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-ʃ [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, jit-
ters, periods, deadlines);

4. standard approaches are not possible to extend with
other useful properties such as monitoring and en-
forcement of real-time properties [22, 21], due to the
restricted definition of their sets of tasks (e.g., defin-
ning 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 min-
imization and produces significant issues (e.g., under
and over estimations, local minimums and maximums,
etc.).

Our work intends to integrate the description of the schedul-
ing behavior with the schedulability analysis, which enables
to draw counter-examples when the system is not schedula-
able. 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
intend 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 com-
positional resource models, on which we intend 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 uni-
ified 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 du-
rations (MTL$^{-}$), namely restricted metric temporal logic
with durations (RMTL$^{-}$), a schedulability analysis for pe-
riodic resource models and coupled periodic resource models
as well as the encoding of both models in RMTL$^{-}$. Our en-
coding 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 com-
ponent knowing only the high-level specifications (instead of
the internals) of the other components, but excludes the in-
crement of components at runtime. A synthetic workload is
also described to exemplify the schedulability analysis using
RMTL$^{-}$. For the best of our knowledge this is the first ap-
proach that combines MTL$^{-}$ with schedulability analysis.

The paper is organized as follows. Section 2 introduces re-
search work that relates to the one presented in this pa-
per. 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$^{-}$
logic, including a set of necessary axioms. Section 5 in-
troduces the new concept of schedulability analysis using
MTL$^{-}$ 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 spe-
cific 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 sys-
tem. 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. Intu-
itively, the process consists by modeling the scheduling be-

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 per-
formed 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 check-
ing schedulability relative to a non-preemptive scheduling
strategy for task automata is decidable, and more gener-
ally 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 execu-
tion 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 preempt-
tive schedulers when the computation times of tasks may
vary within a known interval [14].
For multi-processor systems, the problem is also undecidable [14]. Kral 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
\[ \Gamma = \{\tau_1, \tau_2, \ldots, \tau_n\}, \]
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 \( \tau_i \); and a set of periodic resource model \( \Omega = \{\omega_1, \omega_2, \ldots, \omega_m\} \) with
\[ \omega_j = (\tau, \pi, \theta, rm), \]
where \( \tau \subseteq \Gamma, \pi \) is the replenishment period, \( \theta \) is the server budget, and \( rm \) is the RM scheduling policy.

The outputs of a resource model \( \omega \) are sequences of events. Considering a par \( (\omega, \tau_i) \) with \( \omega \in \Omega \) and \( \tau_i \in \tau \), each event can be of one of the following types: a release-event \( \text{erel}_i(\omega, \tau_i) \); a start-event \( \text{est}_i(\omega, \tau_i) \); a sleep-event \( \text{esleep}(\omega, \tau_i) \); a resume-event \( \text{erresume}(\omega, \tau_i) \); or a stop-event \( \text{estop}(\omega, \tau_i) \). 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 \( \text{ernew}(\omega) \) 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 \( x = \left\lceil \frac{(p - e)}{p} \right\rceil \).

Lehoczky et al. [17] proposed a demand-bound function \( dbfRM(\tau, t, i) \) 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
\[ dbfRM(\tau, t, i) = \sum_{\tau_k \in \gamma_i(t)} t. e_k, \]
where \( \gamma_i(t) = \{\tau_1, \ldots, \tau_i\} \) is a function that returns a set of tasks with higher-priority (and including) than task \( \tau_i \). 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 \( \tau \) of a resource model is said schedulable according to a RM policy if, and only if,
\[ \forall \tau_i \in \tau, \exists l_i \in [0, p_i] \text{ s.t. } dbfRM(\tau, t, i) \leq sbf_\omega(t_i). \]

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-\( \mathcal{F} \), a fragment of MTL-\( \mathcal{F} \) [16] where the evaluation is carried out with respect to sequences of events produced by resource models.

The main motivation for proposing RMTL-\( \mathcal{F} \) 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-\( \mathcal{F} \) and MTL-\( \mathcal{F} \) 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.
We are now able to define the semantic of the MTL-$f$. The semantic of MTL-$f$ is separated in two parts: terms and formulae. The semantic of terms is defined using the notation $T[\tau](\sigma, \vartheta)$ in Table 2. All terms represent numerical values in $\mathbb{R}_0$. The term $\int_0^\tau \varphi$ 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-$f$ logic. The semantic of the MTL-$f$ formula is defined inductively in Table 2, where the satisfaction of a formula $\varphi$ in a model $(\sigma, \vartheta)$ 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 specifications. For illustrative purposes, we now introduce a practical example of the expressive power of RMTL-$f$’s language.

