A Process Algebraic Framework for Modeling Resource Demand and Supply*

Anna Philippou[1], Insup Lee[2], Oleg Sokolsky[2], and Jin-Young Choi[3]

[1] Department of Computer Science, University of Cyprus, Nicosia, Cyprus
[2] Department of Computer and Info. Science, University of Pennsylvania, Philadelphia, PA, U.S.A.
[3] Department of Computer Science, Korea University, Seoul, Korea

Abstract. As real-time embedded systems become more complex, resource partitioning is increasingly used to guarantee real-time performance. Recently, several compositional frameworks of resource partitioning have been proposed using real-time scheduling theory with various notions of real-time tasks running under restricted resource supply environments. However, these real-time scheduling-based approaches are limited in their expressiveness in that, although capable of describing resource-demand tasks, they are unable to model resource supply. This paper describes a process algebraic framework for reasoning about resource demand *and* supply inspired by the timed process algebra ACSR. In ACSR, real-time tasks are specified by enunciating their consumption needs for resources. To also accommodate resource-supply processes we define PADS where, given a resource CPU, the complimented resource $\overline{CPU}$ denotes for availability of CPU for the corresponding demand process. Using PADS, we define a supply-demand relation where a pair (S, T) belongs to the relation if the demand process T can be scheduled under supply S. We develop a theory of compositional schedulability analysis as well as a technique for synthesizing an optimal supply process for a set of tasks. We illustrate our technique via a number of examples.

1 Introduction

The increasing complexity of real-time embedded systems demands compositional design and analysis methods for the assurance of timing requirements. Component-based design has been widely accepted as a compositional approach to facilitate the design of complex systems. It provides means for decomposing a complex system into simpler components and for composing the components using interfaces that abstract component complexities. Such approaches are increasingly used in practice for real-time systems. For example, ARINC-653 standard by the Engineering Standards for Avionics and Cabin Systems committee specifies partition-based design of avionics applications. Also, hypervisors for real-time virtual machines provide temporal partitions to guarantee real-time performance [15, 11].

To take advantage of the component-based design of real-time systems, schedulability analysis should support compositional analysis using component interfaces.

* This research was supported in part by NSF grants CNS-0834524 and CNS-0720703.

These interfaces should abstract the timing requirements of a component with a minimum resource supply that is needed to meet the resource demand of the component. Component-based real-time systems often involve hierarchical scheduling frameworks that support resource sharing among components as well as associated scheduling algorithms [5, 20]. To facilitate the analysis of such systems, resource component interfaces and their compositional analysis have been proposed [16, 21, 22, 8, 23, 12]

This paper presents a formal treatment of the problem of compositional hierarchical scheduling by introducing a process algebraic framework, PADS, for modeling resource demand and supply inspired by the timed process algebra ACSR [13, 14]. The notions of resource demand and supply are fundamental in defining the meaning of compositional real-time scheduling analysis. Our proposed algebraic framework formally defines both of these notions. As in ACSR, a task in our formalism is specified by describing its consumption needs for resources. To also accommodate resource-supply processes, we extend the notion of a *resource* and given a resource cpu we use $\overline{cpu}$ to denote the availability of the resource for consumption by a requesting task. Our formalism then addresses the following issues:

1. *Schedulability*: We define a *supply simulation relation* $\models$ that captures when a task T is schedulable by a supply S, $S \models T$.
2. *Compositionality*: We explore conditions under which we may safely compose schedulable systems. Specifically, we are interested to define functions on supplies, $\circ$, and appropriate conditions, f, such that if T_1 is schedulable by S_1 and T_2 by S_2 then the parallel composition of T_1 and T_2 is schedulable by $S_1 \circ S_2$, assuming that condition f holds:
$$\frac{S_1 \models T_1, S_2 \models T_2}{S_1 \circ S_2 \models T_1 \| T_2}, \quad f(S_1, S_2)$$
3. *Supply Synthesis*: We propose a method by which we can generate a supply process to schedule a set of tasks, assuming that such a scheduler exists. Our method is based on the notion of a *demand* of a task which is a supply that can schedule the task and, at the same time, it is optimal in the sense that (1) it does not reserve more resources than those required and (2) it captures all possibilities in which a task can be scheduled. We then prove that two or more tasks are schedulable if and only if they can be scheduled by the composition of their demands.

Related work. As mentioned above, this work brings together two long-standing lines of research. On the one hand, there has been much work on compositional hierarchical scheduling based on real-time scheduling theory [16, 21, 22, 8, 6, 7]. Typically, such approaches to schedulability analysis rely on overapproximations of task demand using, for example, demand bound functions and underapproximations of resource supply using, supply bound functions. Efficient algorithms are developed to ensure that demand never exceeds supply. On the other hand, several formal approaches to scheduling based on process algebras [3, 14, 13, 19, 18], task automata [10, 9], preemptive Petri nets [4], etc., have been developed. To the best of our knowledge, none of these approaches consider the problem of modeling resource supply explicitly. Instead, sharing of a continuously available processing resource between a set of tasks has been considered.

Our approach to supply synthesis is conceptually similar to the work of Altisen *et al.* on applying controller synthesis to scheduling problems [1, 2]. The difference is that we are not aiming to generate schedulers, but rather an *interface* for a task set, an abstraction that can be used in a component-based approach to real-time system design.

The rest of the paper is structured as follows. Section 2 presents our process algebra and its semantics. Section 3 contains our results on compositional schedulability analysis and interface construction, followed by examples illustrating the application of the theory in Section 4. Section 5 concludes the paper.

2 The Language

In our calculus, PADS (Process Algebra for Demand and Supply), we consider a system to be a set of processes operating on a set of serially reusable resources denoted by R. These processes are (1) the *tasks* of the system, which require the use of resources in order to complete their jobs, and (2) the *supplies*, that specify when each resource is available to the tasks. Based on this, each resource $r \in \mathsf{R}$ can be requested by a task, r, granted by a supply, $\overline{r}$, or consumed, $\overset{\leftrightarrow}{r}$, when a supply and a request for the resource are simultaneously available. An action in the formalism is a set containing resource requests, grants and consumptions, where each resource may be represented at most once. For example, the action $\{r_1, r_2\}$ represents a request for the resources r_1 and r_2 whereas the action $\{\overline{r_1}, \overset{\leftrightarrow}{r}_2, r_3\}$ involves the granting of resource r_1, consumption of resource r_2 and request for resource r_3. We take a discrete time approach: we assume that all actions require one unit of time to complete measured on a global clock with action $\emptyset$ representing idling for one time unit since no resource is being employed.

We write *Act*, ranged over by α and β, for the set of all actions and distinguish Act_R, the set of actions involving only resource requests, ranged over by ρ, and Act_G, the set of actions involving only resource grants, ranged over by γ. Given $\alpha \in Act$ we use the notation $\overline{\alpha}$ to reverse between resource grants and requests in action α, so, $\overline{\{r_1, \overline{r_2}, \overset{\leftrightarrow}{r}_3\}} = \{\overline{r_1}, r_2, \overset{\leftrightarrow}{r}_3\}$. Finally, we write *res*(α) for the set of resources occurring in α: *res*$(\alpha) = \{r \in \mathsf{R} | r \in \alpha \text{ or } \overline{r} \in \alpha \text{ or } \overset{\leftrightarrow}{r} \in \alpha\}$.

