Logic-based Schedulability Analysis for Compositional Hard Real-Time Embedded Systems

André de Matos Pedro
CISTER/INESC TEC, ISEP
Rua Dr. António Bernardino
de Almeida 431, 4200-072
Porto, Portugal
anmap@isep.ipp.pt

David Pereira
CISTER/INESC TEC, ISEP
Rua Dr. António Bernardino
de Almeida 431, 4200-072
Porto, Portugal
dmrpe@isep.ipp.pt

Luís Miguel Pinho
CISTER/INESC TEC, ISEP
Rua Dr. António Bernardino
de Almeida 431, 4200-072
Porto, Portugal
lmp@isep.ipp.pt

Jorge Sousa Pinto
HASLab/INESC TEC, UM
Rua da Universidade
Braga, Portugal
jsp@di.uminho.pt

ABSTRACT
Over the past decades several approaches for schedulability analysis have been proposed for both uni-processor and multi-processor real-time systems. Although different techniques are employed, very little has been put forward in using formal specifications, with the consequent possibility for mis-interpretations or ambiguities in the problem statement. Using a logic based approach to schedulability analysis in the design of hard real-time systems eases the synthesis of correct-by-construction procedures for both static and dynamic verification processes. In this paper we propose a novel approach to schedulability analysis based on a timed temporal logic with time durations. Our approach subsumes classical methods for uni-processor scheduling analysis over compositional resource models by providing the developer with counter-examples, and by ruling out schedules that cause unsafe violations on the system. We also provide an example showing the effectiveness of our proposal.

Categories and Subject Descriptors
C.3 [**Special-Purpose and Application-Based Systems**]: Real-time and embedded systems; D.2.4 [**Software Engineering**]: Software/Program Verification—*Formal methods, Model checking*; F.4.1 [**Mathematical Logic and Formal Languages**]: Mathematical Logic —*Temporal logic*

Keywords
Temporal logic, Schedulability analysis, Compositional, Hard Real-Time Systems, Embedded Systems

1. INTRODUCTION
Schedulability analysis is a very important part of the research that is carried out in real-time systems. Due to the complex nature of the scenarios that real-time systems face, functional properties must be coupled with a predictable response time, so that the operations are performed safely and within the expected constraints. Relaxing any of these two conditions, in the case of hard real-time systems, might lead to catastrophic events, including the loss of human lives.

Along almost forty years, a bewildering diversity of schedulability tests for hard real-time systems has been proposed to address the constrains imposed by the required predictability. These tests vary considerably in their complexity, expressivity, and target scheduling policies (e.g., fixed task or job priority, preemptive or non-preemptive). The literature [1, 12] reveals that generally schedulability test works by assuming a worst-case scenario and checking that each of the involved task gets a sufficient allocation of shared resources or jobs always complete before their deadlines. Naturally, cases that are not "the worst" will also succeed.

In this paper we consider periodic resource models [23], [24] for the composability of components each one with its own set of real-time tasks, providing a rigorous definition of their timing properties, intending to be able to formally verify their composition. These definitions are established by the language and semantics of timed temporal logic, an approach that is not new in the context of real-time systems verification [5]. In this paper we also consider a variant of MTL-$\int$ [16] that is well-suited to analyze sequences and durations of timed executions. This type of analysis is sufficient to solve the schedulability decision problem of periodic resource models, and compositional periodic resource models. The reasons for adopting a logic-based paradigm towards schedulability analysis are: it becomes more comprehensive and expressive; rules out potential specification incoherences typical from informal specifications; and it has some benefits relatively to the available analysis, not in terms of efficiency but in terms of easy extension for monitoring approaches such as the acquisition of the maximum detection delay of a task as in [26]. As further context to the work, we note that:

1. the outcome of a classical schedulability analysis is typically a verdict for a certain set of tasks, but no counter-examples are shown if the set of tasks is not schedulable;

2. the behavior of the scheduler is assumed rather than being explicitly included in the schedulability test;

3. the timing description of the tasks is the unique data provided by classical analysis methods (*i.e.*, offsets, jitters, periods, deadlines);

4. standard approaches are not possible to extend with other usefull properties such as monitoring and enforcement of real-time properties [22, 21], due to the restricted definition of their sets of tasks (*e.g.*, defining a bound for two consecutive instructions, the inter-arrival time of an event);

5. some real-time literature [26, 27] commonly considers the estimation of an arrival rate, which implies minimization and produces significant issues (e.g., under and over estimations, local minimums and maximums, *etc.*).

Our work intends to integrate the description of the scheduling behavior with the schedulability analysis, which enables to draw counter-examples when the system is not schedulable. These counter-examples are fundamental for the system designer to understand and adapt the design accordingly.

Another key point of our approach is that the rigorous of our definitions enable the integration of formal-verification techniques such as runtime verification or model checking, by subsuming the corresponding computational artifacts in a correct-by-construction way. It opens the possibility of adopting mature and experimental formal verification tools [2, 6, 19, 4] that are already available for the scenarios we intent to certify in future development of our work.

Although this paper's focus is solely on the schedulability analysis of compositional periodic resource models under the *rate monotonic* (RM) policy, we introduce this work as a foundational approach for schedulability analysis of compositional resource models, on which we intent to use more advanced schedulability policies and principles in the future. Moreover, this research work is part of a long term project whose aim is the development of novel approach for the unified specification of hard real-time systems (functional and non-functional requirements), supported by the combination of off-the-shelf static verification and runtime verification methods.

We provide a fragment of the *metric temporal logic with durations* (MTL-$\int$), namely *restricted metric temporal logic with durations* (RMTL-$\int$), a schedulability analysis for periodic resource models and coupled periodic resource models as well as the encoding of both models in RMTL-$\int$. Our encoding allows us to isolate by construction cases where the *worst-case execution time* (WCET) violations are unsafe to the schedulability of the system, and to analyze each component knowing only the high-level specifications (instead of the internals) of the other components, but excludes the increment of components at runtime. A synthetic workload is also described to exemplify the schedulability analysis using RMTL-$\int$. For the best of our knowledge this is the first approach that combines MTL-$\int$ with schedulability analysis.

