CISTER
Research Centre in
Real-Time & Embedded
Computing Systems

Conference Paper

# SMT-based Schedulability Analysis using RMTL- R

**Andre de Matos Pedro**
**David Pereira**
**Luis Miguel Pinho**
**Jorge Sousa Pinto**

CISTER-TR-161101

# SMT-based Schedulability Analysis using RMTL- R

Andre de Matos Pedro, David Pereira, Luis Miguel Pinho, Jorge Sousa Pinto

*CISTER Research Centre

Polytechnic Institute of Porto (ISEP-IPP)

Rua Dr. António Bernardino de Almeida, 431

4200-072 Porto

Portugal

Tel.: +351.22.8340509, Fax: +351.22.8321159

E-mail:

http://www.cister.isep.ipp.pt

## Abstract

Several methods have been proposed for performingschedulability analysis for both uni-processor and multi-processorreal-time systems. Very few of these works use the power offormal logic to write unambiguous specifications and to allowthe usage of theorem provers for building the proofs of interestwith greater correctness guarantees. In this paper we addressthis challenge by: 1) defining a formal language that allows tospecify periodic resource models; 2) describe a transformationalapproach to reasoning about timing properties of resource modelsby transforming the latter specifications into a SMT problem.

© CISTER Research Center
www.cister.isep.ipp.pt

# SMT-based Schedulability Analysis using RMTL-$\int$

André de Matos Pedro, David Pereira, Luís Miguel Pinho, and Jorge Sousa Pinto*
CISTER/INESC TEC, ISEP, Polytechnic Institute of Porto, Portugal
{anmap,dmrpe,lmp}@isep.ipp.pt
* HASLab/INESC TEC & Universidade do Minho, Portugal
jsp@di.uminho.pt

***Abstract*—Several methods have been proposed for performing schedulability analysis for both uni-processor and multi-processor real-time systems. Very few of these works use the power of formal logic to write unambiguous specifications and to allow the usage of theorem provers for building the proofs of interest with greater correctness guarantees. In this paper we address this challenge by: 1) defining a formal language that allows to specify periodic resource models; 2) describe a transformational approach to reasoning about timing properties of resource models by transforming the latter specifications into a SMT problem.**

## I. Introduction and Motivation

Very few works adopt formal logic as the framework for specifying and reasoning about the scheduling problems at hand. Therefore, specifications may be subject to multiple interpretations, and both the construction and checking of associated proofs becomes error prone. This is not the case when using formal logic, since the syntax and semantics must be defined unambiguously. Practitioners can use modern theorem provers to build machine checkable proofs of the unambiguous specifications that they are interested in showing for the scheduling analysis problem. Furthermore, (timed) temporal logic becomes capable to supply the synthesis algorithms with the scheduling problem that automatically outputs the concrete implementation via the transformation of the specifications into, *e.g.*, finite state machines.

In this paper we focus on the formal treatment of periodic resource models [5] with the goal of analyzing the compositionality of rigorously defined components, each one with its own set of real-time tasks and their associated timing properties. We transform the schedulability problem into a SMT problem in order to integrate the description of the scheduling behavior with the schedulability analysis. This allows to draw counter-examples when the system is not schedulable which can then be used for the system engineers to adapt the design accordingly.

### A. Resource Models

As resource model (RM), we consider a model whose components are of two possible kinds, namely, *simple components* or *supervisor components*. A simple component is denoted by a tuple $C = (\Gamma, \omega, \vartheta, \phi)$ where $\Gamma = \{\tau_1, \ldots, \tau_n\}$ is a set of tasks, $\omega$ is a RM, $\vartheta$ is a scheduler policy, and $\phi$ is a set of properties defined in a *program logic* to monitor the behaviour of $\Gamma$. The supervisor components (or hypervisors) are tuples