2.1 Syntax

The following grammars define the set of tasks, T, the set of supplies S and the set of timed systems P, where I and J are sets of indices, and I is assumed to be nonempty. Furthermore, C ranges over a set of *task constants*, each with an associated definition of the form $C \overset{\text{def}}{=} T$, where T may contain occurrences of C as well as other task constants and, D ranges over a similar set of *supply constants*.

$$\begin{aligned} T &::= \text{FIN} \mid \Sigma_{i\in I}\rho_i : T_i \mid C \\ S &::= \text{FIN} \mid \Sigma_{i\in I}\gamma_i : S_i \mid D \\ P &::= \text{FIN} \mid T \mid S \mid P \| P \mid \Sigma_{j\in J}\alpha_j : P_j \end{aligned}$$

We consider FIN to be the terminated process. Then a task process can be a terminated process, a task constant, or a nondeterministic choice $\Sigma_{i\in I}\rho_i : T_i$. The latter offers the choice of executing each of the actions ρ_i and then proceeding as T_i, where $I \neq \emptyset$. Similarly, a supply process can be a terminated process, a supply constant, or a nondeterministic choice. Finally, a process can be an arbitrary composition of tasks and supplies or a nondeterministic choice between processes $\Sigma_{j\in J}\alpha_j : P_j$.

2.2 Semantics

The semantics of PADS are given in two steps. First, we develop a transition system in which nondeterminism is resolved in all possible ways, $\twoheadrightarrow$. Then, we refine $\twoheadrightarrow$ into $\longrightarrow$ by implementing a type of "angelic" behavior in the way in which tasks resolve their nondeterminism by choosing the best possible outcome given the available supply. The rules for the first-level transition relation can be found in Table 1, whereas the second-level transition relation is subsequently defined on the basis of a preemption relation.

We proceed to consider relation $\twoheadrightarrow$ defined in Table 1. FIN being a well-terminated (and not a deadlocked) process, it allows time to pass (axiom (IDLE)). Nondeterministic choice in tasks and supplies can be resolved by executing an action and then proceeding as its continuation ((SumT) and (SumS)). A constant behaves as the process in its defining equation (Const). Finally, rule (Par) specifies the way in which a parallel system evolves. To begin with, note that the components of a parallel composition evolve synchronously in that the composition advances only if both of the constituent processes are willing to take a step. Furthermore, the rule enunciates the outcome of the synchronization between two parallel processes, the most important aspect being that a request within the one component is satisfied by an available grant in the other. The condition of rule (PAR) imposes a restriction on when two actions may take place simultaneously within a system. Specifically, we say that actions α_1 and α_2 are *compatible* with each other if, whenever r occurs in both actions then one occurrence must be a request and the other a supply of the resource. So, for example, it is not possible to simultaneously offer a resource in one component and consume or offer it in another, nor to request it by two different tasks. We capture this requirement as follows:

$$\mathsf{compatible}(\alpha_1,\alpha_2) = \bigwedge_{r\in res(\alpha_1)\cap res(\alpha_2)} (r\in\alpha_1 \wedge \overline{r}\in\alpha_2) \vee (r\in\alpha_2 \wedge \overline{r}\in\alpha_1)$$

We may now combine compatible actions by transforming a simultaneous request and supply of the same resource into a consumption:

$$\begin{aligned}\alpha_1\oplus\alpha_2 = \{r\in\alpha_1,\alpha_2 | \overline{r}\notin\alpha_1\cup\alpha_2\} \cup \{\overline{r}\in\alpha_1,\alpha_2 | r\notin\alpha_1\cup\alpha_2\} \\ \cup\{\overleftrightarrow{r} \,| r\in\alpha_i, \overline{r}\in\alpha_{(i+1)mod2}, i\in\{1,2\}\} \cup \{\overleftrightarrow{r} \mid \overleftrightarrow{r}\in\alpha_1\cup\alpha_2\}\end{aligned}$$

Note that, the associativity of the parallel composition operator with respect to $\twoheadrightarrow$ follows by the associativity of $\oplus$ which, in turn, is easy to prove by its definition.

Example 1. Consider the supply $S \overset{\text{def}}{=} \{\overline{r_1},\overline{r_2}\} : S$ and the following task processes:

$$\begin{array}{ll} T_1 \overset{\text{def}}{=} \{r_1,r_2\} : \text{FIN} + \{r_1\} : T_1 & T_2 = \{r_2\} : \text{FIN} + \{r_1,r_3\} : T_2 \\ T_3 \overset{\text{def}}{=} \{r_2\} : \text{FIN} & T_4 = \{r_1\} : \text{FIN} \end{array}$$

Table 1. Transition rules for tasks, supplies and systems

(Idle)	$\text{FIN} \overset{\emptyset}{\twoheadrightarrow} \text{FIN}$	
(SumT)	$\Sigma_{i\in I}\rho_i : T_i \overset{\rho_i}{\twoheadrightarrow} T_i$	$I \neq \emptyset$
(SumS)	$\Sigma_{i\in I}\gamma_i : S_i \overset{\gamma_i}{\twoheadrightarrow} S_i$	$I \neq \emptyset$
(Const)	$\dfrac{P \overset{\alpha}{\twoheadrightarrow} P'}{C \overset{\alpha}{\twoheadrightarrow} P'}$ $\quad C \overset{\text{def}}{=} P$	
(Par)	$\dfrac{P_1 \overset{\alpha_1}{\twoheadrightarrow} P_1' \quad P_2 \overset{\alpha_2}{\twoheadrightarrow} P_2'}{P_1 \| P_2 \overset{\alpha_1 \oplus \alpha_2}{\twoheadrightarrow} P_1' \| P_2'}$	$\mathsf{compatible}(\alpha_1, \alpha_2)$
(SumP)	$\Sigma_{j\in J}\alpha_j : P_j \overset{\alpha_j}{\twoheadrightarrow} P_j$	$J \neq \emptyset$

We have:

$$T_1 \| S \overset{\{\overleftrightarrow{r_1}, \overleftrightarrow{r_2}\}}{\twoheadrightarrow} \text{FIN} \| S \qquad (1) \qquad\qquad T_1 \| S \overset{\{\overleftrightarrow{r_1}, \overline{r_2}\}}{\twoheadrightarrow} T_1 \| S \qquad (2)$$

$$T_2 \| S \overset{\{\overline{r_1}, \overleftrightarrow{r_2}\}}{\twoheadrightarrow} \text{FIN} \| S \qquad (3) \qquad\qquad T_2 \| S \overset{\{\overleftrightarrow{r_1}, \overline{r_2}, r_3\}}{\twoheadrightarrow} T_2 \| S \qquad (4)$$

$$(T_1 \| S) \| T_3 \overset{\{\overleftrightarrow{r_1}, \overleftrightarrow{r_2}\}}{\twoheadrightarrow} (T_1 \| S) \| \text{FIN} \qquad (5)$$

Note that $(T_1 \| S) \| T_3$ has no transition other than (5) above, while $(T_1 \| S) \| T_4$ has no transitions altogether since both T_1 and T_4 require r_1 during the first time unit. $\square$

Moving on to the second level of the semantics, we employ a preemption relation on actions to prune away all transitions that do not represent correctly the behavior of a system, as we would expect it. In particular, we make the following two assumptions:

1. Given a supply, a task should respond angelically and, given a nondeterministic set of transitions by which it can evolve, it should choose only between the ones that are satisfied by the available suply, assuming that such options exist. For example, $T_2 \| S$ above should retain only transition (3) out of the available (3) and (4).
2. In addition, we assume that a task behaves greedily and, at each step, it employs as many of the supplied resources as possible. For example, the composition $T_1 \| S$ in Example 1 should only retain transition (1) out of transitions (1) and (2).

