Electronic Communications of the EASST
Volume X (2011)

Proceedings of the 11th International Workshop on Automated Verification of Critical Systems (AVoCS 2011)

Verifying FreeRTOS: from requirements to binary code

Jan Tobias Mühlberg and Leo Freitas

2 pages

Guest Editors: Jens Bendisposto, Cliff Jones, Michael Leuschel, Alexander Romanovsky
Managing Editors: Tiziana Margaria, Julia Padberg, Gabriele Taentzer
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

Verifying FreeRTOS: from requirements to binary code

Jan Tobias Mühlberg[1] **and Leo Freitas**[2]

[1]Dept. of Computer Science, K.U.Leuven, Belgium, jantobias.muehlberg@cs.kuleuven.be
[2]School of Computing Science, Newcastle University, UK, leo.freitas@newcastle.ac.uk

Abstract: This paper reports on ongoing work towards verifying the FreeRTOS real-time operating system kernel. We discuss tools and techniques currently employed and outline future directions of research.

Keywords: Software verification, FreeRTOS, Z, SOCA, VeriFast

1 Introduction

FreeRTOS [Bar11] is an operating system (OS) kernel for embedded real-time applications. It has recently been proposed as case study in the context of the grand challenge on software verification [JOW06]. For this purpose, FreeRTOS is particularly interesting because it is open-source, reasonably small in size, yet relatively complex with respect to the functionality it provides. It features memory management, I/O-device control, tasks management and scheduling, communication and synchronisation directives, and real-time event handling. FreeRTOS has been ported to a range of computing platforms and compilers. The kernel comprises of roughly 3,000 lines of C code with a small fraction of assembly code.

The core of FreeRTOS is its scheduler. It implements different policies for scheduling tasks that share a single processing unit. Being a real-time OS, these policies are not aiming at fair scheduling, but at providing timely responses to events. The scheduler has been subject to recent verification effort [DGM09], where a specification of the task management is proposed.

In this paper we report on ongoing work on the verification of FreeRTOS for structural properties (*e.g.* pointer safety and arithmetic overflow) and liveness properties, ultimately aiming at functional correctness. This includes the reconstruction of a formal specification of FreeRTOS in *Z* [Lin10], bounded model checking of the actual implementation of FreeRTOS with the *SOCA-Verifier* [ML10], as well as annotating the source code with assertions in separation logic to apply the *VeriFast* [JSP10] software verifier.

2 Queues in Z

In [Lin10], an initial formal verification of FreeRTOS scheduler was performed. It included modelling and verification of key data structures using a theorem prover. We took this work forward, and abstracted its main components into simpler structures that were amenable for source-code and binary level verification. This involved refactoring and simplification of the Z model available, as well as proof for feasibility (*i.e.,* preconditions) and well-formedness. This was in preparation for using the SOCA verifier. Our aim is to establish abstract properties of the code within the theorem prover, and then use those as annotations for source-level verification.

The key data structure in the FreeRTOS scheduler is a *Queue* used for scheduling tasks. We modelled this queue (*e.g.,* 10 pages of Z) and proved properties of interest (*e.g.,* 7 theorems and

12 lemmas). For instance, sending items (*e.g.,* inter-task communication among queued tasks) requires knowledge of the task within the scheduler's queue, room for increasing messages from that task, *etc*. Similarly, for receiving items, queue must be known to the scheduler, the task to the queue, and the scheduled messages for that task within the queue must not be empty. These and other properties are specified as predicates proved against the scheduler's *Queue* model.

3 Applying SOCA & VeriFast

With the intention to verify pre- and post-conditions first specified in [DGM09] and improved in [Lin10] at the implementation level, the *SOCA-Verifier* [ML10] was applied to FreeRTOS binary. SOCA is particularly suited for analysing low-level OS components. It implements bounded symbolic execution of compiled and liked program executables. The tool features built-in checks for pointer safety and allows further properties to be specified. To analyse FreeRTOS, we extended the SOCA-Verifier so that programs compiled for ARM processors can be analysed. Using SOCA, we were able to analyse the scheduler functions modelled in [DGM09] with a statement coverage above 85%. The tool did not report any bugs related to pointer safety. Yet, we could not verify further properties with respect to the shape of data structures because these are hard to specify at the object-code level.

VeriFast [JSP10] performs rely-guarantee reasoning for programs written in C. It takes as input the sources of the program, which have to be annotated with method contracts in terms of separation logic and supports specifying and verifying deep data structure properties, such as the safe construction and usage of linked list. Currently, the scheduler and the implementations of data structures like lists and queues in FreeRTOS are being simplified, annotated and verified. In this process, we discover shortcomings in VeriFast that are being fixed to enable the verification of the unmodified source code.

4 Future Work

Beyond the completion of our ongoing research, we will investigate in joining our work on high-level specifications with that on applying implementation-level verification tools through refinement so that code and annotations can be generated from the specifications. A challenge for this will be in the development of a suitable model of pointers. Furthermore, we will research in techniques for specifying and verifying timing properties of FreeRTOS.

Acknowledgements: We thank Jim Woodcock for motivating FreeRTOS to us, and Piyawat Lamsam and Yuhui Lin for their initial input to our work.

References

[Bar11] R. Barry. FreeRTOS. 2011. http://www.freertos.org/.

[DGM09] D. Deharbe, S. Galvao, A. M. Moreira. Formalizing FreeRTOS: First Steps. In *SBMF '09*. LNCS 5902, pp. 101–117. Springer, 2009.

[JOW06] C. Jones, P. O'Hearn, J. Woodcock. Verified Software: A Grand Challenge. *Computer* 39(4):93–95, 2006.

[JSP10] B. Jacobs, J. Smans, F. Piessens. A quick tour of the VeriFast program verifier. In *APLAS 2010*. LNCS 6461, pp. 304–311. Springer, 2010.

[Lin10] Y. Lin. Formal analysis of FreeRTOS. Master's thesis, University of York, 2010.

[ML10] J. T. Mühlberg, G. Lüttgen. Symbolic Object Code Analysis. In *SPIN '10*. LNCS 6349, pp. 4–21. Springer, 2010.