Compositional Timing Analysis of Real-Time Systems Based on Resource Segregation Abstraction

Philipp Reinkemeier, Ingo Stierand

Compositional Timing Analysis of Real-Time Systems based on Resource Segregation Abstraction*

Philipp Reinkemeier[1] and Ingo Stierand[2]

[1] OFFIS - Institute for Information Technology
Oldenburg, Germany
philipp.reinkemeier@offis.de
[2] Carl von Ossietzky Universität Oldenburg
Oldenburg, Germany
stierand@informatik.uni-oldenburg.de

Abstract. For most embedded safety-critical systems not only the functional correctness is of importance, but they must provide their services also in a timely manner. Therefore, it is important to have rigorous analysis techniques for determining timing properties of such systems. The ever increasing complexity of such real-time systems calls for compositional analysis techniques, where timing properties of local systems are composed to infer timing properties of the overall system. In analytical timing analysis approaches the dynamic timing behavior of a system is characterized by mathematical formulas abstracting from the state-dependent behavior of the system. While these approaches scale well and also support compositional reasoning, the results often exhibit large over-approximations. Our approach for compositional timing analysis is based on ω-regular languages, which can be employed in automata-based model-checking frameworks. To tackle the scalability problem due to state-space explosion, we present a technique to abstract an application by means of its resource demands. The technique allows to carry out an analysis independently for each application that shall be deployed on the same platform using its granted resource supply. Integration of the applications on the platform can then be analyzed based on the different resource supplies without considering details of the applications.

1 Introduction and Related Work

Developing safety-critical real-time systems is becoming increasingly complex as the number of functions realized by these systems grows. Additionally an increasing number of functions is realized in software, which is then integrated

* This work was partly supported by the Federal Ministry for Education and Research (BMBF) under support code 01IS11035M, 'Automotive, Railway and Avionics Multicore Systems (ARAMiS)', and by the German Research Council (DFG) as part of the Transregional Collaborative Research Center 'Automatic Verification and Analysis of Complex Systems' (SFB/TR 14 AVACS).

on a common target platform in order to save costs. The integration on a common platform causes interferences between the different software-functions due to their shared resource usage. It is desirable to bound these interferences in a way to make guarantees about the timing behavior of the individual software-functions. A schedulability analysis delivers such bounds for interferences between software-tasks sharing a CPU by means of a scheduling strategy.

We present a compositional analysis framework using real-time interfaces based on ω-regular languages. Following the idea of interface-based design, components are described by interfaces and can be composed if their corresponding interfaces are compatible. The contribution of this work is a framework allowing to formally capture the resource demand of an interface, that we call segregation property. Compatibility of interfaces then can be reduced to compatibility of their segregation properties. Further a refinement relation is defined, which leads to a sufficient condition for compatibility of segregation properties. The framework can be used in (but is not restricted to) scenarios like the following: The bottom part of Fig. 1 shows a target platform that is envisioned by say an

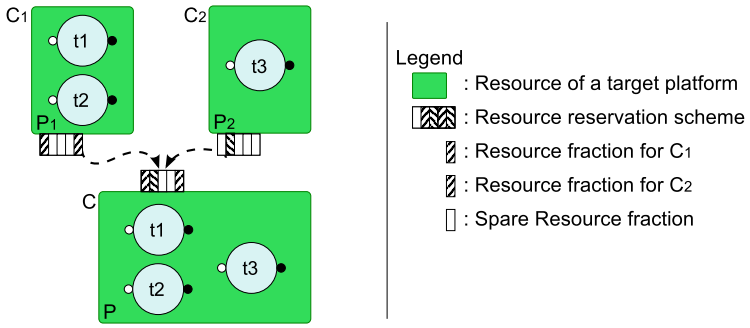

Fig. 1. Exemplary Integration Scenario using Resource Segregation

Original Equipment Manufacturer (OEM). It consists of a processing node (P). Suppose the OEM wants to implement two applications, components C_1 and C_2, on this architecture and delegates their actual implementation to two different suppliers. Both applications share the same resource of the target platform, i.e. tasks t_1, t_2 and t_3 are all executed on P after integration. Therefore, a resource reservation is assigned to each component, guaranteeing a certain amount of resource supply. Then the timing behavior of both components can be analyzed independently from each other based on their resource demands and the guaranteed resource supply. Verification of the successful integration of C_1 and C_2 on the platform then amounts to check whether the reserved resource supplies can be composed.