The paper is organized as follows. Section 2 introduces research work that relates to the one presented in this paper. Section 3 introduces the preliminary concepts that are necessary for our schedulability analysis as a background. Section 4 describes the syntax and semantics of the MTL-$\int$ logic, including a set of necessary axioms. Section 5 introduces the new concept of schedulability analysis using MTL-$\int$ and timed execution traces. Section 6 exemplifies how to use runtime monitors through a practical application of the method of schedulability analysis that we propose. Finally, Section 7 draws some conclusions and points to further work directions.

2. RELATED WORK

So far, not many alternative approaches for schedulability analysis of real-time systems have been proposed nor specific formalisms for tests have emerged. Here, we describe alternatives for schedulability analysis based on *timed automata* [9, 14], *Petri nets* [25], and *process algebraic* [20].

2.1 Automata-based

In the last decade, some efforts have been done to use timed automaton as a model to check the schedulability of a system. Fersman et al. [9, 10] proposes the *timed automata extended with real-time tasks* to specify real-time systems as timed automata but assuming an implicit scheduler. Intuitively, the process consists by modeling the scheduling behavior (e.g., following the RM, or EDF policies) as a timed automaton and couple it with the model of the system as another timed automaton. The reachability analysis is performed to decide if the multiplication of both automata does not allows the model to reach the error state defined in the scheduler automaton. Since, the schedulability test remains a reachability analysis problem, we can solve it with model checker tools such as UPPAAL [2] and NuSMV [6].

2.1.1 Decidable results of Task Automata

The problem of checking schedulability of a task automaton is undecidable [11, 15]. Recently, some progress has been made to show that a significant fragment of task automata is decidable. Yi et al. [9] proved that the problem of checking schedulability relative to a non-preemptive scheduling strategy for task automata is decidable, and more generally proved that the problem of checking schedulability is decidable for task automata without task feedback (i.e., the precise finishing time of a task cannot influence the new task releases) or with fixed computation times (i.e., the best case execution time is not different from the worst-case execution time). Indeed, the schedulability problem for a single-processor system is undecidable over these assumptions but a open question still remains for decidable results of preemptive schedulers when the computation times of tasks may vary within a known interval [14].

For multi-processor systems, the problem is also undecidable [14]. Krcal et al. [14] proved that the schedulability of multi-processor system is decidable for non-preemptive schedulers (as for uni-processor setting) or using tasks with constant execution times.

2.2 Other Approaches

Timed Petri nets may employ the same decision method of automata-based approaches for schedulability analysis of real-time systems. Tsai et al. [25] present timing constraint Petri nets as a model to specify real-time systems, and decide its schedulability using reachability analysis of states, where the timing and behavioral properties should be formalized in different levels of abstraction. Zonghua and Shin [13] describes a translation from *Timed petri net* (TPN) to timed automata.

The schedulability analysis of real-time systems with aid of process algebraic was initially proposed by Ben-Abdallah et al. [3]. Philippou et al [20] formalizes the problem of compositional hierarchical scheduling by introducing a process algebraic framework for modeling resource demand and supply, which was inspired in the *timed process algebra*.

3. PRELIMINARIES

In this section we introduce the main concepts that support our formalization of the schedulability test for periodic resource models.

3.1 Basic Notions

In the rest of the paper, we will assume *tasks sets*

$$\Gamma = \{\tau_1, \tau_2, ..., \tau_n\},$$

such that $n \in \mathbb{N}^+$ is the identifier of periodic tasks, and $\tau_i = (p_i, e_i)$ with p_i and e_i are, respectively, the period and the worst-case execution time of the periodic task τ_i; and a set of *periodic resource model* $\Omega = \{\omega_1, \omega_2, ..., \omega_m\}$ with

$$\omega_j = (\tau, \pi, \theta, rm),$$

where $\tau \subseteq \Gamma$, π is the *replenishment period*, θ is the *server budget*, and rm is the RM scheduling policy.

The outputs of a resource model ω are *sequences of events*. Considering a par (ω, τ_i) with $\omega \in \Omega$ and $\tau_i \in \tau$, each event can be of one of the following types: a *release-event* $\mathsf{erelease}(\omega, \tau_i)$; a *start-event* $\mathsf{estart}(\omega, \tau_i)$; a *sleep-event* $\mathsf{esleep}(\omega, \tau_i)$; a *resume-event* $\mathsf{eresume}(\omega, \tau_i)$; or a *stop-event* $\mathsf{estop}(\omega, \tau_i)$. In addition, we assume a parameterized event $\varepsilon(\omega_j, \tau_i, id)$ that denotes the critical events of a task, where id is the event identifier, and $\mathsf{erenewal}(\omega)$ denotes the budget release of a resource model. We denote sets of events by $\mathcal{E}$.

A sequence of events, also known as *execution trace*, is an infinite sequence

$$\rho = (e_1, t_1)(e_2, t_2) \cdots$$

of time-stamped events (e_i, t_i) with $e_i \in \mathcal{E}$ and $t_i \in \mathbb{R}^+$. The sequence satisfies monotonicity and progresses, *i.e.*, $t_i \leq t_{i+1}$ for all $i \in \mathbb{N}^+$, and for all $t \in \mathbb{R}^+$ there is some $i > 0$ such that $t_i > t$, respectively.

3.2 Schedulability Analysis of Periodic Resource Models

The schedulability analysis for periodic resource models is provided by Shin and Lee [23, 24]. The authors formulate an analysis based on *resource model supply*. The *supply bound function* $sbf_\omega(t)$ is defined to calculate the minimum resource supply for every interval of length t as follows:

$$sbf_\omega(t) = \begin{cases} t - (k+1)(p-e) & \text{if } t \in \mathcal{I}, \\ (k-1)e & \text{otherwise}, \end{cases}$$

where $\mathcal{I} = [(k+1)p - 2e, (k+1)p - e]$. The value k is given by

$$k = \begin{cases} x & \text{if } x > 1 \\ 1 & \text{otherwise} \end{cases},$$

where $x = \left\lceil \frac{t-(p-e)}{p} \right\rceil$.

