A Theorem Proving Approach to Secure Information Flow in Concurrent Programs (Extended Abstract)

Daniel Bruns, Karlsruhe Institute of Technology

We present an approach to formally prove secure information flow in multi-threaded programs. We start with a precise formalization of noninterference in dynamic logic and then use the rely/guarantee approach to reduce this to thread-modular properties, that can be checked locally. A sound and complete calculus ensures that these properties can be proven without false positives. Currently, we work on an implementation in the KeY verification system.

Secure information flow means that no confidential data must be leaked to public sinks. We regard programs at the source code level and consider information sources and sinks to be the global memory. While mature type systems for checking absence of illegal flow are readily available, they suffer from inherent incompleteness w.r.t. security properties defined semantically, such as noninterference. Logic-based approaches, on the other hand, supported by sound and complete calculi, cater for ultimate precision [4], while declassification is provided for free.

Dynamic logic (DL) is a first-order multi-modal logic where any program fragment π gives rise to a modal operator $\langle\pi\rangle$. The formula $\langle\pi\rangle\phi$ is equivalent to the weakest precondition of π w.r.t. ϕ. But in contrast to *wp* construction or Hoare logics, DL formulae are closed under these modal operators and existential quantification. This allows intricate relational properties about programs such as noninterference to be readily expressible [6]. As a consequence, theorem provers can check noninterference without false positives.

Yet, most logics for program verification, such as DL, are fundamentally restricted to sequential programs. The rely/guarantee approach [5] allows sound and complete thread-local reasoning about concurrent behaviors. In previous work, we presented an adaptation of rely/ guarantee that extends dynamic logic conservatively [3]. This allows us to prove noninterference on single threads, which in turn, implies noninterference for the whole concurrent system.

KeY is a verification system for sequential Java [1], based on DL theorem proving. It is largely automated, but the user can resort to interactive proving at any time. A wide range of proof obligations can be generated—e.g., besides functional correctness also for noninterference. As the aforementioned results are orthogonal to specific issues of the Java language, the overall approach does apply to a real-world language as well. An implementation of the rely/guarantee approach is currently underway.

The approach described so far covers the traditional view of noninterference in which outputs are only observable in terminal program states. In the presence of concurrently running threads, it is natural to consider intermediate states as well. This leads to stronger security properties, that are trace-based instead of state-based, such as low-security observational determinism (LSOD). Dynamic Trace Logic [2] is an extension to DL that allows to state temporal properties about program runs, including LSOD.

This work was supported by the German Research Foundation (DFG) under project "DeduSec" within priority programme 1496 "RS^3."

References

[1] W. Ahrendt, et al. The KeY platform for verification and analysis of Java programs. In *Verified Software: Theories, Tools, and Experiments (VSTTE 2014)*. Springer, 2014

[2] B. Beckert, D. Bruns. Dynamic logic with trace semantics. In *24th International Conference on Automated Deduction (CADE-24)*. Springer, 2013

[3] D. Bruns. Deductive verification of concurrent programs. TR 2015-3, Department of Informatics, Karlsruhe Institute of Technology, 2015

[4] A. Darvas, R. Hähnle, D. Sands. A theorem proving approach to analysis of secure information flow. In *Security in Pervasive Computing*. Springer, 2005

[5] C. B. Jones. Tentative steps toward a development method for interfering programs. *ACM Transactions on Programming Languages and Systems*, 5(4), 1983

[6] C. Scheben, P. H. Schmitt. Verification of information flow properties of JAVA programs without approximations. In *Formal Verification of Object-Oriented Software, FoVeOOS 2011*. Springer, 2012