There have been considerable studies on compositional real-time scheduling frameworks [12, 13, 11, 7, 5]. These studies define interface theories for components abstracting the resource requirement of a component by means of demand

functions [12, 13], bounded-delay resource models [7], or periodic resource models [11, 5]. Based on these theories the required resources of a component, captured by its interface, can for example be abstracted into a single task. This approach gives rise to hierarchical scheduling frameworks where interfaces propagate resource demands between different layers of the hierarchy. Our proposed resource segregation abstraction of a component is an extension of the real-time interfaces presented in [3]. Contrary to the aforementioned approaches, our real-time interfaces and resource segregation are based on ω-regular languages. That means, the approach can for example be employed in automata-based model-checking frameworks. In addition the results we present are not bound to specific task and resource models, like periodic or bounded delay.

Analytic methods provide efficient analysis by abstracting from concrete behavior. The drawback is that this typically leads to over-approximations of the analysis results. Computational methods on the other hand, such as model-checking for automata ([2, 9, 6]), typically provide the expressive power to model and analyze real-time systems without the need for approximate analysis methods. This flexibility comes with costs. Model-checking is computationally expensive, which often prevents analysis of larger systems. The contribution of this paper will help to reduce verification complexity for the application of computational methods.

The paper is structured as follows: Section 2 briefly introduces real-time interfaces presented in [3], where task executions are characterized by ω-regular languages over time slices occupied by the respective tasks. Section 3 provides the formalization of segregation properties for interfaces, which can be used to abstract from concrete behavior of an interface. We define refinement and composition operations on segregation properties that preserve schedulability of the composition of the associated interfaces. Section 4 shows that our approach is consistent with the (analytical) resource models of [11, 5] by the definition of a translation. Section 5 discusses further work and concludes the paper.

2 Real-Time Interfaces

The resource segregation abstraction presented in this work is based on the real-time interfaces presented in [3]. Therefore, we briefly summarize the basic definitions. We assume a set of real-time components are to be executed on a set of resources such as processing nodes and communication channels. Each component consists of a set of tasks. A real-time interface of a component specifies the set of all its legal schedules when it is executed on the resources. For example, consider a component with two tasks 1 and 2, which are scheduled on a single resource in discrete slots of some fixed duration, like shown in Fig. 1 for component P_1. A schedule for this component can be described by an infinite word over the alphabet $0, 1, 2$. 0 means the resource is idle during the slot, and 1 and 2 means the corresponding task is running. The real-time interface of a component is an ω-language containing all legal schedules of the component. Therefore, an interface with a non-empty language contains at least one schedule and is said

to be schedulable. Interfaces can be composed (intersection) to check whether two components together are schedulable.

Definition 1. *A real-time interface I is a tuple (L, T), where $T \subseteq \mathcal{T}$ is the set of tasks of the interface and $L \subseteq T^\omega$ is an ω-regular language denoting the set of legal schedules of I. The empty task 0 denoting an idle slot is part of every interface, i.e. $0 \in T$.*

The intuition of an interface is that it describes the set of schedules that satisfy the requirements of its component. An interface with an empty language is said to be not schedulable. Conversely, an interface with a non-empty language is said to be schedulable, as at least one legal schedule exists for the interface.

Example: Suppose that task t_1 in Fig. 1 is a periodic task t with period $p = 5$ and an execution time $e = 3$. The language of its interface I_1 can be described by the following regular expression: $L_{I_1} = 0^{<5}[t^3 \,|||\, 0^2]^\omega$, where $u \,|||\, v$ denotes all possible interleavings of the finite words u and v. That means, a schedule is legal for interface I_1, as long as it provides 3 slots during a time interval of 5.

Observe, that interface I_1 captures an assumption about the activation pattern of task t_1. The part $0^{<5}$ of the regular expression represents all possible phasings of the initial task activations. This correlates to the formalism of event streams, which is a well-known representation of task activation patterns in real-time systems (cf. [10]) by lower and upper arrival curves $\eta^-(\Delta t)$ and $\eta^+(\Delta t)$.