Lehoczky et al. [17] proposed a demand-bound function $dbf_{RM}(\tau, t, i)$ for RM that computes the worst-case cumulative response demand of a task $\tau_i \in \tau$ for any interval of length t. It is defined by

$$dbf_{RM}(\tau, t, i) = \sum_{\tau_k \in \gamma_\tau(i)} \left\lceil \frac{t}{p_k} \right\rceil . e_k,$$

where $\gamma_\tau(i) = \{\tau_1, ..., \tau_i\}$ is a function that returns a set of tasks with higher-priority (and including) than task τ_i. The *demand-bound function* for resource models is the same since the set of tasks is schedulable using the RM policy. This means that the supply of a resource model shall be greater than the demand of the set of tasks that a resource model contains.

The tasks set τ of a resource model is said schedulable according to a RM policy if, and only if,

$$\forall \tau_i \in \tau,\ \exists t_i \in [0, p_i] \text{ s.t. } dbf_{RM}(\tau, t_i, i) \leq sbf_\omega(t_i).$$

This approach is the state of the art on schedulability analysis for periodic resource models. We will subsume this approach with one based on timed temporal logics. Our approach allows to ensure response time guarantees about the composition with runtime monitors without employing great efforts to find more adequate optimization techniques to find the schedulability answer.

4. RESTRICTED METRIC TEMPORAL LOGIC WITH DURATIONS

In this section we introduce the RMTL-$\int$, a fragment of MTL-$\int$ [16] where the evaluation is carried out with respect to sequences of events produced by resource models.

The main motivation for proposing RMTL-$\int$ comes from the fact that restricting some terms and relations over terms, the logic is suitable for generation of runtime monitors as well as to thinking statically over real-time constraints. The main difference between RMTL-$\int$ and MTL-$\int$ is that the former uses only the relation $\leq$, $<$, and $=$ over terms, and excludes functions from the language of terms. This restriction allows us to turn our logic terms always Riemann integrable.

Language of RTML-$\int$ terms		
δ	$::=$	$\alpha \mid x \mid \int^{\delta} \varphi$
Language of RTML-$\int$ formulae		
φ	$::=$	$p \mid \delta_1 \sim \delta_2 \mid \varphi_1 \vee \varphi_2 \mid \neg\varphi \mid \varphi_1 \, U_{\sim\rho} \, \varphi_2 \mid \varphi_1 \, S_{\sim\rho} \, \varphi_2 \mid \exists x \, \varphi$

Table 1: Syntax of RMTL-$\int$

Definition 1 (RMTL-$\int$). *Let $\mathcal{P}$ be a set of propositions and $\mathcal{V}$ a set of logical variables (interpreted over $\mathbb{R}$). The syntax of RMTL-$\int$ is inductively defined according to the rules depicted in Table 1, where δ are terms, $\int^{\delta} \varphi$ is the duration of the formula φ in the interval $[0, \delta]$, $x \in \mathcal{V}$, $p \in \mathcal{P}$, $\rho \in \mathbb{R}_{\geq 0}$, $\sim \in \{<, \leq, =\}$, and $\alpha \in \mathbb{R}$.*

We are now able to define the semantic of the MTL-$\int$. The semantic of MTL-$\int$ is separated in two parts: *terms* and *formulas*. The semantic of terms is defined using the notation $\mathcal{T}[\![\tau]\!](\sigma, \vartheta)t$ in Table 2. All terms represent numerical values in $\mathbb{R}_0^+$. The term $\int^{\delta} \varphi$ is the integral over the Boolean function $B_{\varphi(\sigma,\vartheta)}(t)$ (whose return value is 1 if $(\sigma, \vartheta, t) \models \varphi$, and 0 otherwise). Since $B_{\varphi(\sigma,\vartheta)}(t)$ behaves as a step function, it is always Riemann integrable. The same is not true in the MTL-$\int$ logic. The semantic of the MTL-$\int$ formula is defined inductively in Table 2, where the satisfability of a formula φ in a model (σ, ϑ) at time t is defined by $(\sigma, \vartheta, t) \models \varphi$.

Along the remaining of the paper we will frequently refer to the abbreviations presented in Table 3 in order to ease the presentation of specific schedulability related specification. For illustrative purposes, we now introduce a pratical example of the expressive power of RMTL-$\int$'s language.

Example 1. *To ensure that a task responds in a bounded response time, the formula $\psi_1 \implies \Diamond_{\leq\alpha} \psi_2$ is sufficient. The proposition ψ_1 describes a set of events that may violate the system, the proposition ψ_2 describes the task invocation, and α is the maximum expected response time bound. Informally, the formula means that if a fault event occurs, then the task executes within α time units.*

Operator	Abbreviation	Equivalent Formula
Eventually	$\Diamond_{\sim\alpha}\varphi$	$true \; U_{\sim\alpha} \; \varphi$
Always	$\Box_{\sim\alpha}\varphi$	$\neg(\Diamond_{\sim\alpha}\neg\varphi)$
Next	$\bigcirc_{\varphi_1}\varphi_2$	$\varphi_1 \; U_{\sim\infty} \; \varphi_2$
Implies Next	$\varphi_1 \ominus \varphi_2$	$\neg\varphi_1 \vee \bigcirc_{\varphi_1}\varphi_2$

Table 3: Syntactic abbreviations for RMTL-$\int$

5. SCHEDULABILITY ANALYSIS USING MTL-$\int$

Our schedulability analysis consists in the evaluation of a logic formula over a trace (or a set of traces) produced by a periodic resource model. Regarding these our approach remains a model-checking problem [7], where the model checks a set of logic properties, and otherwise generates counter-examples.

In order to decrease the state space search we can assume for uni-processor systems the critical instant theorem [18]. This assumption reduces our problem to just one trace acceptance for a set of logic properties. This assumptions allows us to identify the relevant traces and combine our approach with the foundational real-time systems theory.

We will describe the encoding of the schedulability test for periodic resource models [23, 24] as well as their composition using our MTL-$\int$ fragment.

