, France (2 participants),
- Technical University Kaiserslautern, Software Technology Group, Germany (2 participants),
- Université de Paris-Sud, DÉMONS research team, France (1 participant),
- Turku Center for Computer Science, Finland (1 participant – ESF representative).

All groups/institutions that were represented by two participants (except for Gemalto) consisted of at least one junior researcher (all except one still before their PhDs at the time of the Workshop). Within the rest of the group the “academic” age of participants was distributed more or less uniformly, from PhD students to full Professors.

Gender repartition :

Female : 4 Male : 21

Geographical distribution :

AT	1	FI	1	SE	2
BE	1	FR	5	US	2
CH	2	IE	1		
DE	7	NL	4		