PADS: An approach to modeling resource demand and supply for the formal analysis of hierarchical scheduling

Anna Philippou\textsuperscript{a}, Insup Lee\textsuperscript{b}, Oleg Sokolsky\textsuperscript{b}

\textsuperscript{a}Department of Computer Science, University of Cyprus, Nicosia, Cyprus
\textsuperscript{b}Department of Computer and Info. Science, University of Pennsylvania, Philadelphia, PA, U.S.A.

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 $\text{CPU}^{\text{C}}$ denotes for availability of CPU for the corresponding demand process. Using PADS, we define a supply-demand relation where a pair $(T, S)$ 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. Furthermore, we define ordering relations between supplies which describe when a supply offers more resource capacity than another. With this notion it is possible to formally represent hierarchical scheduling approaches that assign more “generous” resource allocations to tasks in exchange for a simple representation. We illustrate our techniques 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 standards by the Engineering Standards for Avionics and Cabin Systems committee specify 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. 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, 21]. To facilitate the analysis of such systems, resource component interfaces and their compositional analysis have been proposed [16, 22, 23, 8, 24, 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 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 \( S \vdash T \) that captures when a task \( T \) is schedulable by a supply \( S \).

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:

   \[
   S_1 \vdash T_1, S_2 \vdash T_2 \quad \Rightarrow \quad S_1 \circ S_2 \vdash T_1 \parallel T_2, \quad f(S_1, S_2)
   \]

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.

4. **Task and Supply Orderings**: We propose an ordering between tasks which defines when a task is more “demanding” than another, meaning that it requires more resources in to execute correctly. We also propose two orderings between supplies which define when a supply is more “generous” than another meaning that it offers a greater resource allocation. The main result accompanying these notions
is that any supply that schedules a more demanding task may also schedule a less demanding task and that any task schedulable by a less generous supply is also schedulable by a more generous supply. This result comes to complement our supply synthesis approach since it allows us to check whether a supply $S$ schedules a task set as follows: We begin by constructing the optimal supply/demand, $D$, for the task set and then check whether $S$ is more generous than $D$. In the affirmative case we may conclude that the task set is also schedulable by $S$.

**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, 22, 23, 8, 6, 7]. Typically, such approaches to schedulability analysis rely on over-approximations of task demand using, for example, demand bound functions and under-approximations 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 modeling of 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 present paper extends our previous work of [20] as follows. It introduces priorities to the framework, thus allowing us to represent schedulability with respect to particular schedulers and it contains all the proofs missing from [20] adopted for the extended framework. Furthermore, it introduces ordering relations between tasks and supplies and associated results that enable us to formally represent techniques for over-approximating optimal resources as can be found in e.g. [22].

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 3.3. Section 4 presents hierarchies between tasks and supplies and develops their properties and, finally, 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 $r \in R$ can be requested by a task, $r$, granted by a supply, $\bar{r}$, or consumed, $\bar{r}$, when a supply and a request for the resource are simultaneously available.
2.2. Semantics

An action in PADS is a set relating to resource requests, grants and consumptions, where each resource may be represented at most once. Resource requests and consumptions are associated with a priority, where priorities are drawn from the nonnegative integers. These priorities are used to arbitrate between actions the intention being that an action with a higher priority always wins. Supplies of resources are not associated with priorities since a resource can either be supplied or not supplied to a component and cannot be simultaneously offered to two or more tasks in a system. For example, the action \( \{(r_1, 1), (r_2, 2)\} \) represents a request for the resources \( r_1 \) and \( r_2 \) at priorities 1 and 3, respectively, whereas the action \( \{\pi_1, (\overline{r}_2, 2), (r_3, 1)\} \) involves the granting of resource \( r_1 \), consumption of resource \( r_2 \) at priority level 2 and request for resource \( r_3 \) at priority level 1. 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 \( \text{Act} \), ranged over by \( \alpha \) and \( \beta \), for the set of all actions and distinguish \( \text{Act}_R \), the set of actions involving only resource requests, ranged over by \( \rho \), and \( \text{Act}_G \), the set of actions involving only resource grants, ranged over by \( \gamma \). Given \( \alpha \in \text{Act} \) we write \( \alpha^3 \) to remove all priorities from resource-priority pairs in \( \alpha \), e.g. \( \{(r_1, 2), \pi_2, (\overline{r}_3, 1)\}^3 = \{r_1, \pi_2, \overline{r}_3\} \) and \( \text{res}(\alpha) \) for the set of resources occurring in \( \alpha \), i.e. \( \text{res}(\alpha) = \{r \in R | (r, p) \in \alpha \text{ or } \pi \in \alpha \text{ or } (\overline{r}, p) \in \alpha, \text{ where } p \in \mathbb{N}\} \). Finally, given an action \( \alpha \) and a resource \( r \), we write \( \pi_{\alpha}(r) \) for the priority at which resource \( r \) is employed within action \( \alpha \) where we consider all supplied resources to be employed at priority level 0, e.g. for \( \alpha = \{(r_1, 2), \pi_2, (r_3, 4)\}, \) we write \( \pi_{\alpha}(r_1) = 2, \pi_{\alpha}(r_2) = 0 \) and \( \pi_{\alpha}(r_3) = 4 \).