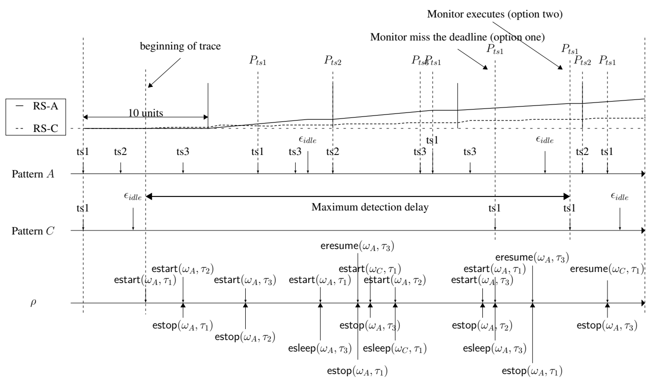
Monitor executes (option two)
Monitor miss the deadline (option one)
beginning of trace
$P_{ts1}$
$P_{ts2}$
$P_{ts3}$
RS-A
RS-C
10 units
$\epsilon_{idle}$
ts1
ts2
ts3
Pattern $A$
Maximum detection delay
Pattern $C$
$\rho$
eresume$(\omega_A, \tau_3)$
estart$(\omega_A, \tau_2)$
estart$(\omega_A, \tau_1)$
estart$(\omega_A, \tau_3)$
estart$(\omega_C, \tau_1)$
eresume$(\omega_C, \tau_1)$
estop$(\omega_A, \tau_1)$
estop$(\omega_A, \tau_2)$
estop$(\omega_A, \tau_3)$
esleep$(\omega_A, \tau_3)$
esleep$(\omega_C, \tau_1)$

Figure 1: Example of patterns and the global trace generated by the composition of resource models.

$\text{H} = (\Omega, \phi_h)$ where $\Omega$ is a set of periodic resource models, and $\phi_h$ is a set of timed properties to check. Having these two kinds of components is justified by the fact that the framework was originally designed to be able to account for the specification and reasoning about runtime monitors as artifacts to check, upon run-time, that the RM behave as specified.

Figure 1 shows an example of a concrete CMF instance. It considers two components RS-A and RS-C. The compontent RS-A consisting of ts1, ts2, and ts3; the component RS-C considers only a task, namely ts1. We can see in the Figure 1 two distinct patterns of execution, according to the task events estart, esleep, eresume, and estop, each denoting a job's status of execution (started, sleeping, resumed, and finished).

### B. Adopted Formal Logic

For this work, we adopt the RMTL-$\int$ logic [3], a fragment of the MTL-$\int$ [2] with a restriction over the relations that can be defined at the term level. RMTL-$\int$ was introduce with the original aim of easing specification of periodic resource models and their verification/enforcement of properties during run-time. The syntax of RMTL-$\int$ is defined in a mutually inductive way. Let $P$ and $V$ denote, respectively, non-empty finite sets of propositions and variables. The terms denoted by $\eta$ are of the form $\alpha \in \mathbb{R}$, $x \in V$, or $\int^{\eta} \varphi$. They correspond respectively, to a real-valued constant, a logic variable, and the duration of the formula $\varphi$. The formulae denoted by $\varphi$ are of the form $p \in P$ (proposition), $\eta_1 < \eta_2$ (relation between terms), $\neg\varphi$ (negation), $\varphi_1 \vee \varphi_2$ (disjunction), $\varphi_1 \, \mathcal{U}_I \, \varphi_2$ (interval-bounded *until*), $\varphi_1 \, \mathcal{S}_I \, \varphi_2$ (interval-bounded *since*), or $\exists x \, \varphi$ (*existential* quantifier).

The semantical interpretation of RMTL-$\int$ formulas is defined elsewhere [3]. The model to interpret the formulas are sets of time-labelled traces produced by a periodic RM. As an example, we can use the RTML-$\int$ formula

$$\int^{10} \mathsf{estop}(\text{RS-A}(ts1)) < 9 \tag{1}$$

to denote that the task $ts1$ belonging to the resource model RS-A must hold in at most 9 time units in any execution trace before time 10 (see the time line Pattern $A$ of the Figure 1).

## II. Specification of Resource Models

In order to allow for non-ambiguous specification of resource models and facilitate the construction of a RMTL-$\int$ formulae that has specifications of these models, we propose a simple language and transformation semantics. This language, named $\mathcal{L}$, has expressions to declare tasks and resource models, together with concurrency relations (higher priority or same priority between tasks and resource models). Let $\tau_1, \ldots, \tau_k$ be task names, $\rho_1, \ldots, \rho_l$ resource model names, $\mathsf{op}_t \in \{\succ, \bowtie\}$ and $\mathsf{op}_m \in \{\|, \gg\}$. The syntax of $\mathcal{L}$ is inductively defined by

$$\begin{aligned} tsk &::= \tau_i(C, P) \mid tsk_1 \ \mathsf{op}_t \ tsk_2 \\ rm &::= \rho_j(tsk, B, P) \mid rm_1 \ \mathsf{op}_m \ rm_2, \end{aligned}$$

where $C$ is a WCET, $tsk$ is a set of tasks, $B$ and $P$ are natural numbers denoting, respectively, a budget and period. The operator $\succ$ represents urgency among tasks, *i.e.*, if $tsk_1 \succ tsk_2$ holds then $tsk_1$ is a task with more urgency than $tsk_2$; the operator $\bowtie$ denotes that two tasks have exactly the same urgency in the system. Similarly, the operator $\gg$ denotes a urgency relation over resource models, and $\|$ denotes concurrent execution between two resource models with the same level of urgency in the system. For instance, a possible RM specification for Figure 1 can be expressed as

$$\text{RC-A}(ts1(10, 8) \succ (ts2(5, 20) \bowtie ts3(7, 27))) \parallel \text{RS-C}(ts1(4, 33)).$$