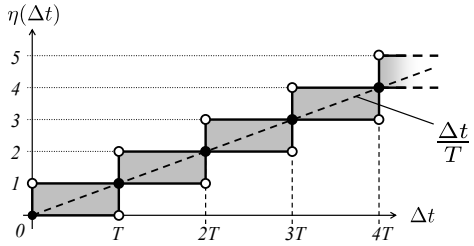

Fig. 2. Arrival curves of periodic events

Key to dealing with interfaces having different alphabets is the following projection operation: For alphabet T and language L of an interface I and $T' \subseteq T$, we consider its *projection* $pr_{T'}(L)$ to T', which is the unique extension of the function $T \to T'$ that is identity on the elements of T' and maps every element of $T \setminus T'$ to 0. We will also need the *inverse projection* $pr^{-1}_{T''}(L)$, for $T'' \supseteq T$, which is the language over T'' whose words projected to T belong to L.

Definition 2. *Given two interfaces $I_1 = (L_1, T_1)$ and $I_2 = (L_2, T_2)$ the parallel composition $I_1 \parallel I_2$ is the interface (L, T), where*

- $T = T_1 \cup T_2$ *and*

– $L = pr_T^{-1}(L_1) \cap pr_T^{-1}(L_2)$

The intuition of this definition is that a schedule is legal for $I_1 \parallel I_2$ if its restriction to T_2 is legal for I_2 and its restriction to T_1 is legal for I_1. That means tasks of an interface are allowed to run when the resource is idle in the other interface.

Definition 3. *Given two interfaces $I = (L, T)$ and $I' = (L', T')$, then I' refines I, $I' \preceq I$, if and only if:*

– $T' \supseteq T$ *and*
– $pr_T(L') \subseteq L$

The intuition of this definition is that all schedules legal in I' are (modulo projection) also legal schedules in I and I' is able to schedule more tasks from the set $T' \setminus T$ in the gaps left by schedules in I.

The following lemmas provide useful properties of the real-time interface framework.

Lemma 1. *Parallel composition of interfaces is associative and commutative.*

An associative and commutative composition operation guarantees that composable interfaces may be assembled together in any order. Therefore, real-interfaces support *incremental design*.

Lemma 2. *Refinement of interfaces is a partial order.*

As refinement is a partial order, it is ensured that: If interface $I' \preceq I$, then for any interface $I'' \preceq I'$ it holds that $I'' \preceq I$. That means interfaces can be refined iteratively.

Lemma 3. *Refinement is compositional. That means $I' \preceq I$ implies $I' \parallel J \preceq I \parallel J$.*

A compositional refinement allows to refine composable interfaces separately, while maintaining composability. Together with commutativity and associativity of the composition operator, we have that the real-time interfaces support *independent implementability*. Proofs for Lemma 1-3 are presented in [3].

3 Resource Segregation

While real-time interfaces are powerful enough to cope with complex designs and scenarios like depicted in Fig. 1, the refinement relation involves complex language inclusion checks. Moreover, the details of all components and their tasks must be known in order to compose them. Therefore, we introduce an abstraction for a real-time interface consisting of multiple tasks that we call *segregation property*. These segregation properties will be defined such that compositionality of segregation properties ensures compositionality of their interfaces, respectively. That means given two interfaces I_1 and I_2 and segregation properties B_{I_1} and B_{I_2}, we look for a composition operator $\parallel$ and a simple property φ, such that

$$B_{I_1} \parallel B_{I_2} \models \varphi \implies I_1 \parallel I_2 \text{ is schedulable}$$

3.1 Interface Composability

Recall, that an interface I describes a set of legal schedules. It represents for the activation patterns of its tasks a set of possible discrete slot allocations under which the tasks can be executed successfully. A segregation property B_I for an interface abstracts from the tasks of the interface, and only exposes a set of possible slots reservations for which the interface is schedulable. Note that a segregation property for an interface indeed may contain more available slots than are used by the respective interface.

