Chapter 8
Model Checking of Multitasking Real-Time Applications Based on the Timed Automata Model Using One Clock

Libor Waszniowski
Czech Technical University in Prague, Czech Republic

Zdeněk Hanzálek
Czech Technical University in Prague, Czech Republic

ABSTRACT
The aim of this chapter is to show how a multitasking real-time application running under a real-time operating system can be modeled by timed automata. The application under consideration consists of several preemptive tasks and interrupts service routines that can be synchronized by events and can share resources. A real-time operating system compliant with an OSEK/VDX standard is considered for demonstration. A model checking tool UPPAAL is used to verify time and logical properties of the proposed model. Since the complexity of the model-checking verification exponentially grows with the number of clocks used in a model, the proposed model uses only one clock for measuring execution time of all modeled tasks.

INTRODUCTION
This chapter deals with formal modeling of software applications running under real-time operating system (OS). The typical application under assumption is a complex controller consisting of periodic and aperiodic tasks. Tasks are constrained by deadlines and can be synchronized via inter-task communication primitives. Tasks are assumed to be asynchronous and triggered by external events or by time events. An operating system compliant with the automotive standard OSEK/VDX (OSEK) is considered as an example. The objective is to use an existing model-checking tool for automatic verification of the model described in this chapter.

The proposed model is based on timed automata (Alur and Dill, 1994). It considers an operating system, application tasks and a controlled environment behavior. It assumes a fine-grain model of the task
internal structure consisting of computations, OS calls, selected variables, code branching and loops. Therefore the model combines both, logic and timing characteristics of the discrete event system enabling one to check rather complex properties (safety and bounded liveness properties, state reachability or schedulability) by model checking tool (UPPAAL (Larsen, et al., 1997) in our case) in finite time. Deadlock freeness of the application, occurrence of the race condition during access to shared data structures, a concrete value of some essential variable under certain conditions, end to end response time of an arbitrary event, proper ordering and timing of events in the control application or the controlled environment can be verified, for example.

Due to the composability of timed automata, models of different parts of the system can be combined together (under condition of syntactic and semantic compatibility of these models). For example, a single processor system model can be expanded to a distributed system model by adding a communication layer model. Since many real-time systems are safety-critical, they are often constructed as fault-tolerant systems. The proposed modeling methodology is flexible enough, to allow incorporation of many fault-tolerant methods used in this kind of application.

It is generally known that timed automata and model-checking are susceptible to the state space explosion. This fact restricts the size of verified application to a small size that seems to be unusable in practice. Therefore we try to show in this chapter, how to reach a compromise between a model of reasonable size on one side, and reasonable granularity on the other side. Model of proper granularity allows a detailed formal analysis of real-time properties that can not be made by response time analysis.

When modeling preemption in a multitasking application, it is necessary to stop a clock variable measuring the execution time of a preempted task and remember its value until the task is scheduled again. This can be done in hybrid automata, but not in timed automata. On the other hand, it has been shown in (Henzinger, et al., 1998) that the most general decidable hybrid automaton is the initialized rectangular automaton. The rectangular automata however, are not expressive enough to model preemption. Precise preemption modeling requires a so called stopwatch automata belonging to the undecidable class of hybrid automata (Henzinger, et al., 1998).

For timed automata, however, the reachability problem is decidable and the verification algorithm is therefore guaranteed to terminate. This is a motivation of our work providing a timed automata based over-approximate model of preemptive tasks. The over-approximation of the model means that besides the real behavior of the system, also some additional behavior is modeled. Therefore only properties preserved by this approximation can be verified by a model-checking tool. This does not degrade the usability of the model too much since the most important classes of properties (safety and bounded liveness) are preserved (Berard, et al., 2001).

The complexity of the timed automata based model checking is exponential on the number of clocks and on the largest time constant appearing in the timing constraints. Even though the growing complexity due to time constants can be avoided by the symbolic representation of sets of states, the number of clocks remains a problem determining the feasibility of the model checking in practice (Daws & Yovine 1996). Even though a method for reducing the number of clocks of timed automata has been proposed in (Daws & Yovine 1996), we have minimized the number of clocks already in the phase of the model construction.

This approach is motivated by the fact that only one task can be running on the processor at any given time and only the running task can be finished. Therefore, only the clock of the task currently running can affect the progress of the model. Timed automata of the other tasks can-