The next step of our method consists in the transformation of a specification written in $\mathcal{L}$ into an equivalent RMTL-$\int$ specification. We can then check for the satisfiability of a scheduling property over the generated set of formulas, like for instance checking if task $ts1$ in RS-A halts before time 9, again using the Equation 1. Next, we convert this formula into the SMT-LIBv2 language using our tool [4] and delegate the reasoning to the Z3 solver [1]. More complex examples can be seen in the tool's repository [4]. The first experimental results indicate that this method is indeed feasible for small sets of tasks and resource models.

To better exemplify how the process is done, let us assume the Listing 1 that shows an incomplete candidate encoding of the point-wise semantics for the RMTL-$\int$ duration term. The uninterpreted function `compute`$_p$ evaluates a proposition at the instant `mt`, and `p`$_a$ is a proposition representing an event. It is true from the beginning of the event's occurrence until the next event is triggered in the system. Our goal is to find a trace (or set of traces) that satisfies these constraints, henceforth if the answer we obtain is *unsat* then the system is impossible to be scheduled (somehow the constraints may be incoherent); otherwise, we have a flow of the system for which these constraints result in a schedulable behaviour.

```
(define-fun indicator ((mt Time)) Int
  (ite (= (compute_p trace mt p_a) TVTRUE) 1 0)
)

(declare-fun evaln ((Time)) Int)
(assert (= 0 (evaln 0))) (assert (forall ((x Int)) (=> (> x 0) (= (evaln x ) (+ (
        evaln (- x 1)) (indicator x) )))))

(assert (< (evaln 10) 9 ))
```

Listing 1: RMTL-$\int$ duration term encoding using SMT-Libv2.

Comparatively to classic approaches, it is clear that this type of reasoning allows to construct and extend our constraints easily, instead of needing to reformulate every step of the analysis (it is a constructive approach). Note also that the expressiveness to deal with temporal order is of extremely importance when dealing with systems depending on a time, which using just sets of inequalities and equalities alone cannot provide. It is therefore important to reuse such sets of (in-)equalities and combined them with logic connectives to get a fine-grained description of the system. Furthermore, the recent developments of SMT solvers positively impact our approach, namely due to the efficiency of the underlying reasoning methods that increases the chances of constructing the proofs we need in a fully automatic way.

## III. Conclusion and Further Work

In this paper we have described an alternative approach to scheduling analysis following a formal based rigorous specification of the components of the scheduling hierarchy, and its transformation into the SMTLIBv2 language for which we have used the Z3 solver to obtain valid schedules. Our plan in terms of future work is to improve on the developments done so far and on the kind of system we target, in order to understand how the proposal scales for systems which have characteristics very close to those used in the industry.

## IV. Acknowledgments

This work was partially supported by National Funds through FCT/MEC (Portuguese Foundation for Science and Technology) and co-financed by ERDF (European Regional Development Fund) under the PT2020 Partnership, within the CISTER Research Unit (CEC/04234); also by by FCT/MEC and the EU ARTEMIS JU within project ARTEMIS/0001/2013 - JU grant nr. 621429 (EMC2).

## References

[1] L. De Moura and N. Bjørner. Z3: An efficient smt solver. In *TACAS'08/ETAPS'08*, pages 337–340, 2008.

[2] Y. Lakhneche and J. Hooman. Metric temporal logic with durations. *Theor. Comput. Sci.*, 138(1):169–199, 1995.

[3] A. Pedro, D. Pereira, L.M. Pinho, and J.S. Pinto. Logic-based schedulability analysis for compositional hard real-time embedded systems. *SIGBED Review*, 12(1):56–64, 2015.

[4] A. Pedro, D. Pereira, L.M. Pinho, and J.S. Pinto. The rmtld3syth tool. `https://github.com/cistergit/rmtld3synth`, 2016.

[5] I. Shin and I. Lee. Compositional real-time scheduling framework with periodic model. *ACM TECS*, 7(3):30:1–30:39, 2008.