Given the above, we define the *preemption relation* $\prec$ so that $\alpha \prec \beta$ if one of the following hold:

1. $\{r | \overline{r} \in \alpha, \overleftrightarrow{r} \in \alpha\} = \{r | \overline{r} \in \beta, \overleftrightarrow{r} \in \beta\}$, $\alpha \cap \mathsf{R} \neq \emptyset$ and $\beta \cap \mathsf{R} = \emptyset$,
2. $\alpha \cap \mathsf{R} = \beta \cap \mathsf{R} = \emptyset$, $\{r | \overline{r} \in \alpha, \overleftrightarrow{r} \in \alpha\} = \{r | \overline{r} \in \beta, \overleftrightarrow{r} \in \beta\}$ and $\{r | \overleftrightarrow{r} \in \alpha\} \subseteq \{r | \overleftrightarrow{r} \in \beta\}$.

Intuitively, an action precludes another if it makes better usage of the same offered resources. In particular, an action β preempts an action α, if, either α and β concern the same offered resources (granted or consumed) but α, unlike β, also contains some unsatisfied resource requests (condition (1)), or α and β contain only the same granted and consumed resources but β consumes more resources than α (condition (2)).

We may now define the relation $\xrightarrow{\alpha}$ by the following rule:

$$\frac{P \overset{\alpha}{\twoheadrightarrow} Q}{P \xrightarrow{\alpha} Q}, \quad \text{there is no } P \overset{\beta}{\twoheadrightarrow}, \alpha \prec \beta$$

We conclude this section by introducing some notations. We write $P \longrightarrow$ if there exists action α such that $P \xrightarrow{\alpha}$. If $P \not\xrightarrow{\alpha}$ for all actions α, we write $P = \delta$, where δ is the deadlocked process. We write $P \Longrightarrow P'$ if there exist actions $\alpha_1 \ldots \alpha_n$ and processes $P_1, \ldots, P_{n-1}$ such that $P \xrightarrow{\alpha_1} P_1 \xrightarrow{\alpha_2} \ldots P_{n-1} \xrightarrow{\alpha_n} P'$. The set of traces of P, $\mathsf{traces}(P)$, is defined to be the set of all infinite sequences of actions $\alpha_1\alpha_2\ldots$ such that $P \xrightarrow{\alpha_1} P_1 \xrightarrow{\alpha_2} \ldots$. Finally, for $w = \alpha_1\alpha_2\ldots$, we write $\overline{w}$ for $\overline{\alpha_1}\,\overline{\alpha_2}\ldots$

3 Schedulability

In this section we present a theory of schedulability for our calculus. We begin by defining when a set of tasks is considered to be schedulable by a supply. Then we present an alternative characterization based on a type of simulation relations and we prove the two definitions to be equivalent. In what follows we write T^* for the set containing all processes of the form $T_1\|\ldots\|T_n$, $n \geq 1$, and S^* for the set containing all processes of the form $S_1\|\ldots\|S_n$, $n \geq 1$. For simplicity, we refer to elements of T^* and S^* simply as tasks and supplies, respectively.

Definition 1. *A task $T \in \mathsf{T}^*$ is* schedulable under *supply $S \in \mathsf{S}^*$ if whenever $T\|S \Longrightarrow P$ then $P \neq \delta$ and for all $P \xrightarrow{\alpha}$ we have $\alpha \cap \mathsf{R} = \emptyset$.*

According to this definition, a task T is schedulable under supply S if at no point during their interaction does the system deadlock and, moreover, no request for a resource remains unsatisfied.

Example 2. Here follow some examples relating to the above definition.

- $T \overset{\text{def}}{=} \{r\}$:FIN is not schedulable under $S \overset{\text{def}}{=} \{\overline{r'}\}$:FIN. We have $T\|S \xrightarrow{\{r,\overline{r'}\}}$ and the definition is violated.
- $T \overset{\text{def}}{=} \{r\}$:FIN is schedulable under $S \overset{\text{def}}{=} \{\overline{r}, \overline{r'}\}$:FIN. We have $T\|S \xrightarrow{\{\overleftrightarrow{r},\overline{r'}\}}$ FIN$\|$FIN which satisfies the definition.
- $T \overset{\text{def}}{=} \{r\}$:FIN $+ \emptyset$:$\{r\}$:FIN is schedulable under $S \overset{\text{def}}{=} \{\overline{r}\}$:FIN. The composition of the two processes has only one possible transition $T\|S \xrightarrow{\{\overleftrightarrow{r}\}}$ FIN$\|$FIN. Note that the transition $T\|S \overset{\{\overline{r}\}}{\twoheadrightarrow} \{r\}$:FIN$\|$FIN at the lower-level of the semantics is pruned by the preemption relation. Thus, the definition is satisfied. The same holds for $T \overset{\text{def}}{=} \{r\}$:FIN$+\{r'\}$:FIN and $S \overset{\text{def}}{=} \{\overline{r}\}$:FIN since $\{r', \overline{r}\} \prec \{\overleftrightarrow{r}\}$. This illustrates that as long as there is some possible way of scheduling a task's requirements by an available supply, the task is considered to be schedulable by the supply. □

Before moving on to our alternative schedulability definition we introduce the following useful notation: For $T \in \mathsf{T}^*$ and $\alpha, \beta \in \mathit{Act}$, we write, $\beta \trianglelefteq_T \alpha$, if there exists no γ such that $T \xrightarrow{\gamma} T'$ and $\beta \subset \gamma \subseteq \alpha$.

Definition 2. *A relation $\mathcal{S} \subseteq \mathsf{T}^* \times \mathsf{S}^*$ is a* supply simulation relation *if for all $(T, S) \in \mathcal{S}$, $S \longrightarrow$, and, if $S \xrightarrow{\alpha} S'$ then*

1. *there exists $T \xrightarrow{\beta} T'$ with $\beta \subseteq \overline{\alpha}$ and $(T', S') \in \mathcal{S}$, and*
2. *if $T \xrightarrow{\beta} T'$ with $\beta \trianglelefteq_T \overline{\alpha}$, then $(T', S') \in \mathcal{S}$.*

If there exists a supply relation between T and S, then we write $S \models T$ and we say that S schedules *T.*

That is, a task and a supply are related by a supply simulation relation if (i) the supply is able to offer resources to the task ($S \longrightarrow$), (ii) if a supply offers a set of resources then the task will be able to respond by employing some (or all) of these resources and remain schedulable by the resulting state of the supply (clause 1), and (iii) given a set of resources offered by the supply, any maximal transition by which the task can accept the offered supply will result in a state that remains schedulable by the remaining supply (clause 2). Here, by a *maximal response* of the task, we mean all greedy transitions β by which the task can employ the offered resources α, that is, where $\beta \trianglelefteq_T \overline{\alpha}$. Note that any non-maximal transition of T taking place as a response to $S \xrightarrow{\alpha}$ would be subsequently pruned in the composition $S \| T$ as it would be preempted by greedier responses of T. Therefore, such transitions can be ignored.

