Functionally Deterministic Scheduling

Frédéric Boniol*, Claire Pagetti*,**, François Revest**

*IRIT-ENSEEIHT, 2 rue C. Camichel. F31071 Toulouse, France
frederic.boniol@enseeiht.fr,
**ONERA-CERT, 2 av. E. Belin. F31055 Toulouse - France
claire.pagetti,francois.revest@cert.fr

Abstract. The aim of this paper is to prove that the question "*is a scheduling functionally deterministic for a given set of periodic real time tasks and a given conservative scheduling policy*" is decidable. For thar purpose, we encode the tasks and the scheduler by an acyclic stopwatch automaton. We show then that the previous question can be encoded by a decidable reachability problem.

1 Introduction

Embedded systems architecture is classically decomposed into three main parts. The *control software* is often designed by a set of communicating functional modules, also called tasks, usually encoded with a high level programming language (e.g. synchronous language) or a low level one (e.g. Ada or C). Each functional module is characterized by real-time attributes (e.g. period, deadline) and a set of precedence constraints. The *material architecture* organizes hardware resources such as processors or devices. The *scheduler* decides in which order functional modules will be executed so that both precedence and deadline constraints are satisfied.

Behavioral correctness is proved as the result of the logical correctness, demonstrated with the use of formal verification techniques (e.g. theorem proving or model-checking) on the functional part, and the real-time correctness which ensures that all the computations in the system complete within their deadlines. This is a non trivial problem due both to precedence constraints between tasks, and to resource sharing constraints. This problem is addressed by the real-time scheduling theory which proposes a set of dynamic scheduling policies and methods for guaranteeing/proving that a tasks configuration is schedulable.

However, in spite of their mutual dependencies, these two items (functional verification and schedulability) are seldom addressed at the same time: schedulability methods take into account only partial information on functional aspects, and conversely the verification problem of real-time preemptive modules has been shown undecidable (Ermont and Boniol, 2007). To overcome this difficulty, a third property is often required on critical systems, especially for systems under certification: *determinism*, i.e. all computations produce the same results and actions when dealing with the same environment input. The benefit of this property, if ensured, is to limit the combinatorial explosion, allowing an easier abstraction of real-time attributes in the functional view. For instance, preemptive modules may be abstracted by non preemptive ones characterized by fixed beginning and end dates. The interesting consequence