Symbolic Execution for Chalice

Master's Thesis Description, 14th October 2010

Malte Schwerhoff

Overview

The goal of this master's thesis is the development of a programme verifier based on the concept of symbolic execution [SJP09] for the Chalice programming language [LM09]. Once developed, the verifier is to be gradually extended to eventually become a verifier for the Scala programming language [Ode09]. Consequently, good software engineering is one of the pivotal success criteria of this thesis.

Chalice

Chalice, an experimental object-oriented programming language, incorporates different techniques to address concurrency issues such as deadlocks and race conditions: Implicit dynamic frames [SP09] are used determine an upper bound on the set of heap locations that a certain method might change, whereas fractional permissions [Boy03] are used to control read- and write-access to – possibly shared – heap locations. The current verifier for Chalice utilises Boogie [Bar+05] to verify Chalice programmes encoded in the Boogie programming language [Lei08] Boogie implements a classical weakest-precondition calculus and depends itself on an automatic theorem prover, e.g. Z3 [MB08], to discharge the resulting proof obligations.

Scope

The verifier that is to be developed in the context of this thesis must be able to handle a subset of the Chalice language including fractional permissions and fork-join concurrency.

Since the verifier is to be extended in following projects, it must be designed with modularity and extensibility in mind. In particular, it must be possible to exchange one implementation of the symbolic state for another without having to modify the overall verifier. Likewise, extensions of the verifier should be possible without entailing modifications of the already existing code. The verifier should use the existing Chalice parser, but it should not be strongly coupled to it.

The verifier must be well-documented and thoroughly tested with a reasonable test suite that covers all supported language constructs of Chalice in non-trivial settings. The test suite will include all tests from the already existing test suite for Chalice that cover language constructs also covered by the verifier.

The thesis also includes three presentations (initial, mid-turn, final) and a report describing the theoretical and implementational challenges and accomplishments of the project.

Scope extensions

Depending on the course of the project it might be possible to extend the verifier with channels [LMS10] or to refine the existing permission model. Another possible extension is a deadlock-detection module, as it exists for the current verification-condition-based verifier.

Bibliography

[SJP09] Jan Smans, Bart Jacobs and Frank Piessens. *Symbolic Execution for Implicit Dynamic Frames*. Tech. rep. Katholieke Universiteit Leuven, Belgium, 2009. URL: http://people.cs.kuleuven.be/~jan.smans/oopsla09.pdf.

[LM09] K. Rustan M. Leino and Peter Müller. 'A Basis for Verifying Multi-threaded Programs'. In: *ESOP*. Ed. by Giuseppe Castagna. Vol. 5502. Lecture Notes in Computer Science. Springer, 2009, pp. 378–393. ISBN: 978-3-642-00589-3.

[Ode09] Martin Odersky. *The Scala Language Specification: Version 2.7*. Mar. 2009. URL: http://www.scala-lang.org/docu/files/ScalaReference.pdf.

[SP09] Jan Smans, Bart Jacobs 0002 and Frank Piessens. 'Implicit Dynamic Frames: Combining Dynamic Frames and Separation Logic'. In: *ECOOP*. Ed. by Sophia Drossopoulou. Vol. 5653. Lecture Notes in Computer Science. Springer, 2009, pp. 148–172. ISBN: 978-3-642-03012-3.

[Boy03] John Boyland. 'Checking Interference with Fractional Permissions'. In: *SAS*. Ed. by Radhia Cousot. Vol. 2694. Lecture Notes in Computer Science. Springer, 2003, pp. 55–72. ISBN: 3-540-40325-6.

[Bar+05] Michael Barnett et al. 'Boogie: A Modular Reusable Verifier for Object-Oriented Programs'. In: *FMCO*. Ed. by Frank S. de Boer et al. Vol. 4111. Lecture Notes in Computer Science. Springer, 2005, pp. 364–387. ISBN: 3-540-36749-7.

[Lei08] K. Rustan M. Leino. *This is Boogie 2. (Draft)*. June 2008. URL: http://research.microsoft.com/en-us/um/people/leino/papers/krml178.pdf.

[MB08] Leonardo Mendonça de Moura and Nikolaj Bjørner. 'Z3: An Efficient SMT Solver'. In: *TACAS*. Ed. by C. R. Ramakrishnan and Jakob Rehof. Vol. 4963. Lecture Notes in Computer Science. Springer, 2008, pp. 337–340. ISBN: 978-3-540-78799-0.

[LMS10] K. Rustan M. Leino, Peter Müller and Jan Smans. 'Deadlock-Free Channels and Locks'. In: *ESOP*. Ed. by Andrew D. Gordon. Vol. 6012. Lecture Notes in Computer Science. Springer, 2010, pp. 407–426. ISBN: 978-3-642-11956-9.