We may now prove that the two alternative ways of defining schedulability of a task by a supply coincide.

Lemma 1. *A task $T \in \mathsf{T}^*$ is schedulable under supply S if and only if $S \models T$.*

PROOF: To begin with, suppose there exists a supply simulation relation $\mathcal{R}$ between T and S. We will show that if $S \| T \xrightarrow{\alpha} S' \| T'$ then $S' \| T' \neq \delta$, $\alpha \cap \mathsf{R} = \emptyset$ and $(S', T') \in \mathcal{R}$. So suppose that $S \| T \xrightarrow{\alpha} S' \| T'$, $S \xrightarrow{\alpha_1} S'$ and $T \xrightarrow{\alpha_2} T'$, $\alpha = \alpha_1 \oplus \alpha_2$. We know that for some $\beta \subseteq \overline{\alpha_1}$, $T \xrightarrow{\beta} T''$ (Definition 2(1)). This implies that $\alpha_2 \subseteq \overline{\alpha_1}$ (otherwise $\alpha_1 \oplus \alpha_2 \prec \alpha_1 \oplus \beta$ and $S \| T \not\xrightarrow{\alpha}$). Consequently, we deduce that $\alpha \cap \mathsf{R} = \emptyset$. In addition, since T' is schedulable by S', by Definition 2 we have that $S' \longrightarrow$ and for each $S' \xrightarrow{\beta_1}$ there exists $T' \xrightarrow{\beta_2}$ such that $S' \| T' \longrightarrow$, that is, $S' \| T' \neq \delta$. And, finally, we may observe that there is no $T \xrightarrow{\gamma}$, $\alpha_2 \subset \gamma \subseteq \overline{\alpha_1}$ (otherwise $\alpha_1 \oplus \alpha_2 \prec \gamma \oplus \alpha_2$), which, by Definition 2(2), implies that $(S', T') \in \mathcal{S}$.

Conversely, suppose that task T is schedulable by supply S. We will show that $\mathcal{R} = \{(S, T) | S \text{ is schedulable by } T\}$ is a supply simulation relation. Suppose $(S, T) \in \mathcal{R}$. Since $S \| T \neq \delta$, $S \longrightarrow$. Furthermore, if $S \xrightarrow{\alpha} S'$ then $T \longrightarrow$. Since T is schedulable by S, there exists $T \xrightarrow{\beta} T'$, $\beta \subseteq \overline{\alpha}$. If not, that is for all $T \xrightarrow{\gamma} T''$, $\gamma \cap \overline{\alpha} \neq \gamma$, then $S \| T \xrightarrow{\alpha'}$, $\alpha' \cap \mathsf{R} \neq \emptyset$ which contradicts our assumption of T being schedulable by S. Next, suppose that $T \xrightarrow{\beta} T'$, $\beta \subseteq \alpha$ and for no $\beta \subset \gamma \subseteq \overline{\alpha}$. Then, clearly, $S \| T \xrightarrow{\alpha \oplus \beta} S' \| T'$, where T' is schedulable by S', which implies that $(S', T') \in \mathcal{R}$, as required. □

We define when a task is schedulable and this is done in the following obvious way.

Definition 3. *A task $T \in \mathsf{T}^*$ is* schedulable *if there exists a supply S with $S \models T$.*

We observe that the crux of the schedulability of a task by a supply lies in the capability of the task to operate acceptably for all possible behaviors of the supply. Furthermore, at each point during its execution and for each supply provision of the supplier, the task must behave well in all its nondeterministic executions that can take place by employing the resources available. The notion of a cylinder, defined below is intended to capture the relevant executions of the task given a behavior of the supply.

Definition 4. *Given a task $T \in \mathsf{T}^*$ and an infinite trace $w = \alpha_1\alpha_2\ldots$, we define the* w-cylinder *of T to be the set $A = \cup_{i\geq 1} A_i$, where*

$$\begin{aligned} A_1 &= \{(T, \alpha_1, P_1) | T \xrightarrow{\alpha_1} P_1\} \\ A_i &= \{(P_i, \beta_i, P_i') | P_i \xrightarrow{\beta_i} P_i', \beta_i \trianglelefteq_{P_i} \alpha_i, \exists (Q, \gamma, P_i) \in A_{i-1}\}, \quad i > 1 \end{aligned}$$

Furthermore, we say that an w-cylinder $A = \cup_{i\geq 1} A_i$ is live *if (i) A contains no triple of the form (Q, α, δ), (ii) $A_i \neq \emptyset$ for all i and (3) $\bigcup_{(P,\beta,Q)\in A_i} \beta = \alpha_i$.*

Lemma 2. *A task $T \in \mathsf{T}^*$ is schedulable if and only if it possesses a live cylinder.*

PROOF: Suppose T has a live w-cylinder where $w = \alpha_1\alpha_2\ldots$. Consider supply S_0 defined by the following set of equations $S_i \stackrel{\text{def}}{=} \overline{\alpha_{i+1}}{:}S_{i+1}$. Then, we may confirm that $S_0 \models T$. The details are omitted. On the other hand, if T is schedulable, then there exists a supply S that schedules it. If we consider a trace $w = \overline{\alpha_1}\,\overline{\alpha_2}\ldots$ of S, we may construct an associated cylinder of T and confirm that it is live. □

3.1 Matching Supplies to Tasks

In this section we focus our attention to the problem of collecting the resource requirements of a task into a matching supply. Specifically, given a task, we would like to generate a supply process which schedules the task and at the same time is optimal in that (1) it does not reserve more resources than those required by the task and (2) it provides all the alternative resource assignments in which the task can be scheduled. Both of these properties are important during the compositional scheduling of real-time tasks. The first property is clearly desirable since conservation of resources becomes critical when real-time components are composed. For the second property, we observe that capturing all possible ways of scheduling a task gives flexibility when one tries to compositionally schedule a set of tasks where the challenge is to share the resources between the tasks in ways that are acceptable to each one of them.

We begin by defining a function on combining supplies. This is helpful for a subsequent definition that considers matching supplies to tasks.

Definition 5. *Given supplies S_1 and S_2 we define $S_1 \otimes S_2 =$*

$$S_1 \otimes S_2 = \begin{cases} S_1 & \textit{if } S_2 = \text{FIN} \\ S_2 & \textit{if } S_1 = \text{FIN} \\ \varSigma_{i\in I}\varSigma_{j\in J}\ \alpha_i \cup \beta_j{:}(\bigotimes_{k\in I, \alpha_k \trianglelefteq_{S_1} \alpha_i\cup\beta_j} P_k \otimes \bigotimes_{l\in J, \beta_l \trianglelefteq_{S_2} \alpha_i\cup\beta_j} Q_l) & \\ & \textit{if } S_1 \stackrel{\text{def}}{=} \sum_{i\in I} \alpha_i{:}P_i \textit{ and } S_2 \stackrel{\text{def}}{=} \sum_{j\in J} \beta_i{:}Q_i \end{cases}$$

