Clara: A Framework for Partially Evaluating Finite-State Runtime Monitors Ahead of Time*

Eric Bodden¹, Patrick Lam², and Laurie Hendren³

¹ Technische Universität Darmstadt, Germany
² University of Waterloo, Ontario, Canada
³ McGill University, Montreal, Quebec, Canada

Abstract. Researchers have developed a number of runtime verification tools that generate runtime monitors in the form of AspectJ aspects. In this work, we present Clara, a novel framework to statically optimize such monitoring aspects with respect to a given program under test. Clara uses a sequence of increasingly precise static analyses to automatically convert a monitoring aspect into a residual runtime monitor. The residual monitor only watches events triggered by program locations that the analyses failed to prove safe at compile time. In two-thirds of the cases in our experiments, the static analysis succeeds on all locations, proving that the program fulfills the stated properties, and completely obviating the need for runtime monitoring. In the remaining cases, the residual runtime monitor is usually much more efficient than a full monitor, yet still captures all property violations at runtime.

1 Introduction

Finite-state properties, also known as typestate [1] properties, constrain the set of acceptable operations on a single object or a group of objects, depending on the object’s or group’s history. Many formalisms allow programmers to easily express typestate properties, including linear temporal logic, regular expressions, message sequence charts and live sequence charts [2, Chapter 2]. Potential applications of runtime monitoring include the evaluation of arbitrary queries over the runtime program state and the enforcement of stated properties. For instance, a monitor could detect attempts to circumvent an access-control policy and then either log the attempt or stop the detected unauthorized access. Researchers have proposed and implemented runtime monitoring tools [3, 4, 5, 6, 7] which compile high-level temporal specifications into monitor implementations.

While runtime monitoring could be useful for finding violations in practice, it is subject to the same problems as software testing. Runtime monitoring gives no static guarantees: a particular program run can only prove the presence of property violations, not their absence. Hence, developers and testers must exercise judgment in deciding when to stop monitoring program runs, since exhaustive testing is generally infeasible. Furthermore, although significant advances have

* This work was supported by NSERC and CASED (www.cased.de)
been made [8,9,10], runtime monitors can still slow down monitored programs significantly, sometimes by several orders of magnitude.

In this paper we therefore propose CLARA, a framework for partially evaluating runtime monitors at compile time. Partial ahead-of-time evaluation addresses all of the problems mentioned above. CLARA specializes a given runtime monitor to a program under test. The result is a residual runtime monitor that only monitors events triggered by program locations that the analyses failed to prove safe at compile time. In our experiments, CLARA’s analyses can prove that the program is free of program locations that could drive the monitor into an error state in 68% of all cases. In these cases, CLARA gives the strong static guarantee that the program can never violate the stated property, eliminating the need for runtime monitoring of that program. In many other cases, the residual runtime monitor will require much less instrumentation than the original monitor, therefore yielding a greatly reduced runtime overhead. In 65% of all cases that showed overhead originally, no overhead remains after applying the analyses.

CLARA’s principal design goal is to provide a maximally general framework for statically analyzing runtime monitors. We anticipate that CLARA will appeal to researchers in runtime verification, as it supports a large variety of runtime monitoring tools. Researchers in static analysis, on the other hand, can easily extend CLARA with novel static analyses to understand and optimize runtime monitors even further. How do we achieve this generality? CLARA’s design is based on the crucial observation that most current runtime-verification tools for Java share two common properties: (1) internally, they use a finite-state-machine model of the property, and (2) they generate runtime monitors in the form of AspectJ aspects [11]. Figure 1 shows a state-machine model for the “ConnectionClosed” property: a disconnected connection should not be written to, unless the connection is potentially reconnected at some later point. Figure 2 shows a monitoring aspect for this property. The remainder of the paper explains this aspect and its analysis in more detail. CLARA takes such monitoring aspects as input and weaves the aspects into the program under test. While weaving, CLARA conducts static analyses, suppressing calls to the monitoring aspect when it can statically prove that these calls are unnecessary.

To perform its static analysis, CLARA must understand the monitoring aspect’s internal transition structure. Because every aspect-generating monitoring tool uses a different code-generation strategy, and we wish to be independent of that strategy, CLARA expects the monitoring aspect to carry an annotation

![Fig. 1. “ConnectionClosed” typestate property: no write after close](image-url)