Recent Advances in Extended Static Checking

Joe Kiniry, University College Dublin

http://mobius.inria.fr/

Mobius

Contract n° IST 015905
Extended Static Checking for Java

• ESC/Java is an *extended static checker*
  • originated with DEC/Compaq SRC
  • used a minimal annotation language
  • behaves like a compiler
    • error messages similar to javac & gcc
    • completely automated
    • hides enormous complexity from user

• ESC/Java2: an *über*-extended static checker for JML-annotated Java
  • includes about a dozen new ESCers
  • integrated with Eclipse
  • supports multiple logics and provers
  • represents the work of dozens of people
The Mobius Program Verification Environment

- general-purpose architecture for JML/Java-centric program analysis and verification
- formalized toolbus and architecture based upon existing component model
- integration of automatic and interactive provers (e.g., Simplify, CVC3, and Coq)
- ESC/Java2 is evolving into ESC/Java3 within this framework
Evolution of Static and Interactive Checkers/Verifiers

- Note: graph is not to scale
ESC/Java2 Architecture

ESC a package/module/method

command line switches

scanning/lexing

parsing

typechecking

Java to DSA translation

DSA form

Java + JML AST

Java to GC

DSA to GC

Java Source

Java Bytecode

JML specifications

BML specifications

Java Source

Java Bytecode

JML specifications

BML specifications

Java + JML AST

Java to GC

GC language

emit VC to prover

concrete syntax

pretty-print

verification condition

VCgen

Decision

interpret feedback

produce warning

display warning or pass

output "pass" or warning

produce "pass"
ESC’s Key Strengths

- push-button automation
- tool robustness
- user feedback with no user specifications
- integration with popular IDE (Eclipse)
- large amounts of decent documentation
- availability of slides, examples, tutorials
- community size and involvement
- popularity amongst FM community
ESC’s Main Weaknesses

- IP issues because of non-standard license
- platform design, implementation, and documentation problems
- the need for fairly complete specs
- insufficient developer involvement (for all the ideas we have)
- false positives and false negatives; aka soundness and completeness issues
- sometimes complex user feedback
Some Features Available in ESC/Java2 in the Mobius PVE

- nearly full JML coverage (with refinement)
- purity checking
- frame axiom checking
- soundness & completeness warning system
- specification consistency checker
- specification-aware dead code detection
- under-defined specification checking
- universes type checking
- AST and GC graphical rendering
- generic automatic prover interface
- sorted and unsorted verification condition representation
- support for multiple automated and interactive provers
- proof generation from an automated prover
- incorporation of other lightweight static checkers
- process integration
Some Features Available in ESC/Java2 in the Mobius PVE

- nearly full JML coverage (with refinement)
- purity checking
- frame axiom checking
- soundness & completeness warning system
- specification consistency checker
- specification-aware dead code detection
- under-defined specification checking
- universes type checking
- AST and GC graphical rendering
- generic automatic prover interface
- **sorted and unsorted verification condition representation**
- support for multiple automated and interactive provers
- proof generation from an automated prover
- incorporation of other lightweight static checkers
- process integration
Mobius PVE Verification Bus Features

• Mobius VC back-end
  • unsorted and sorted VC representation
  • logic-aware syntax generation to several automatic and interactive theorem provers
    • generation of Mobius VCs in Base Logic in Coq

• Mobius ESC VC back-end
  • generation of ESC VCs in ESC Logics
  • generate ESC VCs for several automatic and interactive theorem provers
  • extended static checking of ESC VCs with rich in-editor feedback
Mobius PVE Verification Bus Features

- Mobius Prover back-end
  - generic interaction with a variety of automatic and interactive theorem provers
    - automatic provers supported
      - Simplify, SMT, CVC3, Yices, Fx7
    - interactive provers supported
      - Coq and PVS
  - proof status maintenance
  - proof unit/smoke testing
  - automatic and seamless proof sharing amongst distributed collaborators
Mobius PVE Verification Bus Features

• integration of other lightweight static checkers
  • e.g., CheckStyle, FindBugs, and PMD
  • tune rules to guide programmer toward writing code that is easier to ESC and verify
  • regular, lightweight checking early is dramatically better than heavyweight checking late
• help system and process management
  • task and feature tracking
  • online hypertext architecture docs and help
Focus of Current Work

• completely FLOSS software foundation
• complete documentation
• rich platform integration
• multiple provers and multiple logics
• FreeBoogie integration
• improved developer feedback
• full integration into software process
• integration with full verification and refinement-based modeling languages
• new, complex case studies
Ongoing Work

• new JML model classes
  • pure, immutable, functional, executable, referential equality-centric, fully unit tested and ESCed, tuned for static checking

• FreeBoogie subsystem
  • FreeBoogiePL = structured and unstructured BoogiePL + explicit heap + separation logic
  • FreeBoogie VC generation to target Mobius VC back-end, thus support multiple provers
Ongoing Work

- reflective unit testing
  - new generation of JMLunit that is specification, source, and bytecode-aware
- EBON-JML bicompiler
  - seamless and reversible translation to/from EBON to JML
  - refinement will support informal and formal documentation, system events, scenarios, allocation, and ownership
- domain-specific annotations with formal semantics integrated into refinement
Ongoing Work

• ESC/Java3
  • use new JML front-end (JML[345])
  • reason about Java 1.5 source \textit{and} bytecode
  • target FreeBoogie as IR
  • leverage recent work in separation logic
  • support automated multiple provers
  • selectively target interactive provers
  • perform contextually aware, cross-prover, cross-logic checking
  • deeper process integration, particularly across large teams as architecture evolves
• dozens of groups use ESC/Java2 for teaching and research
  • use in teaching
    • Java programming
    • software engineering
    • formal methods
  • examples of external research
    • specification under-definedness checker
    • PVS VC generation
    • Houdini “rebirth”
    • Daikon + ESC/Java2 integration
Mobius PVE Status

• full support available for:
  • all Java and nearly all JML features
    • editing, compilation, doc generation, etc.
  • code complexity and style checking
  • source and bytecode-level static checkers
  • partial BML support
    • no editing of BML or bytecode
  • Mobius VC back-end
  • Mobius Prover back-end
  • interactive proof support for Coq
Mobius PVE Next Steps

- next version to integrate the subsystems:
  - full BML support
  - Universe type inference
  - FreeBoogie and the Race Condition Checker
  - user feedback of proof state in JML/Java
  - proof status and unit/smoke testing
  - Mobius VC generator (in Coq)
  - interactive proof support for PVS
  - Coq PCC certificate generation
  - basic help system and process management
• integrate the Mobius prover interface in the KeY system
  • KeY can use new SMT provers “for free”
  • provers run locally or remotely
• JML to JavaDL bicompiler
  • theory-aware translation to/from a fragment of JavaDL and JML
• an ESC-centric sublogic
  • understand what kinds of VCs are automatically discharged and build new VCgen that targets that sublogic