Essentially, the joined supply $S_1 \otimes S_2$, joins together the various summands of the individual supplies as follows: in its topmost summand it unites all available grants of S_1 with all available grants of S_2, while the continuation process consists of the join of those continuations of S_1 and S_2 which appear after "maximal" subsets of the initial action in question. For example we have:

$$\emptyset : \{\overline{cpu}\} : \emptyset : \mathrm{FIN} \otimes \emptyset : \emptyset : \{\overline{cpu}\} : \mathrm{FIN} \quad = \quad \emptyset : \{\overline{cpu}\} : \{\overline{cpu}\} : \mathrm{FIN}$$
$$\emptyset : \{\overline{cpu}\} : \emptyset : \mathrm{FIN} \otimes (\emptyset : \emptyset : \{\overline{cpu}\} : \mathrm{FIN} + \{\overline{cpu}\} : \emptyset : \emptyset : \mathrm{FIN})$$
$$= \quad \emptyset : \{\overline{cpu}\} : \{\overline{cpu}\} : \mathrm{FIN} + \{\overline{cpu}\} : \{\overline{cpu}\} : \emptyset : \mathrm{FIN}$$

Using this definition we now move to define the *demand* of a task. The demand of a task is intended to capture the optimal supply that can schedule a task in the sense we have already discussed. The main point to note in this definition is that we combine all same-prefixed nondeterministic choices of a task by a singly-prefixed supply.

Definition 6. *Given a task $T \overset{\text{def}}{=} \sum_{i\in I} \alpha_i{:}T_i$, we define its* demand *as follows:*

$$\mathsf{demand}(T) \overset{\text{def}}{=} \Sigma_{i\in I}\overline{\alpha_i}{:} \bigotimes_{j\in I, \alpha_i=\alpha_j} \mathsf{demand}(T_j)$$

Example 3. Consider tasks

$$T_1 = \{cpu\} : \emptyset : T_1 + \emptyset : \{cpu\} : T_1$$
$$T_2 = \{cpu\} : \emptyset : \emptyset : T_2 + \emptyset : \{cpu\} : \emptyset : T_2 + \emptyset : \emptyset : \{cpu\} : T_2$$
$$T_3 = \{cpu\} : \emptyset : \emptyset : T_3 + \emptyset : (\{cpu\} : \emptyset : T_3 + \emptyset : \{cpu\} : T_3)$$

Their demands are given by X_1, X_2 and X_3 below, respectively.

$$X_1 = \{\overline{cpu}\} : \emptyset : X_1 + \emptyset : \{\overline{cpu}\} : X_1$$
$$X_2 = \{\overline{cpu}\} : \emptyset : \emptyset : X_2 + \emptyset : \{\overline{cpu}\} : \{\overline{cpu}\} : X_2$$
$$X_3 = \{\overline{cpu}\} : \emptyset : \emptyset : X_3 + \emptyset : (\{\overline{cpu}\} : \emptyset : X_3 + \emptyset : \{\overline{cpu}\} : X_3)$$

□

The next lemma considers the optimality of $\mathsf{demand}(T)$ following the requirements posed at the beginning of this section. We write $w' \leq w$ for the infinite traces $w' = \alpha_1\alpha_2\ldots$ and $w = \beta_1\beta_2\ldots$, if either $w' = w$, or there exists j such that $\alpha_j \subset \beta_j$ and $\alpha_i = \beta_i$ for all $1 \leq i < j$.

Lemma 3. *A task T possesses a live w-cylinder if and only if there exists $w' \leq \overline{w}$ such that $w' \in \mathsf{traces}(\mathsf{demand}(T))$.*

PROOF: Suppose $\mathsf{demand}(T) \xrightarrow{\overline{\alpha_1}} T_1 \xrightarrow{\overline{\alpha_2}} T_2 \xrightarrow{\overline{\alpha_3}} \ldots$. We may show that for the w-cylinder $A = \cup_{i\geq} A_i$, where $w = \alpha_1\alpha_2\ldots$ we have $T_i = \bigotimes_{(P,\beta,Q)\in A_i} \mathsf{demand}(Q)$ and A is live. The details are omitted.

To establish the opposite direction suppose $A = \cup_{i\geq} A_i$ is a live w-cylinder of T. Now consider the w'-cylinder of T, $B = \cup_{i\geq} B_i$, where $w' = \beta_1\beta_2\ldots$ is defined such that $\beta_1 = \alpha_1$, $B_1 = A_1$ and $\beta_i \in \{\gamma_1 \cup \ldots \cup \gamma_n | P_i \xrightarrow{\gamma_i}, P_i \in \{P | (P, \alpha, Q) \in B_{i-1}\}\}$, $B_i = \{(P, \alpha, Q) | P \xrightarrow{\alpha} Q, \alpha \trianglelefteq_P \beta_i, \exists (Q, \gamma, P) \in B_{i-1}\}$. We may now prove, that $w' \in \mathsf{traces}(\mathsf{demand}(T))$ and, therefore, that it is a live cylinder of T. $\square$

As a consequence of the result we conclude that a task T is schedulable by its demand. Furthermore, $\mathsf{demand}(T)$ may schedule all cylinders of T and it schedules them exactly, i.e. it offers exactly the resources that are necessary for the scheduling.

3.2 Compositional Theory

We proceed to consider the schedulability problem of a set of tasks. The first issue we tackle is the compositionality problem: If T_1 is schedulable by S_1 and T_2 by S_2 can we combine S_1 and S_2 into a supply that schedules $T_1 \| T_2$? We begin by noting a certain subtlety in this problem which we need to consider while answering it.

Consider the tasks

$$T_1 = \{r\}{:}\emptyset{:}\mathrm{FIN} + \emptyset{:}\{r\}{:}\mathrm{FIN} \text{ and } T_2 = \{r\}{:}\emptyset{:}\mathrm{FIN} + \emptyset{:}\{r\}{:}\{r\}{:}\mathrm{FIN}.$$

These tasks are schedulable under supplies $S_1 = \emptyset{:}\{\overline{r}\}{:}\mathrm{FIN}$ and $S_2 = \{\overline{r}\}{:}\emptyset{:}\mathrm{FIN}$, respectively. That is, it is sufficient for task T_1 to obtain resource r during the second time unit and for task T_2 during the first time unit. However, a supply $S = \{\overline{r}\}{:}\{\overline{r}\}{:}\mathrm{FIN}$, offering r during both time units, fails to schedule $T_1 \| T_2$. This is due to the fact that the supply for resource r during the first time unit is intended for task T_2 but may be consumed by task T_1 leading to a deadlock of the system during the third time unit.