5.1 Encoding Notations

To ease the encoding of schedulability analysis properties, we first introduce some syntactical notations and formulae abbreviations. Let ω be a resource model in Ω and let τ_i be a task in τ. The set of tasks with higher-priority (and including) than τ_i for ω is denoted by $\gamma_{\omega}^{\tau_i}$. The set of resource models with higher-priority (and excluding) than ω is denoted by γ_{Ω}^{ω}. For events, we have the following notations: $\varepsilon(\omega, \cdot)$ denotes the set of events that can be generated by the resource model ω; $evs^{+}(\omega_j, \tau_i)$ is defined by

$$evs(\omega_j, \tau_i) \vee \mathsf{estop}(\omega_j, \tau_i) \vee \varepsilon(\omega_j, \tau_i, \cdot) \vee \mathsf{erelease}(\omega_j, \tau_i),$$

with $evs(\omega_j, \tau_i)$ defined by

$$\mathsf{estart}(\omega_j, \tau_i) \vee \mathsf{eresume}(\omega_j, \tau_i) \vee \mathsf{erenewal}(\omega),$$

which specifies all events that a task τ_i in the resource model ω_j can trigger; $evs^{-}(\omega_j, \tau_i)$ denotes the formula resulting from the removal of the $\mathsf{erelease}(\omega_j, \tau_i)$ and $\mathsf{estop}(\omega_j, \tau_i)$ events from $evs^{+}(\omega_j, \tau_i)$; and $evs^{*}(\omega_j, \tau_i)$ denotes the formula resulting from the removal of the $\mathsf{estart}(\omega_j, \tau_i)$ and $\mathsf{estop}(\omega_j, \tau_i)$ events from $evs^{+}(\omega_j, \tau_i)$.

For event occurrences we establish a MTL-$\int$ formula that specifies the exact number of times that a proposition p holds through the following recurrent function:

$$\mathsf{occur}(p, n) \stackrel{def}{=} \begin{cases} \Box\neg e & \text{if } n = 0 \\ \neg e \, U \, (e \, U \, \mathsf{occur}(p, (n-1))) & \text{otherwise} \end{cases},$$

where $p \in \mathcal{P}$ and $n \in \mathbb{N}$ is the number of occurrences to check. Furthermore, we also introduce the definition of a function that restricts occur in the sense that it captures the period for the event under consideration. Such function is the following:

$$\mu(e, t_e, p_e, 0) \stackrel{def}{=} \Box\neg e,$$

$$\mu(e, t_e, p_e, (n+1)) \stackrel{def}{=} \neg e \, U_{=(p_e - t_e)} \, (e \, U_{\leq t_e} \, o(e, t_e, p_e, n)),$$

where t_e is the time that event e consumes, and p_e the period of the event e. This definition allows us to restrict the

Evaluation of RMTL-$\int$ terms		
$\mathcal{T}[\![\alpha]\!](\sigma,\vartheta)t$	$=$	α
$\mathcal{T}[\![x]\!](\sigma,\vartheta)t$	$=$	$\vartheta(x)$
$\mathcal{T}[\![\int^{\delta}\varphi]\!](\sigma,\vartheta)t$	$=$	$\begin{cases}\int_t^{t+\mathcal{T}[\![\delta]\!](\sigma,\vartheta)t} B_{\varphi(\sigma,\vartheta)}(t_*)\ dt_* \text{ if } \mathcal{T}[\![\delta]\!](\sigma,\vartheta)t \geq 0 \\ 0 \text{ otherwise}\end{cases}$
Evaluation of RMTL-$\int$ formulas		
(σ,ϑ,t)	$\models$	p iff $\sigma(p)(t) = true$ and $t < \lvert\sigma\rvert$
(σ,ϑ,t)	$\models$	$\delta_1 \sim \delta_2$ iff $\mathcal{T}[\![\delta_1]\!](\sigma,\vartheta)t \sim \mathcal{T}[\![\delta_2]\!](\sigma,\vartheta)t$
(σ,ϑ,t)	$\models$	$\varphi_1 \vee \varphi_2$ iff $(\sigma,\vartheta,t) \models \varphi_1$ or $(\sigma,\vartheta,t) \models \varphi_2$
(σ,ϑ,t)	$\models$	$\neg\varphi$ iff $(\sigma,\vartheta,t) \not\models \varphi$
(σ,ϑ,t)	$\models$	$\varphi_1\ U_{\sim\rho}\ \varphi_2$ iff $\exists t' \in \mathbb{R}_{\geq 0}$ st. $t \leq t' \sim t+\rho \wedge (\sigma,\vartheta,t') \models \varphi_2$, and st. $\forall t'' \in \mathbb{R}_{\geq 0}$, $t \leq t'' < t', (\sigma,\vartheta,t'') \models \varphi_1$
(σ,ϑ,t)	$\models$	$\varphi_1\ S_{\sim\rho}\ \varphi_2$ iff $\exists t' \in \mathbb{R}_{\geq 0}$ st. $t-\rho \sim t' \leq t \wedge (\sigma,\vartheta,j) \models \varphi_2$, and $\forall t'' \in \mathbb{R}_{\geq 0}$, $t' < t'' \leq t, (\sigma,\vartheta,t'') \models \varphi_1$
(σ,ϑ,t)	$\models$	$\exists x\,\varphi$ iff $\exists v \in \mathbb{R}$ st. $(\sigma,\vartheta_x^v,t) \models \varphi$

Table 2: Semantic of RMTL-$\int$

occurrence of e by a the period p_e. In the following we introduce an example to give, using MTL-$\int$, the occurrences of a certain event in a trace.

EXAMPLE 2. *Suppose that we require to minimize the parameter z of the following formula*

$$\Diamond_{\leq\alpha}\ \mathsf{occur}(\varepsilon(\omega,\tau_i,\cdot),z).$$

Informally, the formula indicates that there exists at most z occurrences of $\varepsilon(\omega,\tau_i,\cdot)$ until α time units. The maximal value to which z can be assigned is the positive infinity (∞). We select this maximum for z as the initial point, and decrease successively the variable z until a the formula holds or zero is achieved. This allows to find a value for z in the interval $[0,\infty)$ and to obtain the exact number of events that a trace contains. Note that this is not trivially solved, and some assumptions about the trace must be made, such as the number of events that are required to minimize z in practice (e.g., ∞ is replaced by $|\rho|$, the length of trace ρ).