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.

\[
T ::= \text{FIN} \mid \Sigma_{i \in I} \rho_i : T_i \mid C \\
S ::= \text{FIN} \mid \Sigma_{i \in I} \pi_i : S_i \mid D \\
P ::= \delta \mid T \mid S \mid P \parallel P \mid \Sigma_{j \in J} \alpha_j : P_j
\]

We consider \( \text{FIN} \) 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 \( \rho_i \) and then proceeding as \( T_i \). Similarly, a supply process can be a terminated process, a supply constant, or a nondeterministic choice. Finally, a process can be a deadlocked system, \( \delta \), 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, \( \rightarrow \rightarrow \). Then, we refine \( \rightarrow \rightarrow \)
Consider the supply \( \sim \) on the basis of a preemption relation which implements a type of “angelic” behavior in the way in which tasks resolve their nondeterminism, choosing the best possible outcome given the available supply, and by taking priorities into account.

We proceed to consider relation \( \rightarrow \) defined in Table 1. FIN being a well-terminated (and not a deadlocked) process, it allows time to pass (axiom (\text{IDLE})). Non-deterministic choice in tasks and supplies can be resolved by executing an action and then proceeding as its continuation ((\text{SumT}) and (\text{SumS})). A constant behaves as the process in its defining equation (\text{Const}). Finally, rule (\text{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 (\text{Par}) imposes a restriction on when two actions may take place simultaneously within a system. Specifically, we say that actions \( \alpha_1 \) and \( \alpha_2 \) 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:

\[
\text{compatible}(\alpha_1, \alpha_2) = \bigwedge_{r \in \text{res}(\alpha_1) \cap \text{res}(\alpha_2)} (r \in \alpha_1^\downarrow \land \overline{r} \in \alpha_2^\downarrow) \lor (r \in \alpha_2^\downarrow \land \overline{r} \in \alpha_1^\downarrow)
\]

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

\[
\alpha_1 \oplus \alpha_2 = \{(r, p) \in \alpha_1 \cup \alpha_2 | \overline{r} \not\in \alpha_1 \cup \alpha_2 \} \cup \{\overline{r} \in \alpha_1 \cup \alpha_2 | (r, p) \not\in \alpha_1 \cup \alpha_2 \}
\]

\[
\cup \{(\overline{r}, p) | (r, p) \in \alpha_1, \overline{r} \in \alpha_2(i+1)_{\text{mod}2}, i \in \{1, 2\} \text{ or } (\overline{r}, p) \in \alpha_1 \cup \alpha_2 \}
\]

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

**Example 2.1.** Consider the supply \( S \overset{\text{def}}{=} \{r_1, r_2\} : S \) and the following task processes:

\[
T_1 \overset{\text{def}}{=} \{(r_1, 1), (r_2, 1)\} : \text{FIN} + \{ (r_2, 2) \} : T_1 \quad T_3 \overset{\text{def}}{=} \{(r_2, 2)\} : \text{FIN}
\]

\[
T_2 \overset{\text{def}}{=} \{(r_2, 1)\} : \text{FIN} + \{ (r_2, 2), (r_3, 1) \} : T_2 \quad T_4 \overset{\text{def}}{=} \{(r_1, 1)\} : \text{FIN}
\]