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

<table>
<thead>
<tr>
<th>Operator</th>
<th>Abbreviation</th>
<th>Equivalent Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eventually</td>
<td>$\bowtie_{\leq \alpha} \varphi$</td>
<td>true $U_{\leq \alpha} \varphi$</td>
</tr>
<tr>
<td>Always</td>
<td>$\Box_{\leq \alpha} \varphi$</td>
<td>$\neg (\bowtie_{\leq \alpha} \neg \varphi)$</td>
</tr>
<tr>
<td>Next</td>
<td>$\psi_2 \land \varphi_1 \land \neg \psi_2$</td>
<td>$\psi_1 \land U_{\leq \alpha} \varphi_2$</td>
</tr>
<tr>
<td>Implies Next</td>
<td>$\psi_1 \Rightarrow \varphi_2$</td>
<td>$\neg \psi_1 \lor \Box \psi_2$</td>
</tr>
</tbody>
</table>

Table 3: Syntactic abbreviations for RMTL-$f$.

**5. SCHEDULABILITY ANALYSIS USING MTL-$f$**

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-$f$ fragment.

**5.1 Encoding Notations**

To ease the encoding of schedulability analysis properties, we first introduce some syntactical notations and formulae abbreviations. Let $\omega$ be a resource model in $\Omega$ and let $\tau_1$ be a task in $\tau$. The set of tasks with higher-priority (and including) than $\tau_1$ for $\omega$ is denoted by $\gamma_\omega[\tau_1]$. The set of resource models with higher-priority (and excluding) than $\omega$ is denoted by $\gamma^\omega[\tau_1]$. For events, we have the following notations: $\epsilon(\omega, \cdot)$ denotes the set of events that can be generated by the resource model $\omega$; $\text{evs}^+(\omega, \tau_1)$ is defined by

$$\epsilon(\omega, \cdot) \cup \text{estop}(\omega, \tau_1) \cup \epsilon(\omega, \tau_1, \cdot) \cup \text{erelease}(\omega, \tau_1),$$

with $\text{evs}^+(\omega, \tau_1)$ defined by

$$\epsilon(\omega, \cdot) \cup \text{estart}(\omega, \tau_1) \cup \text{eresume}(\omega, \tau_1) \cup \text{erenewal}(\omega),$$

which specifies all events that a task $\tau_1$ in the resource model $\omega_1$ can trigger; $\text{evs}^-(\omega_1, \tau_1)$ denotes the formula resulting from the removal of the $\text{erelease}(\omega_1, \tau_1)$ and $\text{estop}(\omega_1, \tau_1)$ events from $\text{evs}^+(\omega_1, \tau_1)$; and $\text{evs}^+(\omega_1, \tau_1)$ denotes the formula resulting from the removal of the $\text{estart}(\omega_1, \tau_1)$ and $\text{stop}(\omega_1, \tau_1)$ events from $\text{evs}^+(\omega_1, \tau_1)$.

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

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

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:

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

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

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
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

\[ \diamond_{<\alpha} \text{occur}(\varepsilon(\omega, \tau_i, \cdot), z). \]

Informally, the formula indicates that there exists at most $z$ occurrences of $\varepsilon(\omega, \tau_i, \cdot)$ until a time units. The maximal value to which $z$ can be assigned is the positive infinity ($\infty$). We select this maximum for $z$ as the initial point, and decrease successively the variable $z$ during 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., $\infty$ is replaced by $|p|$, the length of trace $p$).