To resolve this issue, we associate tasks with their matching supplies by annotating each resource reference by a number which distinguishes the task in which the resource is employed/supplied. Precisely, we assume that each task is associated with a resource identity and if resource r is requested by a task with identifier i we write $r[i]$ for the request and, similarly, if a supply of r is intended for the task with identifier i we write $\overline{r[i]}$ for the supply. So, we say that task $\{r[1]\}{:}\mathrm{FIN}$ is schedulable by supply $\{\overline{r[1]}\}{:}\mathrm{FIN}$ and task $\{r[2]\}{:}\mathrm{FIN}$ by supply $\{\overline{r[2]}\}{:}\mathrm{FIN}$. However, note that resources $r[1]$ and $r[2]$ do refer to the same resource and for all other purposes should be treated as the same. So, for example, $\{r[1]\} \cap \{r[2]\} \neq \emptyset$. To model this precisely we write:

- $P[i]$ for the process P with all its resources r renamed as $r[i]$.
- $\alpha \cap_R \beta$ for $\{r \in R | r[i] \in \alpha, r[j] \in \beta, \text{ or } \overline{r[i]} \in \alpha, \overline{r[j]} \in \beta\}$

Furthermore, we use the notation $\alpha[i] = \{r | r[i] \in \alpha\}$ and, if $w = \alpha_1\alpha_2\ldots$, $w[i] = \alpha_1[i]\alpha_2[i]\ldots$. We have the following result:

Lemma 4. *If T_1 is schedulable by S_1, T_2 is schedulable by S_2 and $S_1 \| S_2$ does not deadlock, then $T_1[1] \| T_2[2]$ is schedulable by $S_1[1] \| S_2[2]$.*

PROOF: We will show that $\mathcal{R}$, below, is a supply simulation relation.

$$\mathcal{R} = \{(T_1[1] \| T_2[2], S_1[1] \| S_2[2]) | S_1 \models T_1, S_2 \models T_2, S_1[1] \| S_2[2] \text{ does not deadlock}\}$$

Let $(T_1[1] \| T_2[2], S_1[1] \| S_2[2]) \in \mathcal{R}$. By the definition of $\mathcal{R}$, $S_1[1] \| S_2[2] \longrightarrow$. So consider $S_1[1] \| S_2[2] \xrightarrow{\alpha} S_1'[1] \| S_2'[2]$. It must be that $\alpha = \alpha_1[1] \oplus \alpha_2[2]$, where $S_1 \xrightarrow{\alpha_1} S_1'$,

$S_2 \xrightarrow{\alpha_2} S_2'$ and $\alpha_1 \cap \alpha_2 = \emptyset$. Since $S_1 \models T_1$, $S_2 \models T_2$, we have $T_1 \xrightarrow{\beta_1} T_1'$, $S_1' \models T_1'$, and similarly $T_2 \xrightarrow{\beta_2} T_2'$, $S_2' \models T_2'$. In fact, for all $T_1 \xrightarrow{\beta_1} T_1'$, $\beta_1 \trianglelefteq_{T_1} \overline{\alpha_1}$, it holds that $S_1' \models T_1'$, and for all $T_2 \xrightarrow{\beta_2} T_2'$, $\beta_2 \trianglelefteq_{T_2} \overline{\alpha_2}$, it holds that $S_2' \models T_2'$. This implies that for all $T_1[1]\|T_2[2] \xrightarrow{\beta} T_1'[1]\|T_2'[2]$, $\beta \trianglelefteq_{T_1[1]\|T_2[2]} \overline{\alpha}$, $(T_1'[1]\|T_2'[2], S_1'[1]\|S_2'[2]) \in \mathcal{R}$ and there exists at least one such α-transition. This completes the proof. □

However, note that even if $S_1\|S_2$ deadlocks, it is still possible that the schedules S_1 and S_2 can be combined to produce a schedule for $T_1\|T_2$. In particular, we may suspect that every infinite trace of $S_1\|S_2$ is capable of scheduling $T_1\|T_2$, and in fact we can show that the part of the transition system that pertains to non-deadlocking behavior achieves exactly that. The following operator on supplies extracts this type of behavior.

$$S_1 \times S_2 = \begin{cases} S_1 & \text{if } S_2 = \text{FIN} \\ S_2 & \text{if } S_1 = \text{FIN} \\ (\alpha \cup \beta){:}(S_1' \times S_2') & \text{if } S_1 = \alpha{:}S_1', S_2 = \beta{:}S_2', \alpha \cap \beta = \emptyset, S_1' \times S_2' \neq \delta \\ \delta & \text{if } S_1 = \alpha{:}S_1', S_2 = \beta{:}S_2', \alpha \cap \beta \neq \emptyset \text{ or } S_1' \times S_2' = \delta \\ \Sigma_{i\in I, j\in J}(S_1^i \times S_2^j) & \text{if } S_1 = \Sigma_{i\in I} S_1^i, S_2 = \Sigma_{j\in J} S_2^j \end{cases}$$

Note that the set of recursive equations used in the definition of $S_1 \times S_2$ may allow more than one solution. Consider, for example, $S_1 = \{\overline{r_1}\} : S_1$ and $S_2 = \{\overline{r_2}\} : S_2$. It is easy to see that $S_1 \times S_2 = \delta$ is a trivial solution. However, we are interested in the maximal solution to this set of equations, which in this case is $S_1 \times S_2 = \{\overline{r_1}, \overline{r_2}\} : S_1 \times S_2$. For finite-state processes, the maximal solution can be computed iteratively. Due to space restrictions the proof is omitted.

It is easy to see that, if $S_1\|S_2$ does not deadlock then $S_1 \times S_2 \neq \delta$. However, the opposite is not true. By the construction of $\times$, $S_1 \times S_2$ selects the part of the transition system of $S_1\|S_2$ that does not lead to deadlocked states. For example, consider

$$S_1 \overset{\text{def}}{=} \{r\}{:}\{r\}{:}\text{FIN} + \emptyset{:}\{r\}\{r\}{:}\text{FIN} \quad \text{and} \quad S_2 \overset{\text{def}}{=} \emptyset{:}\{r\}{:}\text{FIN} + \{r\}{:}\emptyset{:}\text{FIN}$$

Then, although $S_1\|S_2 \xrightarrow{\{r\}} \{r\}{:}\text{FIN}\|\{r\}{:}\text{FIN} = \delta$, $S_1 \times S_2 = \{r\}{:}(\{r\} : \{r\}{:}\text{FIN} \times \emptyset{:}\text{FIN})$, and $(\{r\}\{r\}{:}\text{FIN} \times \emptyset{:}\text{FIN}) = \{r\}\{r\}{:}\text{FIN}$.

Lemma 5. *If T_1 is schedulable by S_1, T_2 is schedulable by S_2 and $S_1 \times S_2 \neq \delta$, then $T_1[1]\|T_2[2]$ is schedulable by $S_1[1] \times S_2[2]$.*

PROOF: The proof is similar to that of the previous lemma. □

At this point we turn our attention to the problem of constructing an interface for a set of mutually schedulable tasks. To do this, we employ the notion of demands and we prove the following:

Lemma 6. *If $T_1[1]\|T_2[2]$ has a live w-cylinder then there exists a trace $w' \leq \overline{w}$ such that $w' \in \mathsf{traces}(\mathsf{demand}(T_1[1]) \times \mathsf{demand}(T_2[2]))$.*

PROOF: Suppose that the w-cylinder of $T_1[1]\|T_2[2]$ is live. It is easy to see that $w[1]$ and $w[2]$ give rise to live cylinders in $T_1[1]$ and $T_2[2]$. Then, by Lemma 3, there exist $w_1 \leq \overline{w[1]}$ and $w_2 \leq \overline{w[2]}$ such that $w_1 \in \mathsf{traces}(\mathsf{demand}(T_1[1]))$ and $w_2 \in$

$\mathsf{traces}(\mathsf{demand}(T_2[2]))$. This implies that, $w_1 \cup w_2 \leq \overline{w}$ is a trace of $\mathsf{demand}(T_1[1]) \times \mathsf{demand}(T_2[2])$, as required. □