We have:

\[
T_1 \parallel S \xrightarrow{(r_1, 1), (r_2, 1)} \text{FIN} \parallel S \quad (1) \quad T_1 \parallel S \xrightarrow{(r_2, 2)} T_1 \parallel S \quad (2)
\]

\[
T_2 \parallel S \xrightarrow{(r_3, r_2, 1)} \text{FIN} \parallel S \quad (3) \quad T_2 \parallel S \xrightarrow{(r_2, 2, 2), (r_3, 1)} T_2 \parallel S \quad (4)
\]

\[
(T_1 \parallel S) \parallel T_3 \xrightarrow{(r_1, 2), (r_2, 2)} (T_1 \parallel S) \parallel \text{FIN} \quad (5)
\]

Note that \((T_1 \parallel S) \parallel T_3\) has no transition other than (5) above, while \((T_1 \parallel S) \parallel T_4\) has no transitions altogether since both \(T_1\) and \(T_4\) require \(r_1\) during the first time unit. \( \square \)
Table 1: Transition rules for tasks, supplies and systems

<table>
<thead>
<tr>
<th>(Idle)</th>
<th>FIN $\xrightarrow{}$ FIN</th>
</tr>
</thead>
<tbody>
<tr>
<td>(SumT)</td>
<td>$\Sigma_{i \in I} : T_i \xrightarrow{\rho_i} T_i$ $I \neq \emptyset$</td>
</tr>
<tr>
<td>(SumS)</td>
<td>$\Sigma_{i \in I} : S_i \xrightarrow{\gamma_i} S_i$ $I \neq \emptyset$</td>
</tr>
<tr>
<td>(Const)</td>
<td>$P \xrightarrow{\alpha} P'$ $C = P$</td>
</tr>
<tr>
<td>(Par)</td>
<td>$P_1 \xrightarrow{\alpha_1} P'_1$, $P_2 \xrightarrow{\alpha_2} P'_2$, $P_1 \parallel P_2 \xrightarrow{\alpha_1 \oplus \alpha_2} P'_1 \parallel P'_2$ compatible($\alpha_1, \alpha_2$)</td>
</tr>
<tr>
<td>(SumP)</td>
<td>$\Sigma_{j \in J} : P_j \xrightarrow{\alpha_j} P_j$ $J \neq \emptyset$</td>
</tr>
</tbody>
</table>

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 enabled transitions, it should choose only between the ones that are satisfied by the available supply, assuming that such options exist. For example, $T_2 \parallel S$ 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 $T_1 \parallel S$ above should only retain transition (1) out of transitions (1) and (2).

Given the above, we define the preemption relation as follows:

Definition 2.2. We define the preemption relation $\prec \in \text{Act} \times \text{Act}$ so that $\alpha \prec \beta$ if one of the following hold:

1. $\{ r \mid r \in \alpha^+, \overline{r} \in \alpha^- \} = \{ r \mid r \in \beta^+, \overline{r} \in \beta^- \}$, $\alpha^+ \cap R \neq \emptyset$ and $\beta^+ \cap R = \emptyset$,
2. $\alpha^+ \cap R = \beta^+ \cap R = \emptyset$, $\text{res}(\alpha) = \text{res}(\beta)$ and $\{ r \mid r \in \alpha^+ \} \subset \{ r \mid \overline{r} \in \beta^- \}$.
3. $\text{res}(\alpha) = \text{res}(\beta)$ and for all $r \in \text{res}(\alpha)$, $\pi_\alpha(r) \leq \pi_\beta(r)$ and there exists $r \in \text{res}(\alpha)$, $\pi_\alpha(r) < \pi_\beta(r)$.

Intuitively, an action precludes another if it makes better usage of the same offered resources. In particular, an action $\beta$ preempts an action $\alpha$, if, either $\alpha$ and $\beta$ concern the same offered resources (granted or consumed) but $\alpha$, unlike $\beta$, also contains some unsatisfied resource requests (condition (1)), or $\alpha$ and $\beta$ contain only the same granted and consumed resources but $\beta$ consumes more resources than $\alpha$ (condition (2)). Finally, an action precludes another if it uses the same resources but at a higher or equal priority with at least one resource being used at a higher priority (condition (3)).