5.2 Encoding Periodic Resource Models

Schedulability analysis over the language of RMTL-$\int$ is divided in two parts: the encoding of the scheduler's behavior – including their scheduling policy and workload parameters – and the consequent schedulability test. With both parts holding, we are able to evaluate if a given set of workload parameters is enough to be schedulable over a certain scheduler policy. We begin by detailing the encoding phase and the schedulability test.

The behavior of the scheduler is specified by several formulas within capture the budget supply, the schedulability policy, the task durations, and some intrinsic settings of the scheduler. Assuming a correct release of events, the budget supply is specified by the formula

$$\phi(\omega) \equiv \Box_{\leq\infty}\ (\mathsf{erenewal}(\omega) \ominus rp(\omega)),$$

where

$$rp(\omega) \equiv (\Diamond_{=\pi}\ \mathsf{erenewal}(\omega)) \wedge \int^{\pi} \bigvee_{\tau_i\in\tau} evs^{+}(\omega,\tau_i) \leq \theta,$$

ω is one resource model, π and θ their renewal period and budget, and $\mathsf{erenewal}(\omega)$ is the budget renewal event. This formula states that for each occurrence of the event $\mathsf{erenewal}(\omega)$ in the resource model ω, the duration of the other events until π time units does not overpasses the budget θ per period π.

For the partial order of the task releases we introduce the MTL-$\int$ formula

$$\eta(\omega) \equiv \Box_{\leq\infty} \bigwedge_{\tau_i\in\tau} (\mathsf{erelease}(\omega,\tau_i) \ominus sq(\omega,\tau_i)),$$

where

$$sq(\omega,\tau_i) \equiv ev(\omega,\tau_i)\ U_{\leq p_i}\ \mathsf{estop}(\omega,\tau_i),$$

$$ev(\omega,\tau_i) \equiv \left(\bigvee_{\tau_k\in\gamma_\omega^{(\tau_{i-1})}} evs^{+}(\omega,\tau_k)\right) \vee evs^{-}(\omega,\tau_i)$$

and $\gamma_{\omega_j}^{(\tau_{i-1})}$ denotes the set of higher-priority tasks, excluding events triggered by the task τ_i. This formula means that for every event $\mathsf{erelease}(\omega,\tau_i)$ there is always an event $\mathsf{estop}(\omega,\tau_i)$, and that the events occuring before $\mathsf{estop}(\omega,\tau_i)$ should be any event from τ_i's higher-priority tasks.

The duration of tasks allocated to one resource model is specified by the formula

$$\psi^{\leq}(\omega) \equiv \Box_{\leq\infty} \bigwedge_{\tau_i\in\tau} \left(\mathsf{erelease}(\omega,\tau_i) \ominus du^{\leq}(\omega,\tau_i)\right),$$

where

$$du^{\leq}(\omega,\tau_i) \equiv \int^{p_i} \bigvee_{\tau_k\in\gamma_\omega^{(\tau_i)}} evs^{+}(\omega,\tau_k) \leq\ e_i.$$

Note that the $\leq$ operator could be changed to $\geq$ in order to specify the absolute WCET of the task set. We denote the duration of a task by the $\geq$ operator as $\psi^{\geq}(\omega)$.

In order for our formalization to work, we still specify some other features such as the precedence of the event $\mathsf{estop}(\omega,\tau_i)$ (*i.e.*, each event $\mathsf{estart}(\omega,\tau_i)$ is always followed by an event $\mathsf{estop}(\omega,\tau_i)$, and vice-versa), the number of release events, and the time period at which the release of events its triggered. The precedence of the event $\mathsf{estop}(\omega,\tau_i)$ is specified by the formula

$$\xi^1(\omega) \equiv \bigwedge_{\tau_i \in \tau} \left(\mathsf{estop}(\omega, \tau_i) \ominus pr(\omega, \tau_i)\right),$$

where

$$pr(\omega, \tau_i) \equiv es(\omega, \tau_i)\ S_{\le p_i}\ \mathsf{estart}(\omega, \tau_i),$$

and

$$es(\omega, \tau_i) \equiv \left(\bigvee_{\tau_k \in \gamma_\omega^{(\tau_{i-1})}} evs^+(\omega, \tau_k) \right) \vee evs^*(\omega, \tau_i).$$

The release of events is captured by the recursive function $\mu(e, t_e, p_e, n)$. To ensure the periodicity of all events ($\mathsf{erelease}$ and r^*) for certain t time units, we introduce the formula

$$\xi^2(\omega, t) \equiv oc(\omega, t) \wedge \mu\left(\mathsf{erenewal}(\omega), \pi, 0, \left\lfloor \frac{t}{\pi} \right\rfloor\right),$$

where

$$oc(\omega, t) \equiv \bigwedge_{\tau_i \in \tau} \mu\left(\mathsf{erelease}(\omega, \tau_i), p_i, 0, \left\lfloor \frac{t}{p_i} \right\rfloor\right),$$

$\left\lfloor \frac{t}{\pi} \right\rfloor$ is the number of occurrences of $\mathsf{erenewal}(\omega)$ in t, $\left\lfloor \frac{t}{p_i} \right\rfloor$ is the number of occurrences of $\mathsf{erelease}(\omega, \tau_i)$ in t, π is the period of the budget renewal of the resource model ω, and p_i is the period of the task τ_i. Note that this formula is able to specify the number of events that can be released in t units of time for a task or a periodic resource model.

The encoding of the periodic resource model is given by

$$\mathsf{PRM}(\omega, t) \equiv \phi(\omega) \wedge \eta(\omega) \wedge \psi^{\ge}(\omega) \wedge \xi^1(\omega) \wedge \xi^2(\omega, t),$$

where ω is defined according to certain parameters and a workload, which allows us to unroll the sub-formulas. This concludes the formalization of the periodic resource model's behavior in RMTL-$\int$.

5.3 Encoding Coupled Periodic Resource Models

Here we propose an encoding of coupled periodic resource models and an analysis that ensures non-interference, and avoids priority inversion between resource models due to WCET violations.