### 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

\[ \phi(\omega) \equiv \square_{<\infty} (\text{erenewal}(\omega) \land rp(\omega)), \]

where

\[ rp(\omega) \equiv (\diamond_{=\theta} \text{erenewal}(\omega)) \land \int_0^\pi \bigvee_{\tau_i \in \tau} \text{evs}^{+}(\omega, \tau_i) \leq \theta, \]

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

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

\[ \eta(\theta) \equiv \square_{<\infty} \bigwedge_{\tau_i \in \tau} (\text{release}(\omega, \tau_i) \land sq(\omega, \tau_i)), \]

where

\[ sq(\omega, \tau_i) \equiv ev(\omega, \tau_i) \land \text{estop}(\omega, \tau_i), \]

\[ ev(\omega, \tau_i) \equiv \left( \bigvee_{\tau_k \in \gamma^{(\tau_i-1)}} \text{evs}^{+}(\omega, \tau_k) \right) \land \text{evs}^{-}(\omega, \tau_i) \]

and $\gamma^{(\tau_i-1)}$ denotes the set of higher-priority tasks, excluding events triggered by the task $\tau_i$. This formula means that for every event $\text{release}(\omega, \tau_i)$ there is always an event $\text{estop}(\omega, \tau_i)$, and that the events occuring before $\text{estop}(\omega, \tau_i)$ should be any event from $\tau_i$’s higher-priority tasks.

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

\[ \psi^{\leq}(\omega) \equiv \square_{<\infty} \bigwedge_{\tau_i \in \tau} (\text{release}(\omega, \tau_i) \land du^{\leq}(\omega, \tau_i)), \]

where

\[ du^{\leq}(\omega, \tau_i) \equiv \int_{\tau_k \in \gamma^{(\tau_i)}} \text{evs}^{+}(\omega, \tau_k) \leq e_i. \]

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 $\text{estop}(\omega, \tau_i)$ (i.e., each event $\text{estart}(\omega, \tau_i)$ is always followed by an event $\text{estop}(\omega, \tau_i)$, 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 $\text{estop}(\omega, \tau_i)$ is specified
The encoding of the periodic resource model is given by
\[ t \in T \]
where
\[ \omega \] describes the behavior in RMTL-CPRM and concludes the formalization of the periodic resource model's workload, which allows us to unroll the sub-formulas. This is specified by the formula
\[ \psi \equiv \bigwedge_{t \in T} \left( \text{erelease}(\omega, \tau) \cup \text{re}(\omega, \tau) \right) \cap \text{estop}(\omega, \tau_i), \]
and
\[ \text{erelease}(\omega, \tau_i) \equiv \bigvee_{\tau_k \in \gamma^i_{\omega}} \text{evs}^+ (\omega, \tau_k) \lor \text{evs}^* (\omega, \tau_i). \]

The release of events is captured by the recursive function \( \mu(e, t_i, \rho, n) \). To ensure the periodicity of all events (erelease and \( r^* \)) for certain \( t \) time units, we introduce the formula
\[ \xi^3 (\omega, t) \equiv \text{oc}(\omega, t) \land \mu \left( \text{erelease}(\omega, \tau), \pi, 0, \left\lfloor \frac{t}{p_1} \right\rfloor \right), \]
where
\[ \text{oc}(\omega, t) \equiv \bigwedge_{t \in T} \left( \text{erelease}(\omega, \tau_i), p_i, 0, \left\lfloor \frac{t}{p_1} \right\rfloor \right), \]
\[ \left\lfloor \frac{t}{p_1} \right\rfloor \] is the number of occurrences of \( \text{erelease}(\omega, \tau) \) in \( t \), and \[ \left\lfloor \frac{t}{p_1} \right\rfloor \] is the number of occurrences of \( \text{erelease}(\omega, \tau) \) in \( t \), \( \pi \) is the period of the budget renewal of the resource model \( \omega \), and \( p_i \) is the period of the task \( \tau_i \). 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
\[ \text{PRM}(\omega, t) \equiv \phi(\omega) \land \eta(\omega) \land \psi^L (\omega) \land \xi^3 (\omega, t), \]
where \( \omega \) 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-f.

### 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
\[ \phi(\omega)(t) \equiv \bigwedge_{\omega \in \Omega} \left( \phi(\omega) \land \mu \left( \text{erelease}(\omega, \tau), \pi, 0, \left\lfloor \frac{t}{p_1} \right\rfloor \right) \right). \]

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 \( \Omega \) is specified by the formula
\[ \eta(\Omega) \equiv \bigwedge_{\tau_i \in \tau} (\text{erelease}(\omega, \tau_i) \cup \text{re}(\omega, \tau_i)), \]
where
\[ \text{re}(\omega, \tau_i) \equiv \bigvee_{\tau_k \in \gamma^i_{\omega}} \text{evs}^+ (\omega, \tau_k) \lor \text{evs}^* (\omega, \tau_i). \]

The formula \( \text{re}(\omega, \tau_i) \) describes the resource events that can occur before an event \( \text{estop}(\omega, \tau_i) \) event.