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

Intuitively, an action precludes another if it makes better usage of the same offered resources. In particular, an action $\beta$ preempts an action $\alpha$, if, either $\alpha$ and $\beta$ concern the same offered resources (granted or consumed) but $\alpha$, unlike $\beta$, also contains some unsatisfied resource requests (condition (1)), or $\alpha$ and $\beta$ contain only the same granted and consumed resources but $\beta$ consumes more resources than $\alpha$ (condition (2)). Finally, an action precludes another if it uses the same resources but at a higher or equal priority with at least one resource being used at a higher priority (condition (3)).

We may now define the relation $\xrightarrow{\alpha}$ by the following rule:
Consider \( T \) a task \( T = \{ r_1, \ldots, r_\alpha \} \).

We conclude this section by introducing some notations. We write \( P \rightarrow \) if there exists \( \alpha \) such that \( P \xrightarrow{\alpha} \). If \( P \xrightarrow{\alpha} \) for all actions \( \alpha \), we write \( P = \delta \), where \( \delta \) is the deadlocked process. We write \( P \Rightarrow P' \) if there exist \( \alpha_1, \ldots, \alpha_n \) and \( P_1, \ldots, P_{n-1} \) such that \( P \xrightarrow{\alpha_1} P_1 \xrightarrow{\alpha_2} \ldots \xrightarrow{\alpha_{n-1}} P_{n-1} \xrightarrow{\alpha_n} P' \). The set of traces of \( P \), traces(\( P \)), is defined to be the set of all infinite sequences \( \alpha_1 \alpha_2 \ldots \) such that \( P \xrightarrow{\alpha_1} P_1 \xrightarrow{\alpha_2} \ldots \). Furthermore, we write \( \kappa \) for elements of \( 2^R \) and \( \pi \) to transform all resource requests in \( \kappa \) into resource grants, so, \( \{ r_1, r_2 \} = \{ \tau_1, \tau_2 \} \). Extending this notation to traces, given \( w = \kappa_1 \kappa_2 \ldots \), we write \( \overline{w} \) for \( \overline{\kappa_1} \overline{\kappa_2} \ldots \). Finally, given \( \alpha \in \text{Act}_G \), we write \( \alpha^\gamma \) to transform all resource supplies into resource requests, so, \( \{ \tau_1, \tau_2 \}^\gamma = \{ r_1, r_2 \} \).

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 \( T^\times \) for the set containing all processes of the form \( \tau_1 \ldots \tau_n, n \geq 1 \), and \( \text{S}^\times \) for the set containing all processes of the form \( S_1 \ldots S_n, n \geq 1 \). For simplicity, we refer to elements of \( T^\times \) and \( \text{S}^\times \) simply as tasks and supplies, respectively.

**Definition 3.1.** A task \( T \in T^\times \) is schedulable under supply \( S \in \text{S}^\times \) if whenever \( T \parallel S \Rightarrow P \) then \( P \neq \delta \) and for all \( P \xrightarrow{\alpha} \) we have \( \alpha^\gamma \cap R = \emptyset \).

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 3.2.** Consider \( T_1 \overset{\text{def}}{=} \{(r_1, 1)\}:FIN \), \( T_2 \overset{\text{def}}{=} \{(r_1, 1)\}:FIN + \emptyset \), \( S_1 \overset{\text{def}}{=} \{ \tau_1 \}:FIN \), \( S_2 \overset{\text{def}}{=} \{ \tau_2 \}:FIN \) and \( S_3 \overset{\text{def}}{=} \{ \tau_1, \tau_2 \}:FIN \). We have:

- \( T_1 \) is not schedulable under \( S_1 \) since \( T_1 \parallel S_1 \xrightarrow{(r_1,1),\tau_2} \).
- \( T_2 \) is schedulable under \( S_1 \). We have \( T_2 \parallel S_1 \xrightarrow{(\tau_1)}\text{FIN}\parallel\text{FIN} \) as required.
- \( T_2 \) is schedulable under \( S_2 \). Their composition has only one possible transition \( T_2 \parallel S_2 \xrightarrow{\tau_1} \text{FIN}\parallel\text{FIN} \). Note that the transition \( T_2 \parallel S_2 \xrightarrow{\tau_1} \text{FIN} \) at the lower-level of the semantics is pruned by the preemption relation. Thus, the definition is satisfied. The same holds for \( T_3 \overset{\text{def}}{=} \{(r_1,1)\}:FIN + \{(r_2,1)\}:FIN \) and \( S_2 \) since \( \{(r_2,1),\tau_1\} \prec \{(r_1,1)\} \). 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. \( \square \)
Before moving on to our alternative schedulability definition we introduce the following useful notations: For $\beta \in \text{Act}_R$ and $\alpha \in \text{Act}_G$, we write $\text{sat}(\beta, \alpha)$ if $\text{res}(\beta) \subseteq \text{res}(\alpha)$, that is, all resource requests of action $\beta$ are satisfied by the grants of action $\alpha$. For $T \in T^*$ and $\beta \in \text{Act}_R$ and $\alpha \in \text{Act}_G$, we write, $\beta \preceq_T \alpha$, if $\models (\beta, \alpha)$ and there exists no $\gamma \in \text{Act}_R$ such that $T \xrightarrow{\gamma} T'$, $\beta' \subset \gamma$ and $\text{sat}(\gamma, \alpha)$.

**Definition 3.3.** A relation $S \subseteq T^* \times S^*$ is a supply simulation relation if for all $(T, S) \in S$, $S \longrightarrow$, and, if $S \xrightarrow{\alpha} S'$ then

1. there exists $T \xrightarrow{\beta} T'$ with $\text{sat}(\beta, \alpha)$ and $(T', S') \in S$, and
2. if $T \xrightarrow{\beta} T'$ with $\beta \preceq_T \alpha$, then $(T', S') \in S$.

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

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 $\beta$ by which the task can employ the offered resources $\alpha$, that is, where $\beta \preceq_T \alpha$. Note that any non-maximal transition of $T$ taking place as a response to $S \xrightarrow{\alpha}$ would be subsequently pruned in the composition $S \parallel T$ as it would be preempted by greedier responses of $T$. Therefore, such transitions can be ignored.

We may now prove that the two alternative schedulability notions coincide.

**Lemma 3.4.** A task $T \in T^*$ is schedulable under supply $S \in S^*$ if and only if $S \models T$.

**Proof:** To begin with, suppose there exists a supply simulation relation $R$ between $T$ and $S$. We will show that if $S \parallel T \xrightarrow{\alpha} S' \parallel T'$ then $S' \parallel T' \neq \delta$, $\alpha' \cap R = \emptyset$ and $(S', T') \in R$. So suppose that $S \parallel T \xrightarrow{\alpha} S' \parallel T'$, $S \xrightarrow{\alpha} S'$ and $T \xrightarrow{\alpha} T'$, $\alpha = \alpha_1 + \alpha_2$.

We know that for some $\beta$, $\text{sat}(\beta, \alpha_1)$, $T \xrightarrow{\beta} T''$ (Definition 3.3(1)). This implies that $\text{sat}(\alpha_2, \alpha_1)$ (otherwise $\alpha_1 + \alpha_2 \prec \alpha_1 + \beta$ and $S \parallel T \xrightarrow{\alpha} S'$). Consequently, we deduce that $\alpha' \cap R = \emptyset$. In addition, since $(T', S') \in R$, by Definition 3.3 we have that $S' \longrightarrow$ for each $S' \xrightarrow{\beta} S''$ there exists $T' \xrightarrow{\beta} T''$ such that $S' \parallel T'' \longrightarrow$, that is, $S'' \parallel T'' \neq \delta$.

And, finally, we may observe that there is no $T \xrightarrow{\gamma}$, $\alpha_2' \subset \gamma$, $\text{sat}(\gamma, \alpha_1)$ (otherwise $\alpha_1 + \alpha_2 \prec \gamma + \alpha_2$), which, by Definition 3.3(2), implies that $(S', T') \in S$.