The budgets that each resource model is allowed to use is specified by the formula

$$\phi_\Omega(t) \equiv \bigwedge_{\omega \in \Omega} \left(\phi(\omega) \wedge \mu\left(\mathsf{erenewal}(\omega), \pi, 0, \left\lfloor \frac{t}{\pi} \right\rfloor\right)\right).$$

With this formula, each periodic resource model meets the settings assigned to it for a given time t.

Other two definitions need to be formulated. One describes the fixed priority behavior of the periodic resource models, and the other describes the execution time allowed to a given set of resource models and their respective task sets.

The partial order of the task events for a set of resource models Ω is specified by the formula

$$\eta(\Omega) \equiv \Box_{\le \infty} \bigwedge_{\tau_i \in \tau} \bigwedge_{\omega \in \Omega} \left(\mathsf{erelease}(\omega, \tau_i) \ominus po(\Omega, \omega, \tau_i)\right),$$

where

$$po(\Omega, \omega, \tau_i) \equiv \left(re(\Omega, \omega, \tau_i) \vee evs^-(\omega, \tau_i)\right)\ U_{\le p_i}\ \mathsf{estop}(\omega, \tau_i),$$

and

$$re(\Omega, \omega, \tau_i) \equiv \bigvee_{\omega \in \gamma_\Omega^\omega} \bigvee_{\tau_j \in \tau} evs^+(\omega, \tau_j) \vee \bigvee_{\tau_k \in \gamma_\omega^{(\tau_{i-1})}} evs^+(\omega, \tau_k).$$

The formula $re(\Omega, \omega, \tau_i)$ describes the resource events that can occur befor an event $\mathsf{estop}(\omega, \tau_i)$ event.

The execution time allowed for a set of resource models Ω is defined by

$$\psi_\Omega^{\le}(t) = \bigwedge_{\omega \in \Omega} \left(\psi^{\le}(\omega) \wedge oc(\omega, t)\right).$$

To specify the worst-case we can use $\psi^{\ge}(\omega)$ instead of $\psi^{\le}(\omega)$. Substituting the above formula we have $\psi_\Omega^{\ge}(t)$.

The composition of periodic resource models is encoded by the RMTL-$\int$ formula

$$CPRM(\Omega, t) \equiv \phi_\Omega(t) \wedge \eta(\Omega) \wedge \psi_\Omega^{\le}(t) \wedge \left(\bigwedge_{\omega \in \Omega} \xi^1(\omega)\right).$$

5.4 Schedulability Test

To provide schedulability tests for our encodings we need to find a model that satisfies the PRM formula for resource models, and the $CPRM$ formula for a composition of resource models. By the semantic definition of RMTL-$\int$ we need to find an observation, a logical environment, and a duration in accordance with the specified properties. This behavior is formulated by the following definitions.

Definition 2. *Let ω be a resource model in Ω. The resource model ω is schedulable if and only if, there exists a trace ρ of duration t such that $PRM(\omega, t)$ is satisfied, and the duration of ρ is greater or equal to the maximum value of p_i in τ.*

Definition 3. *Let Ω be a set of resource models, and ω a resource model in Ω. The composition of resource models Ω is schedulable if and only if, there exists a trace ρ with duration t such that $CPRM(\Omega, t)$ holds and t of trace ρ is greater or equal than the maximum $p_i \in \tau$, of all resource models in ω.*

Summarizing our definitions states that our schedulability decision problem is a satisfiability problem of a trace regarding a RMTL-$\int$ formula.

5.5 Feasible Tests

Enconding the schedulability test is not enough to ensure that we always obtain a positive or negative answer. In

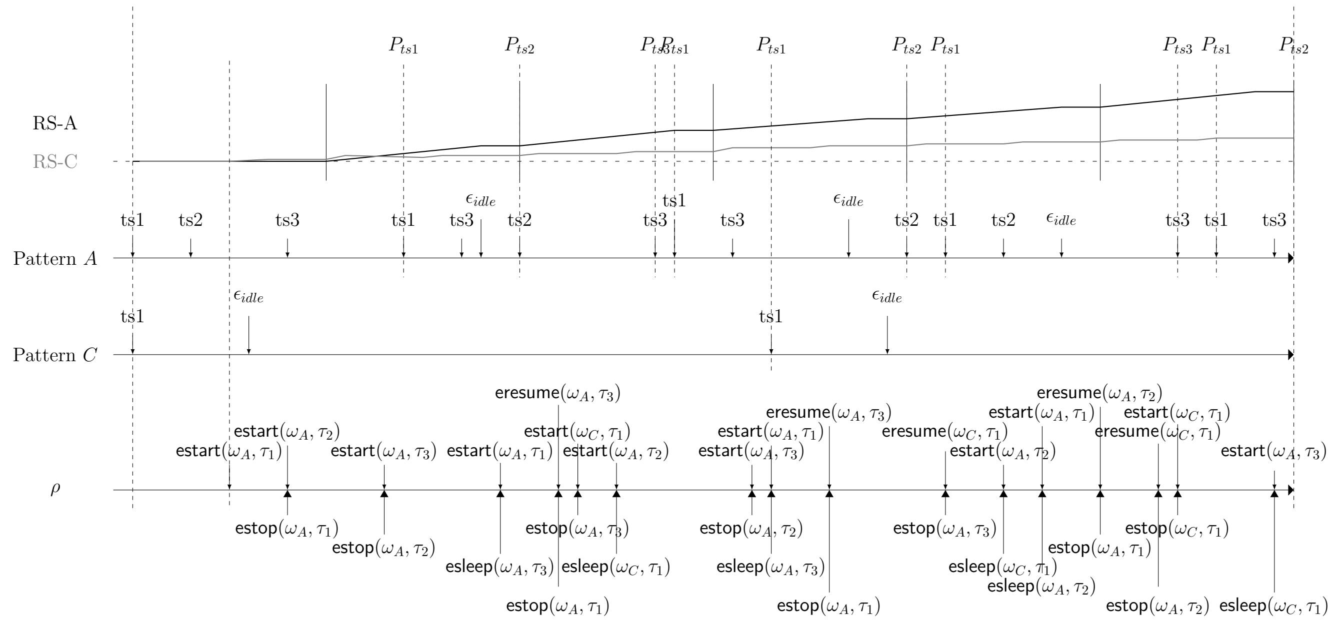

Figure 1: Example of patterns and the global trace generated by the composition of resource models defined in the Table 5

order to cope with this problem, we make the necessary assumption on the structure of traces so that their evaluation indeed produces some verdict, *i.e.*, if the system under consideration is schedulable or not.