Composition of the segregation properties B_{I_1} and B_{I_2} of interfaces I_1 and I_2 then combines non-conflicting slot reservations of B_{I_1} and B_{I_2}. The property φ states that at least one such non-conflicting slot reservation exists, i.e. the set of slot reservations defined by $B_{I_1} \parallel B_{I_2}$ is not empty.

We now define *slot reservation* and *segregation property* of an interface and define a composition operation. We use the composition operation on slot reservations to derive a composability condition for interfaces based on their segregation properties.

Definition 4. *A* slot reservation B *is an ω-regular language over $\{0,1\}$, $B \subseteq \{0,1\}^\omega$. Each ω-word $b \in B$ defines an infinite sequence of slots that are either available (0) or unavailable (1).*

We denote B_I a segregation property *for interface I, if and only if for all $b \in B_I$ holds that I is schedulable for all its activation patterns using only the available slots defined in b.*

Example: For task t_1 in the example above, B_{I_1} is a segregation property of I_1 if it contains for each sub-sequence of length 5 at least 3 available slots. A valid segregation property for I_1 is $B_{I_1} = \bigcup_{\sigma \in C_3^5} \sigma^\omega$, where C_3^5 denotes the set of finite words $\sigma = \sigma_1 \dots \sigma_5$ over $\{0,1\}$ of length 5 obtained by combination of 3 symbols $\sigma_i = 0$ over 5 symbols and the remaining symbols are 1.

We define the parallel composition $B_1 \parallel B_2$ of slot reservations such that we select only those pairs $b_1 \in B_1, b_2 \in B_2$, where no slot is available (0) in both words, combining them into a single word $b \in B_1 \parallel B_2$, where slots are available that are available in either b_1 or b_2 and all other slots remain unavailable (1).

For convenience, we make use of the binary operators $\wedge$ and $\vee$ defined on elements of $\{0,1\}$ with their usual Boolean interpretations. We extend both operators to ω-regular words $b_i = b_{i_1} b_{i_2} \dots \in \{0,1\}^\omega$, by their component-wise application: $b_1 \wedge b_2 = (b_{1_1} \wedge b_{2_1})(b_{1_2} \wedge b_{2_2}) \dots$, and $\vee$ respectively.

Definition 5. *Given two slot reservations B_1 and B_2 the* parallel composition *$B_1 \parallel B_2$ is defined as:*

$$B_1 \parallel B_2 = \{b_1 \wedge b_2 | b_1 \in B_1, b_2 \in B_2 \text{ and } b_1 \vee b_2 = 1^\omega\}$$

Example: Fig. 3 depicts an illustration of the composition of the segregation property B_{I_1} of I_1 from the example above with another slot reservation $B_2 = [0^1 \,|||\, 1^4]^\omega$.

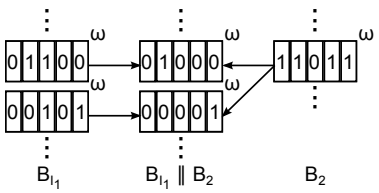

Fig. 3. Illustration of Slot Reservation Composition

The following lemma states the desired condition for composability of interfaces depending on their segregation properties:

Lemma 4. *Two interfaces I_1 and I_2 are composable and can be scheduled together if the parallel composition of their segregation properties is not empty, i.e. $B_{I_1} \parallel B_{I_2} \neq \emptyset$.*

Proof: As B_{I_1} is a segregation property for I_1 and B_{I_2} is a segregation property for I_2, I_1 is schedulable for all its activations patterns for *all* $b \in B_{I_1}$ and I_2 is schedulable for *all* $b \in B_{I_2}$, respectively. According to Definition 5 all words $b \in B_{I_1} \parallel B_{I_2}$ are sequences of slots such that there exists a pair $b_1 \in B_{I_1}, b_2 \in B_{I_2}$, where no slot is available in both words. Thus, interface I_1 can be scheduled using only the available slots in b that are also available in b_1, interface I_2 respectively. Consequently, it holds that each unavailable slot in $b_1 \in B_{I_1}$ is not used by I_1, and I_2 may schedule one of its tasks in these slots, if they are available in $b_2 \in B_{I_2}$. For interface I_2 the same argument applies. Thus, it follows that the language of $I_1 \parallel I_2$ is not empty, which according to Definition 1 means a legal schedule for $I_1 \parallel I_2$ exists. □

