Clara: Partially Evaluating Runtime Monitors at Compile Time

Eric Bodden

Darmstadt University of Technology
bodden@cs.tu-darmstadt.de

Abstract. In this tutorial, we present Clara, a novel static-analysis framework for partially evaluating finite-state runtime monitors at compile time. Clara uses static typestate analyses to automatically convert any AspectJ monitoring aspect into a residual runtime monitor that only monitors events triggered by program locations that the analyses failed to prove safe. If the static analysis succeeds on all locations, this gives strong static guarantees. If not, the efficient residual runtime monitor is guaranteed to capture property violations at runtime.
Researchers can use Clara with most runtime-monitoring tools that implement monitors through AspectJ aspects. In this tutorial, we will show how to use Clara to partially evaluate monitors generated from tracematches and using JavaMOP. We will explain how Clara's analyses work and which positive effect they can have on the monitor's implementation.

1 Introduction

It is challenging to implement runtime-verification tools that are expressive, nevertheless induce only little runtime overhead. It is now widely accepted that, to be expressive enough, runtime-verification tools must be able to track the monitoring state of different objects or even combinations of objects separately. Maintaining these states at runtime is costly, especially when the program under test executes monitored events frequently.

Even worse, to be reasonably confident that a program does not violate the monitored property, programmers must monitor many different program runs. The more code locations a program contains at which the program may violate the monitored property, the more test cases one may need to execute to appropriately cover all possible execution paths through these code locations. Paired with potentially slow runtime monitors, this goal may be hard if not impractical to achieve.

In this tutorial we therefore teach participants how to use the Clara [2] framework to partially evaluate runtime monitors already at compile time. Partial evaluation brings two main benefits:

1. The partially evaluated monitors usually induce a much smaller runtime overhead than monitors that are fully evaluated at runtime.

2. The partial evaluation can drastically reduce the number of code locations that one needs to consider when looking for code that may cause a property violation. This helps programmers to tell apart useful from useless test cases.

We will also show that often CLARA's partial-evaluation algorithms are powerful enough to prove that a given program can never violate the monitored property. In these cases, monitoring becomes entirely obsolete.

CLARA was designed in such a way that it poses minimal restrictions on the runtime-verification tool that generates the runtime monitor. In general, CLARA works with virtually all tools that generate runtime monitors in the form of AspectJ aspects. In this tutorial, participants will learn how to use CLARA to partially evaluate monitors generated from tracematches [1] and using JavaMOP [3], and even how to partially evaluate hand written monitoring aspects.

We propose the following structure for the tutorial:

1. **Problem Statement:** Why partial evaluation?
2. **What's out there?** Overview of existing runtime-monitoring tools
3. **Architecture of Clara:** Important Design Decisions
4. **Static analyses in Clara:** How do they work? How can we use them?
   - **Quick Check:** syntactic, flow-insensitive
   - **Orphan-Shadows Analysis:** pointer-based, flow-insensitive
   - **Nop-Shadows Analysis:** pointer-based, flow-sensitive
5. **Clara as a generic optimizer:** How to allow your own runtime-verification tool to use CLARA for static optimizations
6. **Clara and testing:** Using CLARA to tell apart useful from useless test cases

During the tutorial, presentations will be mixed with hands-on trainings in which participants can actively use CLARA to experience for themselves which effects CLARA's analyses have. Participants will be able to see for themselves that partially evaluated monitors execute faster and require instrumentation at fewer code locations. Our new Eclipse plugin (see appendix) will allow participants to easily interpret CLARA's analysis results.

References

1. Allan, C., Avgustinov, P., Christensen, A.S., Hendren, L., Kuzins, S., Lhoták, O., de Moor, O., Sereni, D., Sittampalam, G., Tibble, J.: Adding Trace Matching with Free Variables to AspectJ. In: OOPSLA. pp. 345–364. ACM Press (Oct 2005)
2. Bodden, E.: Verifying finite-state properties of large-scale programs. Ph.D. thesis, McGill University (2009)
3. Chen, F., Roşu, G.: MOP: an efficient and generic runtime verification framework. In: OOPSLA. pp. 569–588. ACM Press (Oct 2007)

A Eclipse Plugin

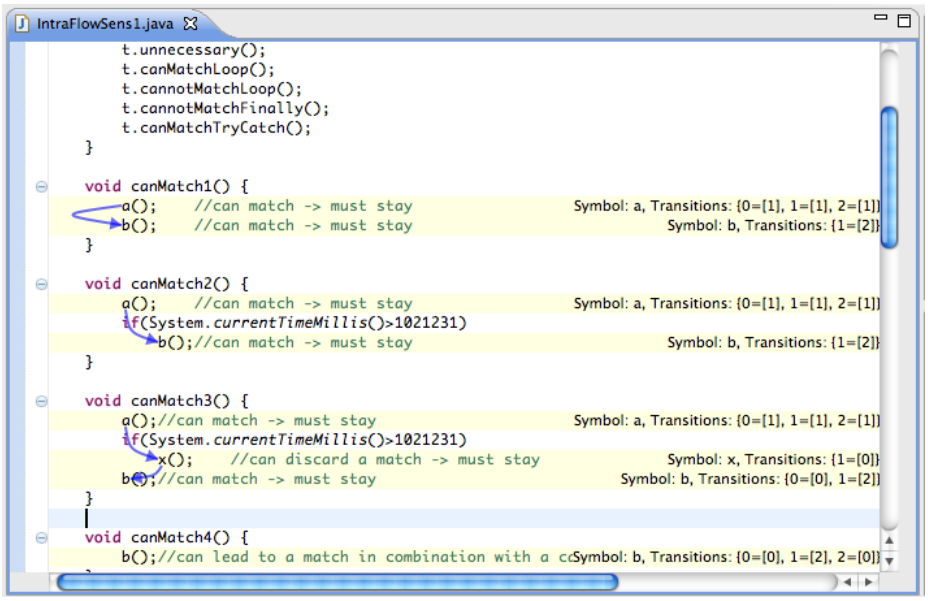

Fig. 1. The screen shot shows the new Eclipse plugin for CLARA. Programmers can use the plugin to easily follow the state transitions of runtime monitors at compile time. Further, the plugin is able to show the effect of CLARA's partial evaluation: by the push of a button one can filter out transitions that the partial evaluation disables.