Timed Model-Based Formal Analysis of a Scheduler of Qplus-AIR, an ARINC-653 Compliance RTOS

Sanghyun YOON¹, Dong-Ah LEE¹, Eunjii PAK¹†, Taeho KIM†, Nonmembers, and Junbeom YOO*a Member

SUMMARY Qplus-AIR is a real-time operating system for avionics, and its safety and correctness should be analyzed and guaranteed. We performed model checking a version of Qplus-AIR with the Times model checker and identified one abnormal case that might result in safety-critical situations.

key words: Qplus-AIR, model checking, real-time operating system, safety

1. Introduction

Safety-critical systems should demonstrate functional safety and correctness through formal methods, as recommended by regulation authorities and international standards such as IEC-61508 [1] and DO-178B/C [2]. ‘Qplus-AIR’ [3]–[6] is a recently-developed real-time operating system (RTOS) for avionics, complying with ARINC-653 specification [7], and it is a DO-178B certifiable RTOS.

Scheduling and synchronization services are critical components of RTOS that are being used in safety-critical applications [8]. The applications typically structured in sets of process (i.e., tasks) that share resources via the RTOS. In such contexts, Qplus-AIR should guarantee and demonstrate its correctness for scheduling all tasks on various operating environments and conditions.

This paper reports a formal verification result about ‘Qplus-653,’ which is a kernel core of Qplus-AIR. The formal verification used the model checker, Times [9], to verify the correctness of Qplus-653. By looking into the source code of Qplus-653, we first modeled various tasks of Qplus-653 using timed automata [10] and then performed model checking with Times to check whether the scheduler of Qplus-653 schedules all tasks successfully in accordance with the prioritized preemptive strategy.

Timed automata has been used to verify correctness of real time systems. In order to check whether timing requirements of tasks are met, [11], [12] set deadline miss states of the tasks in the real time system models and conducted reachability analysis for the states using model checkers such as UPPAAL and Times.

The verification of Qplus-AIR showed that the models satisfy all verification properties, except one that means “The scheduler might execute a process even if its deadline was over.” We analyzed the reason for the result and modified the Qplus-653 source code in order to resolve the problem.

The paper is organized as follows: Sect. 2 includes background information about Qplus-AIR and the Times model checker. Section 3 explains how we modeled Qplus-AIR formally using timed automata as well as with assumptions, and Sect. 4 shares the verification results and analysis on the results. Section 5 concludes the paper and gives remarks on future research direction.

2. Background

2.1 Qplus-AIR

‘Qplus-AIR’ [3]–[6] is a real-time operating system for avionics, which has been developed as a part of the SYNDICATE [13] project. It complies with the ‘ARINC-653’ standard [7], which is a software specification for standard interfaces of avionics. It supports two APIs, i.e., APEX (APpli- cation/EXecutive) and POSIX (Portable Operating System Interface for computer environments), between application software and the operating system. Qplus-AIR provides partitioning for the management of application software; it allows independent execution of those applications spatially and temporally.

‘Qplus-653’ is the kernel-core of Qplus-AIR, and it is a certifiable part for DO-178B level A, which is the highest level of the certification by RTCA (Radio Technical Commission for Aeronautics). Qplus-653 provides not only required services of ARINC-653 (e.g., partition and process scheduling, inter- and intra-partition communication, health monitoring, etc.) but also extra services (e.g., shared I/O region and channel synchronization).

2.2 Times

Times [9] is a toolset for modeling, schedulability analysis, and synthesis of executable code. Times supports system specification consisting of three parts: the control automata modeled as a network of timed automata extended with tasks, a task table with information about the processes triggered (released) when the control automata change location (i.e., state), and a scheduling policy. The schedulabil-
ity analysis checks whether a set of tasks can be scheduled successfully according to specific scheduling policies. Furthermore, Times includes the UPPAAL verification engine, and it enables verification of the Times model with model checking and simulation.

The formal verification using Times has three steps in this paper. The first step is the formal modeling of Qplus-653. We focused on modeling of the ‘scheduler’ in order to verify its correctness. The next step models applications, which the scheduler executes. We modeled various combinations of the applications, and they increase the confidence of the verification result because it makes the scheduler operates in a more complicated way. Finally, Times checks the scheduler’s correctness upon the applications.

3. Modeling a Scheduler of Qplus-653

3.1 Scheduling Mechanism of Qplus-653

The scheduler of Qplus-653 supports hierarchical scheduling. A partition often maps to an individual/independent application consisting of several processes (tasks). The scheduler schedules partitions ‘periodically and sequentially.’ A partition has one or more processes, which share time resources within a partition, as illustrated in Fig. 1. Processes in a partition are scheduled with ‘prioritized preemptive strategy.’ When a partition is allocated with resources, the scheduler activates the highest priority process within the partition (e.g., Proc1 in Partition1). Each process has its own priority (P) and deadline (D) as depicted in Fig. 1.

3.2 The Times Models

Table 1 summarizes the partitions and processes, which this paper modeled for the purpose of the model checking of Qplus-653. Partition1 and Partition2 map software applications working on the embedded OS, Qplus-AIR.