3.2 Refinement of Slot Reservations

Recall, that Definition 4 defines B_I to be a segregation property for interface I, if and only if I is schedulable for all its activation patterns *for every* $b \in B_I$. From this definition we conclude that given a segregation property B_I, *any subset* $B'_I \subseteq B_I$ is also a segregation property for interface I. Further, each $b = \sigma_1\sigma_2 \ldots \in B_I$ defines a sequence of slots, where all slots $\sigma_i = 0$ are available to the interface. Obviously, if the interface is schedulable for $b \in B_I$, then it is also schedulable for b', where $b' = \sigma'_1\sigma'_2 \ldots$ and $\sigma'_i = 0$ and $\sigma_i = 1$ for some i and $\sigma'_j = \sigma_j$ for all other slots $j \neq i$. In other words, we can always make more slots available to an interface without impact on its schedulability.

These observations give rise to a refinement relation on slot reservations. First, we define a partial order on ω-regular words over $\{0,1\}$ as follows: Let be $b, b' \in \{0,1\}^\omega$. We say $b' \leq b$ if and only if $\forall i \in \mathbb{N} : \sigma_{b'_i} = 1 \implies \sigma_{b_i} = 1$. That is, b' precedes b if all slots that are unavailable (1) in b' are also unavailable in b. Indeed b might contain additional unavailable slots that are available (0) in b'. In other words: Slots that are available in b are also available in b'. Obviously, a bottom element 0^ω and a top element 1^ω exist with regard to the partial order $\leq$. For any $b \in \{0,1\}^\omega$ we have that $0^\omega \leq b \leq 1^\omega$.

We extend the relation on ω-regular words over $\{0,1\}$ to slot reservations (ω-regular languages over $\{0,1\}$) as follows:

Definition 6. *Given two slot reservations B' and B, then B' refines B, $B' \preceq B$, if and only if:*

$$\forall b' \in B' : \exists b \in B : b' \leq b$$

The refinement relation $\preceq$ on slot reservations is a pre-order, as mutual refinement not necessarily implies equivalence: $B \preceq B'$ and $B' \preceq B \implies B = B'$. Note, that this definition of refinement captures both observations: Given a segregation property B_I, any subset $B'_I \subseteq B_I$ is also a segregation property for I and it holds that $B'_I \preceq B_I$. Further, for a segregation property B_I, we can construct $B'_I \preceq B_I$ from B_I, where for some $b' \in B'_I$ we make more slots available, i.e. $\exists b \in B_I : b' \leq b$. Still B'_I is a segregation property for I. Thus, given a segregation property B_I for interface I, then any $B'_I \preceq B_I$ is also a segregation property for I.

However, B'_I may be an 'over-approximation' of B_I. Consider the segregation property B_I and a subset $B'_I \subset B_I$. As the interface I is schedulable for all words $b \in B_I$, we can understand B_I as a set of alternative slot reservations 'supported' by the interface I. Thus, this alternative is lost when eliminating a word from B_I in a subset $B'_I \subset B_I$. Now consider $B''_I \preceq B_I$ obtained by replacing some word $b \in B_I$ with a word $b'' \leq b$. The interface I is schedulable using only the available slots in b. b'' may be on over-approximation as more slots can be available in b'' that are not available in b. Both over-approximations of B_I lead to an increased probability of causing slot conflicts when composing them with another segregation property $B_{\tilde{I}}$. But If that composition is still not empty, I and $\tilde{I}$ are composable and can be scheduled together.

Example: Fig. 4 depicts an illustration of the preceding discussion on refinement applied on the segregation property B_{I_1} of I_1 from the example above.