Conversely, suppose that task $T$ is schedulable by supply $S$. We will show that

$$R = \{(S, T) | S \text{ is schedulable by } T\}$$

is a supply simulation relation. Suppose $(S, T) \in R$. Since $S \parallel T \neq \delta$, $S \longrightarrow$ and $T \longrightarrow$. Furthermore, if $S \xrightarrow{\alpha} S'$ then, since $T$ is schedulable by $S$, there exists $T \xrightarrow{\beta} T'$, $\text{sat}(\beta, \alpha)$. If not, that is for all $T \xrightarrow{\beta} T''$, $\text{res}(\beta) - \text{res}(\alpha) \neq \emptyset$, then $S \parallel T \xrightarrow{\gamma}$, $\gamma' \cap R \neq \emptyset$ which contradicts our assumption of $T$ being schedulable by $S$.  

8
Next, suppose that $T \xrightarrow{\beta} T'$, sat($\beta, \alpha$) and $\beta \leq_T \alpha$. 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 3.5.** A task $T \in T^*$ 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 and in doing so in all its enabled nondeterministic executions that can take place as a response to the supply 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 3.6.** Given a task $T \in T^*$ and an infinite trace $w = \kappa_1 \kappa_2 \ldots$, we define the $w$-cylinder of $T$ to be the set $A = \cup_{i \geq 1} A_i$, where

$$A_1 = \{ (T, \alpha_1, P_1) \mid T \xrightarrow{\alpha_1} P_1, \alpha_1^1 = \kappa_1 \}$$

$$A_i = \{ (P_i, \beta_i, P'_i) \mid P_i \xrightarrow{\beta_i} P'_i, \beta_i \leq P_i, \exists (Q, \gamma, P_i) \in A_{i-1} \}, \quad i > 1$$

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 $(Q, \alpha, \delta)$, (ii) $A_i \neq \emptyset$ for all $i$ and (iii) $\bigcup_{(P, \beta, Q) \in A_i} \beta^p = \kappa_i$.

**Lemma 3.7.** A task $T \in T^*$ is schedulable if and only if it possesses a live cylinder.

**Proof:** Suppose $T$ has a live $w$-cylinder where $w = \kappa_1 \kappa_2 \ldots$. Consider supply $S_0$ defined by the following set of equations $S_i \overset{\text{def}}{=} \overline{\kappa_{i+1}} S_{i+1}$. Then, we may confirm that $S_0 \models T$. In particular we show that if $A = \cup_{i \geq 1} A_i$ is the $w$-cylinder of $T$, then

$$\mathcal{R} = \{ (T, S_i) \mid (T, \beta, Q) \in A_i, i \geq 1 \}$$

is a supply relation. So, consider $(T, S_i) \in \mathcal{R}$. To begin with, trivially $S_i \longrightarrow$. Further, if $S_i \xrightarrow{\alpha} S_{i+1}$, then since $A_i \neq \emptyset$, there exists $T \xrightarrow{\beta} Q$, $\beta \leq_T \alpha$, and $(Q, S_{i+1}) \in \mathcal{R}$. In fact, this holds for all $T \xrightarrow{\beta} Q$, where $\beta \leq_T \alpha$ and the result follows.

On the other hand, if $T$ is schedulable, then there exists a supply $S$ that schedules it. Let $w = \overline{p_1} \overline{p_2} \ldots \in \text{traces}(S)$. We may construct a cylinder $A = \cup_{i \geq 1} A_i$ of $T$ as

$$A_1 = \{ (T, \alpha_1, P'_1) \mid P \xrightarrow{\alpha_1} P', \alpha_1^1 = \kappa_1 \}$$

$$A_i = \{ (P_i, \beta_i, P'_i) \mid P_i \xrightarrow{\beta_i} P'_i, \beta_i \leq p, \exists (Q, \gamma, P_i) \in A_{i-1} \}, \quad i > 1$$

Since $T$ is schedulable by $S$ it is straightforward to see that $A$ contains no triple of the form $(Q, \alpha, \delta)$ and also that $A_i \neq \emptyset$ for all $i$. Finally, if we take $\beta_i = \bigcup_{(P, \beta, Q) \in A_i} \beta_p$, we may conclude that $A = \cup_{i \geq 1} A_i$ is a $w'$-cylinder of $T$, where $w' = \beta_1 \beta_2 \ldots$. □

