# Runtime Verification of Log Files, a Trojan Horse for Formal Methods[⋆]

Howard Barringer[1], Alex Groce[3], Klaus Havelund[2], David Rydeheard[1], and Margaret Smith[2]

[1] School of Computer Science
University of Manchester
Oxford Road
Manchester, M13 9PL, UK
{howard.barringer,david.rydeheard}@manchester.ac.uk

[2] Jet Propulsion Laboratory
California Institute of Technology
Pasadena, CA 91109, USA
{klaus.havelund,margaret.h.smith}@jpl.nasa.gov

[3] School of Electrical Engineering and Computer Science
Oregon State University
Corvallis, USA
alex.groce@eecs.oregonstate.edu

Runtime verification is the discipline of monitoring and analyzing program executions. A typical scenario consists of determining whether an execution trace satisfies a user-provided specification. Research challenges include development of expressive and convenient specification languages, development of decision procedures for fast analysis of traces against specifications, and minimization of impact on an instrumented running program being monitored. In this presentation, we motivate and show how a temporal rule-based runtime verification system has been applied to log file analysis in support of the software testing effort for NASA's next 2011 Mars mission MSL (Mars Science Laboratory).

RULER [2] is a general-purpose conditional rule-based system, which has a simple and easily implemented algorithm for effective runtime verification, and into which one can compile a wide range of temporal logics and other specification formalisms used for runtime verification. RULER has been designed with expressive power as well as specification convenience in mind. A RULER specification consists of a set of rules operating on a set of facts. A fact is of the form $F(v_1,...,v_n)$, where $F$ is an identifier and $v_1,\ldots,v_n$ are values of various domains. For example, $FileSent(127)$ is a fact. Facts include observed events as well as internally generated *state*. A rule triggers when its condition (a predicate over the set of facts) is satisfied, and as a result the rule will add and/or remove facts from the set. Specifications can be parameterized with data, or even

[⋆] Part of the research described in this publication was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

with specifications, allowing for temporal logic combinators to be defined. The system has been developed in Java and can be directly applied to monitoring JAVA programs, using for example ASPECTJ [5] for code instrumentation.

The LOGSCOPE [4] system is a derivation from RULER, developed specifically for supporting testing of MSL flight software. It is implemented in PYTHON in order to integrate well with other test scripts written for MSL, also written in PYTHON. The LOGSCOPE specification language removes some of RULER's generality, resulting in the interesting subset of data parameterized state machines, and adds a simple user-friendly temporal logic. The temporal logic is mapped to the core parameterized state machines. A description of the process of introducing LOGSCOPE as part of the MSL testing effort is presented in [3]. The system has been used by test engineers to analyze log files generated by running the flight software. The temporal logic was instrumental in achieving test engineer acceptance of the technology. Detailed logging is already part of the MSL system design approach, and hence there is no added instrumentation overhead caused by this approach. While post-mortem log analysis prevents the autonomous reaction to problems possible with *online* runtime verification, it provides a powerful tool for test automation.

A combined presentation of the two systems is presented in [1]. We will conclude with a brief mention of the current effort to unify these two systems, providing more expressive temporal capability in rule specifications.

## References

1. H. Barringer, K. Havelund, D. Rydeheard, and A. Groce. Rule systems for runtime verification: A short tutorial. In S. Bensalem and D. Peled, editors, *9th international workshop on Runtime Verification (RV'09)*, volume 5779 of *LNCS*, pages 1–24, Grenoble, France, July 2009. Springer.
2. H. Barringer, D. Rydeheard, and K. Havelund. Rule systems for run-time monitoring: from Eagle to RuleR. *Journal of Logic and Computation*, 2009. Advance Access published on November 21, 2008. doi:10.1093/logcom/exn076.
3. A. Groce, K. Havelund, and M. Smith. From scripts to specifications, the evolution of a flight software testing effort. October 2009. Submitted for conference publication.
4. A. Groce, K. Havelund, M. Smith, and H. Barringer. Let's look at the logs: Low-impact runtime verification. *Computer Journal*, July 2009. Submitted for review.
5. G. Kiczales, E. Hilsdale, J. Hugunin, M. Kersten, J. Palm, and W. G. Griswold. An overview of AspectJ. In J. L. Knudsen, editor, *European Conference on Object-oriented Programming*, volume 2072 of *LNCS*, pages 327–353. Springer, 2001.