This result implies that all alternatives of scheduling $T_1[1] \| T_2[2]$ will be explored by $\mathsf{demand}(T_1[1]) \times \mathsf{demand}(T_2[2])$. It can be extended to the composition of an arbitrary number of tasks. We are now ready to present our main theorem:

Theorem 1. *$T_1 \| T_2$ is schedulable if and only if $\mathsf{demand}(T_1[1]) \times \mathsf{demand}(T_2[2]) \neq \delta$. Moreover, if it is schedulable, then it is schedulable by $\mathsf{demand}(T_1[1]) \times \mathsf{demand}(T_2[2])$.*

PROOF: Suppose $T_1[1] \| T_2[2]$ is schedulable. Then, by Lemma 2 it has a live w-cylinder. Consequently, by Lemma 6 there is a trace $w' \leq w$ such that the w' is a trace of $\mathsf{demand}(T_1[1]) \times \mathsf{demand}(T_2[2])$. This implies that $\mathsf{demand}(T_1[1]) \times \mathsf{demand}(T_2[2]) \neq \delta$. On the other hand, if $\mathsf{demand}(T_1[1]) \times \mathsf{demand}(T_2[2]) \neq \delta$, then, since, additionally, $\mathsf{demand}(T_1[1])$ schedules $T_1[1]$ and $(T_2[2])$ schedules $T_2[2]$, then, by Lemma 5, $T_1[1] \| T_2[2]$ is schedulable by $\mathsf{demand}(T_1[1]) \times \mathsf{demand}(T_2[2])$. □

Based on this result we may determine the schedulability and a related scheduler for a set of tasks $T_1, \ldots, T_n$, as follows: For each task, extract its demand and compute the combinations $D_1 = \mathsf{demand}(T_1) \times \mathsf{demand}(T_2)$, $D_2 = D_2 \times \mathsf{demand}(T_3), \ldots$. If this process does not reduce to some $D_i = \delta$ then the tasks are schedulable by D_{n-1}. Furthermore, according to Theorem 1, if they are indeed schedulable then $D_{n-1} \neq \delta$. Thus, this method is guaranteed to produce a schedule if one exists.

4 Examples

Example 4. We first define a simple periodic task with period p and execution time w, $Task_{w,p} = T_{0,0,w,p}$, as follows:

$$T_{e,t,w,p} = \begin{cases} \emptyset : T_{e,t+1,w,p} & \text{if } e = w, t < p \\ T_{0,0,w,p} & \text{if } e = w, t = p \\ \emptyset : T_{e,t+1,w,p} + \{r\} : T_{e+1,t+1,w,p} & \text{if } e < w, w - e < p - t \\ \{r\} : T_{e+1,t+1,w,p} & \text{if } e < w, w - e = p - t \end{cases}$$

Note that in our definition, the task cannot idle if idling will make it miss the deadline. If the supply can avoid giving the resource to the task in this case, the system will have an unmet resource request transition that signals non-schedulability (by Definition 1). Let us consider an instance of a classical scheduling problem for a set of periodic tasks running on a single processor resource: $Task_{2,3} \| Task_{2,7} \| S$, where $S = \{\overline{r}\} : S$. In the figure below, we show the initial part of the state space of the example. Each state is represented as a tuple $ij|km$, where i and j are the first two parameters of the first task and k and m are the first two parameters of the second task. The other two parameters do not change and are omitted to avoid cluttering the figure. We also omit labels on the transitions: all transitions are labeled by $\{\overset{\leftrightarrow}{r}\}$.

$$\begin{tikzcd}
00 \mid 00 \arrow[r] \arrow[dr] & 11 \mid 01 \arrow[r] \arrow[dr] & 22 \mid 02 \arrow[r] & 00 \mid 13 \arrow[r] \arrow[dr] & 11 \mid 14 \arrow[r] \arrow[dr] & 22 \mid 15 \arrow[r] & 00 \mid 26 \arrow[r] & 11 \mid 00 \arrow[r] \arrow[dr] & 22 \mid 01 \arrow[r] & 00 \mid 12 \arrow[r] \arrow[dr] & 11 \mid 13 \arrow[r] \arrow[dr] & \cdots \\
& 01 \mid 11 \arrow[r] & 12 \mid 12 \arrow[ur] & & 01 \mid 24 \arrow[r] & 12 \mid 25 \arrow[ur] & & & 12 \mid 11 \arrow[ur] & & 01 \mid 23 \arrow[r] & \cdots
\end{tikzcd}$$

The tasks are schedulable according to the Definition 1 and the transition system of the composite process, shown above, can be seen as the specification of feasible schedulers for the task set. Non-determinism in the transition system represent different decisions that a scheduler can make. For example, the trace along the top of the figure corresponds to the rate-monotonic scheduling policy, which gives priority to $Task_{2,3}$ as it has the smallest period.

We now consider the demand of a periodic task defined above. It is easy to see that the task process is *resource-deterministic*, that is, its behavior is determined by the availability of resources. For a resource-deterministic task, the demand is obtained by a straightforward replacement of requested resources by matching offered resources. Thus, $\mathsf{demand}(Task_{w,p}) = X_{0,0,w.p}$ is defined below:

$$X_{e,t,w,p} = \begin{cases} \emptyset : X_{e,t+1,w,p} & \text{if } e = w, t < p \\ X_{0,0,w,p} & \text{if } e = w, t = p \\ \emptyset : X_{e,t+1,w,p} + \{\overline{r}\} : X_{e+1,t+1,w,p} & \text{if } e < w, w - e < p - t \\ \{\overline{r}\} : X_{e+1,t+1,w,p} & \text{if } e < w, w - e = p - t \end{cases}$$

It is easy to check that $\mathsf{demand}(Task_{2,3}) \| \mathsf{demand}(Task_{2,7})$ does not deadlock and thus can schedule the two tasks according to Lemma 4.

Let us now consider a task with variable execution time which takes between b and w time units to complete: $Task^{v}_{b,w,p} = Task_{b,p} + Task_{b+1,p} + \ldots + Task_{w,p}$. One can see that $\mathsf{demand}(Task^{v}_{b,w,p}) = \mathsf{demand}(Task_{w,p})$. This observation matches the well-known fact from the real-time systems theory that for independent periodic tasks it is sufficient to consider worst-case execution time of each task [17].

Example 5. To illustrate compositional analysis with partial supplies, we begin with a simple example of time-partitioned supplies that are widely used in practice. Consider a periodic time partition with period P, duration $D \leq P$, and relative start time t_0, which essentially offers a resource r for the interval $[t, t + D)$ during each period: $Part_{t_0,D,P} = P_{0,t_0,D,P}$ is defined as follows where, again, addition is modulo P:

$$P_{t,t_0,D,P} = \begin{cases} \{\overline{r}\} : P_{t+1,t_0,D,P} & \text{if } t_0 \leq t < t_0 + D \\ \emptyset : P_{t+1,t_0,D,P} & \text{otherwise} \end{cases}$$

It is clear that partitions with the same period and non-overlapping service intervals $[t, t + D)$ do not conflict. We can now analyze schedulability of tasks allocated to a partition separately from any other task in the system. It is, for example, trivial to see that partition $Part_{t_0,D,P}$ can schedule a task $Task_{D,P}$ for any t_0.