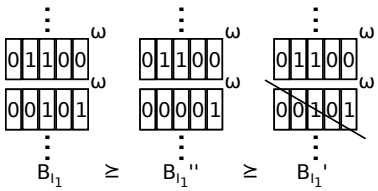

Fig. 4. Illustration of Segregation Property Refinement

The following lemma formalizes these observations and provides a sufficient condition for composability of interfaces (see Lemma 4):

Lemma 5. *Given two interfaces I_1 and I_2 and segregation properties B_{I_1} and B_{I_2}, respectively. Then for any two slot reservations $B'_{I_1} \preceq B_{I_1}$ and $B'_{I_2} \preceq B_{I_2}$ it holds that $B'_{I_1} \parallel B'_{I_2} \neq \emptyset \implies B_{I_1} \parallel B_{I_2} \neq \emptyset$.*

Proof: According to Definition 5 all words $b' \in B'_{I_1} \parallel B'_{I_2}$ are sequences of slots such that there exists a pair $b'_1 \in B'_{I_1}, b'_2 \in B'_{I_2}$, where no slot is available in both words. According to Definition 6 $b_1 \in B_{I_1}, b_2 \in B_{I_2}$ exist, where $b'_1 \leq b_1$ and $b'_2 \leq b_2$. Slots that are unavailable (1) in b'_1 are also unavailable in b_1. The same holds for b'_2 and b_2. It follows that b_1 and b_2 can be composed and $B_{I_1} \parallel B_{I_2}$ contains at least on element. □

4 Periodic Resource Models and Resource Segregation

As discussed in Section 1, the idea of resource segregation and their exploitation in compositional analysis frameworks is not new. However to our best knowledge the principle has only been applied in frameworks that are based on analytical methods. For example the frameworks proposed by I. Lee et. al. [11, 5] are based on the concepts of demand bound functions $dbf(\Delta)$ and supply bound functions $sbf(\Delta)$. The function $dbf(\Delta)$ characterizes the maximal processing demand of a real-time component within any interval of length Δ. The function $sbf(\Delta)$ characterizes the minimal processing power provided by the resource in any time interval of length Δ. The real-time component is considered to be schedulable, if $\forall \Delta : dbf(\Delta) \leq sbf(\Delta)$. Note, that the concept of service curves known from real-time calculus [4] is comparable with these frameworks, as described in [13]. In this section we discuss in more detail the relation of our approach with the frameworks presented in [11, 5]. We will see that our approach is able to capture the models considered in these frameworks, and thus results established in these frameworks also apply in our setting.

Both frameworks are based on the concepts of demand bound functions and supply bound functions, where in [11] a *Periodic Resource Model* is presented and in [5] an *Explicit Deadline Periodic Resource Model* (EDP) is presented. Both models are used to create compositional hierarchical scheduling frameworks. In both frameworks a component is a set of tasks scheduled under a specific strategy. The total resource demand of a component to schedule all its tasks is expressed as a demand bound function $dbf(\Delta)$. The resource models are used to capture the amount of resource allocations of a partitioned resource, which is formally expressed as a supply bound function $sbf(\Delta)$. If a component is schedulable under the considered partitioned resource (defined by the resource model), i.e. $dbf(\Delta) \leq sbf(\Delta)$, then the resource model can be transformed into a task and components can be composed hierarchically. Thus, the composition problem is reduced to the abstraction problem.

The periodic resource model $\Gamma = (\Pi, \Theta)$ characterizes a partitioned resource that repetitively provides Θ units of resource with a repetition period Π. The EDP resource model $\Omega = (\Pi, \Theta, \Delta)$ is an extension of the periodic resource model. It characterizes a partitioned resource that repetitively supplies Θ units of resource within Δ time units, with Π the period of repetition. Keeping in mind the idea of transforming a resource model into a task at the next level of the hierarchy, the relation between both models becomes clear: A periodic resource

model $\Gamma = (\Pi, \Theta)$ is the EDP model $\Omega = (\Pi, \Theta, \Pi)$ (cf. [5]). Therefore, in the following we focus on EDP resource models.

4.1 Real-time Component Model

A real-time component is defined as $C = \langle \{C_1, \ldots, C_n\}, S \rangle$, where C_i is either another real-time component or a sporadic task. A sporadic task is defined by a tuple $\tau = (p, e, d)$, where p is a minimum separation time, e the execution time of the task and d a deadline relative to the release of task τ. It holds that $e \leq d \leq p$. The workload $C_1, \ldots, C_n$ is scheduled under strategy S that is either *RM* (rate monotonic), *DM* (deadline monotonic) or *EDF* (earliest deadline first). The *resource demand* of a component is then the collective resource demand of its tasks under its scheduler S. The *demand bound function* [8, 1] $dbf_C(\Delta)$ characterizes the maximum resource demand for a task set in any given time interval of length Δ.