### 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 greater 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 3.8.** Given supplies $S_1$ and $S_2$ we define $S_1 \otimes S_2$ as

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

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:

$\emptyset : \{\text{cpu}\} : \emptyset : \text{FIN} \otimes \emptyset : \emptyset : \{\text{cpu}\} : \text{FIN} = \emptyset : \{\text{cpu}\} : \{\text{cpu}\} : \text{FIN}$

$\emptyset : \{\text{cpu}\} : \emptyset : \text{FIN} \otimes (\emptyset : \emptyset : \{\text{cpu}\} : \text{FIN} + \{\text{cpu}\} : \emptyset : \text{FIN}) = \emptyset : \{\text{cpu}\} : \{\text{cpu}\} : \text{FIN} + \{\text{cpu}\} : \{\text{cpu}\} : \emptyset : \text{FIN}$

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 3.9.** Given a task $T \in T^*$, we define its demand as follows:

$$\text{demand}(T) \stackrel{\text{def}}{=} \sum_{\alpha : T \rightarrow \alpha'} \overline{\alpha'} : [\otimes_{T', T \rightarrow \alpha', T'} \text{demand}(T')]$$

**Example 3.10.** Consider tasks

$T_1 = \{(\text{cpu}, 2)\} : \emptyset : \emptyset : T_1 + \emptyset : \{(\text{cpu}, 1)\} : \emptyset : T_1 + \emptyset : \{(\text{cpu}, 3)\} : T_1$

$T_2 = \{(\text{cpu}, 1)\} : \emptyset : \emptyset : T_2 + \emptyset : \{(\text{cpu}, 2)\} : \emptyset : T_2 + \emptyset : \{(\text{cpu}, 2)\} : T_2$

$T_3 = \{(\text{cpu}, 1)\} : \{(\text{cpu}, 1)\} : \text{FIN} + \{(\text{cpu}, 2)\} : \emptyset : T_3$

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

$X_1 = \{\text{cpu}\} : \emptyset : \emptyset : X_1 + \emptyset : \{\text{cpu}\} : \{\text{cpu}\} : X_1$

$X_2 = \{\text{cpu}\} : \emptyset : \emptyset : X_2 + \emptyset : \{\text{cpu}\} : \emptyset : X_2 + \emptyset : \{\text{cpu}\} : X_2$

$X_3 = \{\text{cpu}\} : \emptyset : X_3$

□
The next lemma considers the optimality of \( \text{demand}(T) \) 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 \( \alpha_j \subseteq \beta_j \) for all \( i \geq 1 \).

**Lemma 3.11.** If \( w \in \text{traces(\text{demand}(T))} \) then \( T \) possesses a live \( w \)-cylinder and if \( w \in \text{traces}(\text{demand}(T)) \) then there exists \( w' \in \text{traces}(\text{demand}(T)) \) such that \( w \leq w' \).

**Proof:** Suppose \( \text{demand}(T) \xrightarrow{\alpha} S_1 \xrightarrow{\alpha_2} S_2 \xrightarrow{\alpha_3} \ldots \). We will show that for the \( w \)-cylinder \( A = \bigcup_{i \geq 1} A_i \) of \( T \), where \( w = \alpha_1^{\alpha} \alpha_2^{\alpha} \ldots \), we have \( S_i = \otimes_{(P, \beta, Q) \in A,} \text{demand}(Q) \) and \( \alpha \) is live. Consider an arbitrary \( S_i \) and suppose \( S_i = \otimes_{(P, \beta, Q) \in A_i} \text{demand}(Q) \) where \( A_i \neq \emptyset \) and \( A_i \) does not contain elements of the form \( (P, \beta, \delta) \). Then, since \( S_i \overset{\alpha_{i+1}}{\longrightarrow}, \) by the definition of \( \otimes \), it must be that \( \alpha_{i+1} = \bigcup \{ \alpha \ | \ (P, \beta, Q) \in A_i, \text{demand}(Q) \xrightarrow{\alpha} \} \). In addition, \( S_i \overset{\alpha_{i+1}}{\longrightarrow} \otimes_{T' \in B} \text{demand}(T') \), \( B = \{ T \ | \ (P, \beta, Q) \in A_i, Q \xrightarrow{\beta} T', \beta \subseteq Q \alpha_{i+1} \} \). But, \( B = A_{i+1} \) and by the construction of \( \alpha_{i+1} \), \( A_{i+1} \neq \emptyset \) and \( A \) is live, which completes the first part of the proof.