We can similarly define more complex partial supplies. Consider, for example, compositional scheduling based on periodic resource models [21, 22]. A periodic resource model is a supply that guarantees w units of resource execution within a period P, however, the availability of the resource within the period is unknown *a priori*. We can straightforwardly model a periodic resource model as $PRM_{w,P} = \mathsf{demand}(Task_{w,P})$. We can then analyze whether a set of tasks is schedulable with respect to this supply. This analysis will not be limited to independent periodic or sporadic tasks, unlike existing approaches in the literature.

As an example, consider the system $T_1 = Task_{1,3} \| Task_{1,5} \| PRM_{3,5}$. Figure 1 shows the initial state space using the same notation as above, except now the state tuple also includes the state of the supply. Note that, in this transition system we have actions pertaining to resource consumption, abbreviated by $\overleftrightarrow{r}$, actions pertaining to resource requests, abbreviated by $\overline{r}$, and idling actions. Recall that idling and consumed resource actions are incomparable in the preemption relation, while idling preempts unsatisfied resource requests. We see that a poor scheduling decision can make $Task_{1,3}$ miss its deadline. The scenario is seen on the right side of the figure: in the first two time units, one unit of resource goes to $T_{1,5}$ and the other unit of resource is denied to both tasks (this can happen in any order). If on the third step the supply denies access to the resource again, the first task cannot idle, thus we reach a transition labeled by $\{r\}$, which implies that the task misses its deadline, leading to a violation of Definition 1.

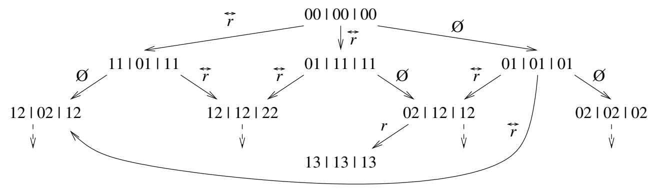

Fig. 1. Scheduling with a periodic resource

5 Conclusions

In this paper, we have presented PADS, a process algebra for resource demand and supply. The algebra can be used to describe a process and its demand on resources necessary for the execution of a real-time task as well as a supply process that describes the behavior of a resource allocator. We have defined precisely the notion of schedulability using demand and supply, that is, when a process can be scheduled under a supply process, and provided a compositional theory of demand-supply schedulability. We believe that PADS is the first process algebra that can describe the behavior of demand and supply processes and compositional schedulability between them.

There are several directions in which the current work can be extended. We are currently adding priorities to resource requests in the same way as in [13]. This allows us to represent schedulability with respect to particular schedulers, which is often a more practical question to analyze. We plan to extend the framework with the notion of order between supplies. This notion will capture the "generosity" of a supply, that is, a more generous supply will be able to schedule any task that the less generous supply can. With this notion, we will be able to formally represent the hierarchical scheduling approaches based on resource models [21] that rely on approximating the necessary supply, making it more generous than necessary, in exchange for a simple representation. It would also be interesting to explore how to extend the notion of schedulability to the notion of *resource satisfiability* between demand and supply of arbitrary resources that are not shared mutually exclusively. Another extension is to explore demand and supply processes in the presence of probabilistic behavior.

References

1. K. Altisen, G. Gößler, A. Pnueli, J. Sifakis, and Y. Yovine. A framework for scheduler synthesis. In *Proceedings of RTSS'99*, pages 154–163. IEEE Computer Society, 1999.
2. K. Altisen, G. Gößler, and J. Sifakis. Scheduler modeling based on the controller synthesis paradigm. *Real-Time Systems*, 23(1-2):55–84, 2002.
3. H. Ben-Abdallah, J.-Y. Choi, D. Clarke, Y. S. Kim, I. Lee, and H.-L. Xie. A Process Algebraic Approach to the Schedulability Analysis of Real-Time Systems. *Real-Time Systems*, 15:189–219, 1998.
4. G. Bucci, A. Fedeli, L. Sassoli, and E. Vicario. Modeling flexible real time systems with preemptive time Petri nets. In *Proceedings of ECRTS'03*, pages 279–286. IEEE Computer Society, 2003.
5. Z. Deng and J. W.-S. Liu. Scheduling real-time applications in an open environment. In *Proceedings of RTSS'97*, pages 308–319. IEEE Computer Society, 1997.
6. A. Easwaran, M. Anand, and I. Lee. Compositional analysis framework using edp resource models. In *Proceedings of RTSS'07*, pages 129–138. IEEE Computer Society, 2007.
7. A. Easwaran, I. Lee, and O. Sokolsky. Interface algebra for analysis of hierarchical real-time systems. In *Proceedings of FIT'08*, 2008.
8. X. Feng and A. Mok. A model of hierarchical real-time virtual resources. In *Proceedings of RTSS'02*, pages 26–35. IEEE Computer Society, 2002.
9. E. Fersman, P. Krcál, P. Pettersson, and W. Yi. Task automata: Schedulability, decidability and undecidability. *Information and Computation*, 205(8):1149–1172, 2007.
10. E. Fersman, P. Pettersson, and W. Yi. Timed automata with asynchronous processes: Schedulability and decidability. In *Proceedings of TACAS'02*, LNCS 2280, pages 67–82, 2002.
11. R.-T. S. GmbH. Real-Time Hybervisor, 2010. www.real-time-systems.com.
12. T. A. Henzinger and S. Matic. An interface algebra for real-time components. In *Proceedings of RTAS'06*, pages 253–263. IEEE Computer Society, 2006.
13. I. Lee, P. Brémond-Grégoire, and R. Gerber. A Process Algebraic Approach to the Specification and Analysis of Resource-Bound Real-Time Systems. *Proceedings of the IEEE*, pages 158–171, 1994.
14. I. Lee, A. Philippou, and O. Sokolsky. Resources in process algebra. *Journal of Logic and Algebraic Programming*, 72:98–122, 2007.
15. Linuxworks. LynxSecure Embedded Hypervisor and Separation Kernel, 2010. www.lynuxworks.com/virtualization/hypervisor.php.
16. G. Lipari and E. Bini. Resource partitioning among real-time applications. In *Proceedings of ECRTS'03*, pages 151–160. IEEE Computer Society, 2003.
17. J. Liu. Real-time systems. *Prentice Hall*, 2000.
18. M. Mousavi, M. Reniers, T. Basten, and M. Chaudron. PARS: a process algebra with resources and schedulers. In *Proceedings of FORMATS'03*, LNCS 2791, pages 134–150, 2003.
19. M. Nunez and I. Rodriguez. PAMR: A process algebra for the management of resources in concurrent systems. In *Proceedings of FORTE'01*, pages 169–184, 2001.
20. S. Saewong, R. Rajkumar, J. Lehoczky, and M. Klein. Analysis of hierarchical fixed-priority scheduling. In *Proceedings of ECRTS'02*, pages 173–181. IEEE Computer Society, 2002.
21. I. Shin and I. Lee. Periodic resource model for compositional real-time guarantees. In *Proceedings of RTSS'03*, pages 2–13. IEEE Computer Society, 2003.
22. I. Shin and I. Lee. Compositional real-time scheduling framework. In *Proceedings of RTSS'04*, pages 57–67. IEEE Computer Society, 2004.
23. L. Thiele, E. Wandeler, and N. Stoimenov. Real-time interfaces for composing real-time systems. In *Proceedings of EMSOFT '06*. ACM, 2006.