To find the worst execution trace we begin by the introduction of the following definition.

DEFINITION 4. *The worst execution of a resource model is a trace that complains the budgets supply, the schedulability policy, the WCET of a task set, and a restricted set of intrinsic formulas.*

To generate the worst execution trace, we can adopt two distinct strategies. On one hand, we can assume some theorem that gives freely and by construction the worst trace that a system can generate (*e.g.*, for uni-processor systems we could adopt the critical instant theorem [18]). On another hand, we can rewrite our schedulability decision test into a Boolean satisfability problem. In this paper, we will focus only on the first one, and address the second one to further work. However, we believe that the last one is able to extend schedulability analysis for multi-processor systems.

Given a worst execution trace, we are able to evaluate the validity of such formula using RMTL-$\int$ for certain task set settings, and to decide if the trace is valid or invalid among the logic formulas that describe the scheduler behavior. In this way, the process remains an evaluation of the logical formula PRM or $CPRM$ for a given trace, instead of checking all traces.

Before introducing the example, we need to make two notes. Our schedulability analysis does not strongly assume the behavior of the scheduler (*e.g.*, the periodic resource model), and if the decision is negatively affirmed, a counter-example is returned (*i.e.*, a trace). We can also state that for every trace generated by a scheduler if the behavior does not correspond to the specified one then the scheduler is not a periodic resource model.

We will introduce our example assuming the critical instant theorem. Assuming this theorem we can find the *worst execution trace* for a certain workload settings. This problem is converted to an acceptance problem. We only need to apply our evaluation of RMTL-$\int$ formulas to draw a verdict about the schedulability.

6. EXEMPLIFICATION OF OUR SCHEDULABILITY ANALYSIS

Our schedulability analysis for several period resource models relax the truth notion of the WCET. This means that the WCET of a task (or several tasks) can be erroneously estimated, and ensures that the remain resource models are schedulable. In the following, we present an example of how to use the Definition 2 and Definition 3 for periodic resource models, and a composition of resource models, respectively.

To demonstrate in practice the schedulability analysis using our logic fragment, a synthetic workload will be described. Suppose, for example, a workload composed by three components, four tasks, and two monitors as depicted in Table 5. By Definition 3 we may conclude that the workload is schedulable if there exists a trace that complaints our logic restrictions.

6.1 Unfold the RMTL-$\int$ formulas

Our schedulability analysis provides two definitions for schedulability testing. According to Figure 1, we will explain step-by-step how the evaluation is performed. Beginning by unfolding the $\phi(\omega)$ of the PRM formula for the resource model RS-C, we have the formula on Table 4. The example

ϕ(RS-C)	True	$\Box_{\leq 4}\ (\mathsf{erenewal}(\omega) \implies (\Diamond_{=5}\ \mathsf{erenewal}(\omega)) \wedge \int^{5} evs^{+}(\mathsf{RS-C}, \tau_1) \leq 1)$
$\psi^{\geq}$(RS-C)	True	$\Box_{\leq 10}\neg\,(\mathsf{erelease}(\mathsf{RS\text{-}C}, \tau_1)) \vee (\neg\,(\mathsf{erelease}(\mathsf{RS\text{-}C}, \tau_1))\ \ U_{\leq 1}\ \int^{p_1} evs^{+}(\mathsf{RS\text{-}C}, \tau_1) \geq e_1)$

Trace	erenewal(RS-A), erenewal(RS-C), estart(RS-A, τ_1), estop(RS-A, τ_1), estart(RS-A, τ_2), erenewal(RS-C), estop(RS-A, τ_2), estart(RS-A, τ_3), erenewal(RS-A), erenewal(RS-C), esleep(RS-A, τ_3), $\cdots$

Table 4: Unfold of the formulas ϕ(RS-C) and $\psi^{\geq}$(RS-C) and their truth value in accordance with trace ρ.

is for a trace with duration 4 but the truth value is the same for the Figure 1. Note that this formula needs to be fed with traces whose durations are multiples of 5. Otherwise the meaning of the formula is false due to the eventually operator. The $\psi^{\geq}(\omega)$ formula ensures that the task can be executed by their WCET. Since the resource model RS-C only contains one task, we need to ensure the worst duration for this task as specified in Table 4. We evaluate the formula with a trace of duration 10. The remaining formulas $\eta(\omega), \xi^1(\omega), \xi^2(\omega, t)$ are trivially satisfied since we have only one task in our resource model RS-C. Note that the release events of the periodic resource models and the tasks are not considered in trace ρ to decrease the trace complexity, but they are considered when the formulas are evaluated.

Considering the resource model RS-A, which has a more elaborated formula after its unfolding, we are able to exemplify why the composition of two schedulable resource models are not schedulable when coupled. In Figure 1 we have generated a counter-example, namely, the trace ρ. Applying the formula $CPRM$ for the composition, the formula $\eta(\Omega)$ does no hold since it is impossible to force the consumption of the WCET of a task until its next period (only five units can be assigned until the period of the next execution). If we change the period of the task τ_1 in RS-C to value 50 instead of 33 our composition of resource models RS-A and RS-C is schedulable. We can check this informally by looking at Figure 1, unfold our formula $CPRM$ and draw a verdict for each formula of the resulting conjunction.

7. CONCLUSION AND FURTHER WORK

In this paper we have introduced a novel approach to schedulability analysis based on timed temporal logics. Compared with classical methods, our approach has a built-in scheduler behavior; avoids the rate approximations of events as experienced in [26], allows us to extend this analysis for runtime monitoring architectures by ensuring the maximum detection delay of the monitors with a simple response time RMTL-$\int$ formula; and supplies a predictable trace set of traces that can be analyzed prior to the execution and provide counter-examples.