The scheduler model schedules partitions and processes cyclically using a clock variable (i.e., sc_time) which represents current time. The model works as follows: first, it checks the deadline of current partition (i.e., currently running partition). If the deadline of the current partition is over, it sets the next deadline of the partition, and the current partition is then changed into the next one. Processes of the new current partition are sorted in a wait queue according to their priorities, and they are transferred to a ready queue. The scheduler also checks the deadline of the current process (Fig. 2(a)). If the deadline is over (i.e., thr_deadline[cur_thr] <= sc_time), it calls the deadline miss handler and elicits a process from the ready queue (Fig. 2(b)). If the ready queue of current partition is not empty (e.g., rdq[0][0]! = −1), the current process is changed to the elected process (i.e., assign_cur_thr in Fig. 2(a)). The model release the current process, then it returns to the initial location and increases the clock variable sc_time, for the convenience of modeling and verification.

4. Formal Verification

We conducted the model checking for the scheduler of Qplus-AIR using Times. The scheduler model has error locations indicating undesirable cases of scheduling. For example, when a periodic process finishing its a period, the model checks deadline miss of the process (Fig. 2(c)). The model transits to the error location deadline_over when the
deadline miss occurs. When the model reaches this error locations, Times generates traces for reachable locations, and we can assume that the model has scheduling errors. We analyzed the traces, the model and source codes to find causes of the error cases. And then, we performed simulation and model checking again with modified version of the scheduler model in order to confirm the analysis results.

4.1 Model Checking with Times

We identified verification requirements from the ARINC-653 specification and the design specification of Qplus-AIR. The verification requirements are translated into temporal logic properties to be used as inputs for Times model checking. The scheduler model passed all of the verification requirements except one. Some of the verification requirements are described as follows:

1. If deadline of the current partition is over, scheduler activates next partition.
2. If the current partition is not in normal state, scheduler shall elect the main thread.
3. A periodic process should not be executed when its deadline is over.

Table 2 shows verification properties translated from the verification requirements. The first and second requirements are translated liveness properties (P1 and P2) for corresponding locations of the requirements, while the last requirement is translated the reachability property (P3) for the error location deadline over in Fig. 2 (c). When the scheduler model should satisfy P1 and P2, besides should not satisfy P3. However, the model checking result shows that the model satisfies P3, that is to say, the model may execute a periodic process even if its deadline is over.

Times generated a trace for P3 as depicted in Fig. 3. The scheduler initializes the system until 30-time unit and thus the wake time of Partition1, Proc1, and Proc2 is 30-time unit. The time capacity for Proc1 and Proc2 is 10-time unit, therefore the processes have the equal deadline (40-time unit) according to Table 1. If Proc1 is executed the until 40-time units, the scheduler model should not execute Proc2, because the deadline of Proc2 is over. Nevertheless, the scheduler executes Proc2 at the time, and the error location of P3 is reachable.

The scheduler model is modeled from Qplus-653 source codes, so we found causes of the error case through analyzing the scheduler model and corresponding source codes. Figure 4 describes pseudo codes related to the error case. When a partition is activated, processes of the partition are sorted in the ready queue according to their priorities. The scheduler calls elect_thread to elect a process to be executed (i.e., current_thread) from the ready queue when the deadline of the current process (e.g., Proc1 in Fig. 3) is over (line 2, Fig. 2 (a)). elect_thread elects and returns a process from the ready queue (e.g., Proc2 in Fig. 3); however it does not check the deadline of the elected process (line 8, Fig. 2 (b)). Therefore, the scheduler executes a periodic process, even if its deadline is over, and it recognizes the deadline miss in the next scheduler call.

We modified the scheduler model and source codes as shown in Fig. 5. The scheduler elects a process from the ready queue (ethread) (line 2). If the deadline of ethread is over, the model calls deadline_miss_handling for the process (line 8). The model checking result with the modified scheduler model shows that the model does not satisfy P3. The Times simulation result also shows that the scheduler model calls deadline_miss_handling when the deadline of ethread is over, and it does not execute the process.
4.2 Discussion

The error case is an unusual case of real-time system scheduling. Time capacities and deadlines of processes typically are defined based on sufficient WCET (Worst Case Execution Time) analysis of the processes. The scheduler model schedules partitions and processes periodically while actual scheduler does periodically and aperiodically. A periodic process calls the scheduler after finishing its period, so the deadline miss time could be smaller than 1-time unit (i.e., a period of scheduler call).

5. Conclusion and Future Work

Qplus-AIR is the real-time operating system for avionics. We performed model checking scheduling part of Qplus-AIR with the Times tool. The scheduler model passed verification properties except one case. We modified source code and model resolved the problem. We are applying other formal verification techniques to verify other parts of Qplus-AIR such as management of memory and health monitoring.

Acknowledgments

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2017R1D1A1B03030065). This work was also partially supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIP) (No. B0101-16-0663, Safety-critical Distributed Modular SW Platform).

References