The execution time allowed for a set of resource models \( \Omega \) is defined by
\[ \psi^L (\omega, t) = \bigwedge_{\omega \in \Omega} \left( \psi^L (\omega) \land \text{oc}(\omega, t) \right). \]

To specify the worst-case we can use \( \psi^L (\omega) \) instead of \( \psi^E (\omega) \). Substituting the above formula we have \( \psi^L (\omega, t) \).

The composition of periodic resource models is encoded by the RMTL-f formula
\[ \text{CPRM}(\Omega, t) \equiv \phi^L (\omega, t) \land \eta(\Omega) \land \psi^L (\omega, t) \land \left( \bigwedge_{\omega \in \Omega} \xi^3 (\omega) \right). \]

### 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-f we need to find an observation, a logical environment, and a source models. By the semantic definition of RMTL-f we need to find an observation, a logical environment, and a source models. By the semantic definition of RMTL-f we need to find an observation, a logical environment, and a source models. By the semantic definition of RMTL-f we need to find an observation, a logical environment, and a source models. By the semantic definition of RMTL-f we need to find an observation, a logical environment, and a source models.

**Definition 2.** Let \( \omega \) be a resource model in \( \Omega \). The resource model \( \omega \) is schedulable if and only if, there exists a trace \( \rho \) of duration \( t \) such that \( \text{PRM}(\omega, t) \) is satisfied, and the duration of \( \rho \) is greater or equal to the maximum value of \( p_i \) in \( \tau \).

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

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

### 5.5 Feasible Tests

Encoding the schedulability test is not enough to ensure that we always obtain a positive or negative answer. In
In 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 satisfiability 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-$f$ 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 formulas to draw a verdict about the schedulability.

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-$f$ 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 formulas. 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.

### 6.1 Unfold the RMTL-$f$ 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
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 $\psi^2(\omega)$ 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 $η(ω)$, $ξ^1(ω)$, $ξ^2(ω)$, $t$ 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 $η(Ω)$ 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.

### Table 4: Unfold of the formulas $ϕ(RS-C)$ and $ψ^2(RS-C)$ and their truth value in accordance with trace $ρ$

| $ϕ(RS-C)$ | True | $}\Diamond_{\omega} (\text{renewal}(ω) \land \int_{0}^{\omega} \text{evs}(ω) \land (RS - C, τ_1) \leq 1$ |
| $ψ^2(RS-C)$ | True | $}\Box_{\omega} (\text{release}(RS-C, τ_1)) \lor (	ext{release}(RS-C, τ_1)) U_{\omega} \{ \text{ts1}, \text{ts2} \}$ |

| Trace | $\text{renewal}(RS-A), \text{renewal}(RS-C), \text{estart}(RS-A, τ_1), \text{estop}(RS-A, τ_1), \text{estart}(RS-A, τ_2), \text{renewal}(RS-C), \text{estop}(RS-A, τ_2), \text{estart}(RS-A, τ_3), \text{renewal}(RS-A), \text{renewal}(RS-C), \text{esleep}(RS-A, τ_1), \cdots$ |

### Table 5: A synthetic workload scheme

<table>
<thead>
<tr>
<th>$\pi$, id</th>
<th>$\theta$, id</th>
<th>$γ^{(\text{rs-id})}_1$</th>
<th>$\rho_{\text{id}}$, id</th>
<th>$γ^{(\text{rs-id})}_2$</th>
<th>$c_{\text{id}}$, id</th>
</tr>
</thead>
<tbody>
<tr>
<td>RS-A</td>
<td>10</td>
<td>8</td>
<td>${\text{rs-C}}$</td>
<td>$\text{ts1}$</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>${\text{ts1}}$</td>
<td>$\text{ts2}$</td>
<td>27</td>
<td>${\text{ts1}, \text{ts2}}$</td>
</tr>
<tr>
<td>RS-C</td>
<td>5</td>
<td>1</td>
<td>$\emptyset$</td>
<td>$\text{ts1}$</td>
<td>33</td>
</tr>
</tbody>
</table>

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-$f$ 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-$f$ 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.

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