In terms of future work, our aim is to implement a verification platform that incorporates the ideas presented in this paper with primary focus on the automatic synthesis of RMTL-$\int$ specifications into runtime monitors and corresponding program instrumentation. Alternatively, we are also interested in encoding our schedulability test into reachability analysis and use model checking tools such as *e.g.* the NUSMV model checker tool [6] to check them, or by using statistical methods to solve such issue, which is commonly natural in cases where the previous methods do not have enough resources to do the job. Yet another alternative is to encode our language in some formal verification framework, such as the Why3 or BoogiePL [8] intermediate verification languages, and rely on their backend provers to improve the chances of automatically proving the properties.

	RS Setting			Task Setting			
rs_{id}	π	θ	$\gamma_{\Omega}^{(rs_{\mathrm{id}})}$	id	p_{id}	$\gamma_{(rs_{\mathrm{id}})}^{(\mathrm{id})}$	e_{id}
RS-A	10	8	{rs-C}	ts1	14	$\emptyset$	3
				ts2	20	{ts1}	5
				ts3	27	{ts1, ts2}	7
RS-C	5	1	$\emptyset$	ts1	33	$\emptyset$	6

Table 5: A synthetic workload scheme

Acknowledgments

This work was partially supported by National Funds through FCT (Portuguese Foundation for Science and Technology) and by ERDF (European Regional Development Fund) through COMPETE (Operational Programme 'Thematic Factors of Competitiveness'), within projects Ref. FCOMP-01-0124-FEDER-022701 (CISTER), FCOMP-01-0124-FEDER-015006 (VIPCORE) and FCOMP-01-0124-FEDER-020486 (AVIACC).

REFERENCES

[1] N. C. Audsley, A. Burns, R. I. Davis, K. W. Tindell, and A. J. Wellings. Fixed priority pre-emptive scheduling: an historical perspective. *Real-Time Syst.*, 8(2-3):173–198, March 1995.

[2] G. Behrmann, A. David, K. G. Larsen, J. Hakansson, P. Petterson, W. Yi, and M. Hendriks. UPPAAL 4.0. QEST '06, pages 125–126, 2006.

[3] H. Ben-Abdallah, J. Choi, D. Clarke, Y. S. Kim, I. Lee, and H. Xie. A process algebraic approach to the schedulability analysisof real-time systems. *Real-Time Syst.*, 15(3):189–219, 1998.

[4] F. Bobot, J. C. Filliâtre, C. Marché, and A. Paskevich. Why3: Shepherd Your Herd of Provers. In *Boogie 2011: First International Workshop on Intermediate Verification Languages*, pages 53–64, 2011.

[5] A. Burns and T. M. Lin. An engineering process for the verification of real-time systems. *Form. Asp. Comput.*, 19(1):111–136, 2007.

[6] A. Cimatti, E. M. Clarke, F. Giunchiglia, and M. Roveri. Nusmv: a new symbolic model checker. *International Journal on Software Tools for Technology Transfer*, 2(4):410–425, 2000.

[7] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model checking*. MIT Press, Cambridge, MA, USA, 1999.

[8] Robert Deline, K. Rustan, and M. Leino. Boogiepl: A typed procedural language for checking object-oriented programs. Technical report, 2005.

[9] E. Fersman, P. Krcal, P. Pettersson, and W. Yi. Task automata: Schedulability, decidability and undecidability. *Inf. Comput.*, 205(8):1149–1172, 2007.

[10] E. Fersman, L. Mokrushin, P. Pettersson, and W. Yi. Schedulability analysis of fixed-priority systems using timed automata. *Theor. Comput. Sci.*, 354(2):301–317, 2006.

[11] E. Fersman, P. Pettersson, and W. Yi. Timed Automata with Asynchronous Processes: Schedulability and Decidability. TACAS '02, pages 67–82, 2002.

[12] C. J. Fidge. Real-time schedulability tests for preemptive multitasking. *Real-Time Syst.*, 14(1):61–93, January 1998.

[13] Z. Gu and K. G. Shin. Analysis of event-driven real-time systems with time petri nets: A translation-based approach. DIPES '02, pages 31–40, 2002.

[14] P. Krčál, M. Stigge, and W. Yi. Multi-processor schedulability analysis of preemptive real-time tasks with variable execution times. FORMATS'07, pages 274–289, 2007.

[15] P. Krčál and W. Yi. Decidable and undecidable problems in schedulability analysis using timed automata. volume 2988 of *TACAS'04*, pages 236–250. Springer-Verlag, 2004.

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

[17] J. Lehoczky, Lui Sha, and Y. Ding. The rate monotonic scheduling algorithm: exact characterization and average case behavior. In *Real Time Systems Symposium*, pages 166–171, 1989.

[18] C. L. Liu and James W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *J. ACM*, 20(1):46–61, January 1973.

[19] Dillon Pariente and Emmanuel Ledinot. Formal Verification of Industrial C Code using Frama-C: a Case Study. FoVeOOS'10, 2010.

[20] Anna Philippou, Insup Lee, Oleg Sokolsky, and Jin-Young Choi. A process algebraic framework for modeling resource demand and supply. FORMATS'10, pages 183–197, 2010.

[21] L. Pike, A. Goodloe, R. Morisset, and S. Niller. Copilot: a hard real-time runtime monitor. RV'10, pages 345–359, 2010.

[22] S. Pinisetty, Y. Falcone, T. Jéron, H. Marchand, A. Rollet, and O. Nguena Timo. Runtime enforcement of timed properties. RV'13, pages 229–244. 2013.

[23] I. Shin and I. Lee. Periodic Resource Model for Compositional Real-Time Guarantees. RTSS '03, pages 2–13, 2003.

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

[25] J. P. Tsai, S. J. Yang, and Y. Chang. Timing constraint petri nets and their application to schedulability analysis of real-time system specifications. *IEEE Trans. Softw. Eng.*, 21(1):32–49, 1995.

[26] H. Zhu, M. B. Dwyer, and S. Goddard. Predictable Runtime Monitoring. ECRTS '09, pages 173 –183, 2009.

[27] H. Zhu, S. Goddard, and M.B. Dwyer. Selecting server parameters for predictable runtime monitoring. RTAS'10, pages 227–236, 2010.