In our framework real-time components translate into interfaces, where each interface I is either a composition of interfaces $I = I_1 \parallel \ldots \parallel I_n$ or an 'atomic interface' in case of a single sporadic task. Given a task $t = (p, e, d)$, the language of the corresponding interface is $L_{I_t} = 0^{<p-1}[(t^e \,|||\, 0^{d-e})0^{p-d}]^\omega$, where $u \,|||\, v$ denotes all possible interleavings of the finite words u and v. Given a component $C = \langle \{C_1, \ldots, C_n\}, S \rangle$, then the condition $I_{C_1} \parallel \ldots \parallel I_{C_n} \neq \emptyset$ determines whether the component is schedulable at all under some scheduling strategy S. Now consider a fixed priority scheduling (FPS), say rate monotonic scheduling. The component is schedulable under FPS if and only if $I_{FPS} \preceq I_{C_1} \parallel \ldots \parallel I_{C_n}$. How to capture the scheduling of a task set under FPS in terms of an interface I_{FPS} is described in [3]. A segregation property $B_{I_{FPS}}$ of interface I_{FPS} characterizes the resource demands of $C = \langle \{C_1, \ldots, C_n\}, FPS \rangle$.

The resource demands of a component C, explicated by the demand bound function $dbf_C(\Delta)$ can be safely over-approximated by any function $f(\Delta)$, with $f(\Delta) \geq dbf_C(\Delta)$. For example in [11] a linear function $ldbf_C(\Delta)$ is given for EDF scheduling that provides an upper bound for $dbf_C(\Delta)$. In our framework over-approximations of the resource demands B_{I_C} of a component translate into refinements of B_{I_C}. As discussed in Section 3.2 any $B'_{I_C} \preceq B_{I_C}$ is also a segregation property for interface I_C, albeit a potential over-approximation of the resource demands defined by B_{I_C}.

4.2 Resource Model and Schedulability

Consider an explicit deadline periodic resource model $\Omega = (\Pi, \Theta, \Delta)$. It characterizes a partitioned resource that repetitively supplies Θ units of resource within Δ time units, with Π the period of repetition. The partitioned resource characterized by Ω, can also be characterized by the following slot reservation:

$$B_\Omega = 1^{\leq(\Pi-\Theta)}[(0^\Theta \,|||\, 1^{\Delta-\Theta})1^{\Pi-\Delta}]^\omega$$

The *resource supply* of a resource is the amount of provided resource allocations. Complementary to the demand bound function for components, the resource

supply bound function $sfb_\Omega(\Delta)$ computes the minimum resource supply for Ω in a given time interval of length Δ.

The resource supply $sfb_\Omega(\Delta)$ can be safely under-approximated by any function $f(\Delta)$, with $f(\Delta) \leq sfb_\Omega(\Delta)$. Analogously, in our framework under-approximations of the resource supply are captured by the refinement relation. B''_Ω, with $B_\Omega \preceq B''_\Omega$, is a potential under-approximation of the resource supply of a resource.

In the context of EDP resource models, schedulability is defined for a real-time component $C = \langle \{C_1, \ldots, C_n\}, S \rangle$ using an EDP resource model Ω. Exact schedulability conditions are given for the scheduling strategies RM, DM and EDF. We will not go into the details of the theorems here and refer to [5] instead. Basically it must holds that $\forall \Delta : dfb_C(\Delta) \leq sfb_\Omega(\Delta)$. Schedulability of C under Ω can be formulated in our framework as refinement. Given the segregation property B_{I_C} and the resource supply B_Ω, then C is schedulable under Ω, if $B_\Omega \preceq B_{I_C}$. Sufficient conditions based on over-approximation of resource demands and under-approximation of resource supplies are induced by transitivity of the refinement relation. Given segregation property B_{I_C}, slot reservation B_Ω and $B'_{I_C} \preceq B_{I_C}$, and $B_\Omega \preceq B''_\Omega$, then $B''_\Omega \preceq B'_{I_C} \implies B_\Omega \preceq B_{I_C}$.

