Applying Language-based Static Verification in an ARM Operating System

Work-in-Progress

Matthew Danish Hongwei Xi Richard West
Boston University
Computer Science Department
111 Cummington Mall
Boston, MA 02215
{md,hwxi,richwest}@cs.bu.edu

1. EXTENDED ABSTRACT

In recent years, we have seen a proliferation of small, embedded, electronic devices controlled by computer processors as powerful as the ARM®. These devices are now responsible for tasks as varied as flying a plane, talking on a cellphone, or helping to perform surgery. Some of these tasks have severe consequences for a mistake caused by faulty programming or missed deadlines. The best defense against these mistakes is to prevent them from happening in the first place.

Advances in programming languages and type theory have lead to the creation of functional programming languages such as ATS [3] which are designed to combine theorem proving with practical systems programming. This allows programmers to bring the rigor of mathematical verification to important properties of their programs. We believe that the usage of languages such as ATS can lead to better assurance that the programs running on an embedded device are correct, responsive, and safe.

A key feature of ATS is that it allows an incremental approach to verification, where some parts of the program are more heavily checked than others, making it easier to make changes and focus proof-writing efforts on the most important portions. Also, the ATS compiler produces C code, it uses native data-layouts, and the ATS language itself is designed to be extremely easy to integrate with existing C code. This makes ATS much better suited than other, similar high-level languages, for systems programming.

Terrier [1] is a new operating system written to run on some of the ARM-based SoC boards from Texas Instruments®: the OMAP3530 and the OMAP4460. These boards are available to developers as the BeagleBoard and the PandaBoard ES respectively. In addition, Terrier supports the Cortex-A9 MPCore symmetric multiprocessor architecture which is used by the OMAP4460. The focus of Terrier is to be a small (currently about 5000 lines of source) and lightweight operating system for real time applications, by using some statically verified algorithms for scheduling and resource management.

Terrier includes an implementation of a Liu and Layland-based [2] rate-monotonic fixed priority scheduler, written in the functional programming style of ATS. The code is annotated with static, dependent and linear types to describe several of the invariants associated with the algorithm. The scheduler written in ATS integrates directly with the rest of the kernel which is written mostly in C. Performance of the scheduler implementation in ATS is essentially equivalent to an implementation in C because the output of the ATS compiler is C code that does not need any additional runtime support.

We are working on extending the rate-monotonic scheduler with stronger proofs and more features. One goal is to show that the algorithm respects some form of temporal isolation. Another goal is to adopt some more modern scheduling algorithms and prove useful properties for them. And finally, we are working on some applications which would potentially test the scheduling capabilities of the system while also demanding strong correctness guarantees.

2. REFERENCES

[1] Matthew Danish. Terrier Homepage. `http://www.github.com/mrd/terrier`.
[2] C. L. Liu and James W. Layland. Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment. *J. ACM*, 20(1):46–61, January 1973.
[3] Hongwei Xi. ATS Homepage. `http://www.ats-lang.org`.