5 Conclusion

This paper proposes a formalization of *segregation properties* enabling compositional timing analysis based on ω-regular languages. By exploiting the formalism of real-time interfaces, segregation properties allow to abstract the concrete behavior of components, and provide conditions under which the composition of a set of components results in a schedulable system. The approach supports the verification process in two directions. Firstly, the abstraction helps to reduce verification complexity, which often prevents analysis of larger systems using model-checking techniques. Secondly, the approach subsumes well-known approaches in the domain of analytical resource models. This enables the elaboration of combined methods to further reduce analysis efforts.

While this initial approach supports only single resources, future work will allow for the expression of multiple resources. In this case, slot reservations do not argue over the alphabet $\{0, 1\}$, but over tuples $(r_1, ..., r_n)$ for n resources, where $r_i \in \{0, 1\}$. Indeed, this requires modified definitions of composition and refinement. A further extension of this approach will allow to support multiprocessor resources. In this case, segregation properties are no more defined over the alphabet $\{0, 1\}$, but for example over sets $\{0, 1, ..., m\}$ characterizing the number of available processing units of the resource.

References

1. Baruah, S., Rosier, L., Howell, R.: Algorithms and complexity concerning the preemptive scheduling of periodic, real-time tasks on one processor. Real-Time Systems 2, 301–324 (1990)

2. Basu, A., Bozga, M., Sifakis, J.: Modeling Heterogeneous Real-time Components in BIP. In: Proc. Conference on Software Engineering and Formal Methods (SEFM) (2006)
3. Bhaduri, P., Stierand, I.: A proposal for real-time interfaces in speeds. In: Design, Automation Test in Europe Conference Exhibition (DATE), 2010. pp. 441–446 (march 2010)
4. Chakraborty, S., Kunzli, S., Thiele, L.: A general framework for analysing system properties in platform-based embedded system designs. In: Design, Automation and Test in Europe Conference and Exhibition (DATE), 2003. pp. 190–195. IEEE Computer Society (2003)
5. Easwaran, A., Anand, M., Lee, I.: Compositional analysis framework using edp resource models. In: Proceedings of the 28th IEEE International Real-Time Systems Symposium. pp. 129–138. RTSS '07, IEEE Computer Society (2007)
6. Guan, N., Ekberg, P., Stigge, M., Yi, W.: Effective and Efficient Scheduling of Certifiable Mixed-Criticality Sporadic Task Systems. In: Proc. Real-Time Systems Symposium (RTSS) (2011)
7. Henzinger, T., Matic, S.: An interface algebra for real-time components. In: Proceedings of the 12th IEEE Real-Time and Embedded Technology and Applications Symposium. pp. 253–266. RTAS '06, IEEE Computer Society (2006)
8. Lehoczky, J.P., Sha, L., Ding, Y.: The rate monotonic scheduling algorithm: Exact characterization and average case behavior. In: IEEE Real-Time Systems Symposium. pp. 166–171. IEEE Computer Society (1989)
9. Perathoner, S., Lampka, K., Thiele, L.: Composing Heterogeneous Components for System-wide Performance Analysis. In: Design, Automation Test in Europe Conference Exhibition (DATE) (2011)
10. Richter, K.: Compositional Scheduling Analysis Using Standard Event Models. Ph.D. thesis, Technical University of Braunschweig, Germany (2004)
11. Shin, I., Lee, I.: Periodic resource model for compositional real-time guarantees. In: Proceedings of the 24th IEEE International Real-Time Systems Symposium. pp. 2–13. RTSS '03, IEEE Computer Society (2003)
12. Thiele, L., Wandeler, E., Stoimenov, N.: Real-time interfaces for composing real-time systems. In: Proceedings of the 6th ACM & IEEE International Conference on Embedded Software. pp. 34–43. EMSOFT '06, ACM (2006)
13. Wandeler, E., Thiele, L.: Interface-based design of real-time systems with hierarchical scheduling. In: Proceedings of the 12th IEEE Real-Time and Embedded Technology and Applications Symposium. pp. 243–252. RTAS '06, IEEE Computer Society (2006)