To establish the second part of the proof if is sufficient to note that if \( T \xrightarrow{\alpha} T' \) then \( \text{demand}(T) \xrightarrow{\alpha} \text{demand}(T') \otimes S \) for some \( S \) and, further, if \( S_1 \xrightarrow{\alpha} S'_1 \) then \( S_1 \otimes S_2 \xrightarrow{\alpha'} S'_1 \times S'_2 \), where \( \alpha \subseteq \alpha' \) for some \( S'_2 \). Then, by the definition of demand, it is easy to see that if \( T \xrightarrow{\alpha_1} T_1 \xrightarrow{\alpha_2} T_2 \xrightarrow{\alpha_3} \ldots \), then \( \text{demand}(T) \xrightarrow{\beta} \text{demand}(T_1) \otimes S_1 \xrightarrow{\beta_1} \text{demand}(T_2) \otimes S_2 \xrightarrow{\beta_2} \ldots \), where \( \alpha_1^{\alpha} \alpha_2^{\beta} \ldots \beta_1^{\beta} \beta_2^{\beta} \ldots \).

Thus, we may conclude that a task \( T \) is schedulable by \( \text{demand}(T) \). Furthermore, \( \text{demand}(T) \) is an optimal supply for \( T \) since each of its executions schedules exactly a cylinder of \( T \), i.e. it offers exactly the resources necessary for scheduling the cylinder, and each possible schedule of \( T \) is captured by an execution of \( \text{demand}(T) \).

### 3.2. Compositional Theory

We proceed to consider the schedulability problem of a set of task components. The first issue we tackle is the compositionality problem: If a component \( T_1 \) is schedulable by \( S_1 \) and an independent component \( T_2 \) by \( S_2 \) can we combine \( S_1 \) and \( S_2 \) into a collective supply that schedules \( T_1 || T_2 \)? We begin by noting a subtlety pertaining to this problem which we need to consider before answering it.

Consider the two components below consisting each of one task:

\[
\begin{align*}
T_1 &= \{(r, 1)\}: \emptyset; \text{FIN} + \emptyset; \{(r, 1)\}; \text{FIN} \\
T_2 &= \{(r, 1)\}: \emptyset; \text{FIN} + \emptyset; \{(r, 1)\}; \{(r, 1)\}; \text{FIN}
\end{align*}
\]

These components are schedulable under supplies \( S_1 = \emptyset; \{\tau\}; \text{FIN} \) and \( S_2 = \{\tau\}; \emptyset; \text{FIN} \), respectively. That is, it is sufficient for component \( T_1 \) to obtain resource \( r \) during the second time unit and for component \( T_2 \) during the first time unit. However, a supply \( S = \{\tau\}; \{\tau\}; \text{FIN} \), 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 component \( T_2 \) but may be consumed by component \( T_1 \) leading to a deadlock of the system during the third time unit. Moreover, if \( T_1 \) employed its resources at priority level 2, this would in fact be destined to happen.
Given supplies

Let $R = \{(r[i], 1)\}$:FIN be the resource and for all other purposes should be treated as the same. So, for example, the component with identifier $i$ which the resource is employed/supplied. Precisely, we assume that each component is annotating each resource reference by a number which distinguishes the component in $S$. However, note that even if $R$ do refer to the same resource and for all other purposes should be treated as the same. So, for example, $(r[1], r[2]) \neq 0$. 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 $\{r \in R \mid r[i] \in res(\alpha) \text{ and } r[j] \in res(\beta)\}$.

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

**Lemma 3.12.** 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 $R$, below, is a supply simulation relation.

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

Let $(T_1[1]|T_2[2], S_1[1]|S_2[2]) \in R$. By the definition of $R$, $S_1[1]|S_2[2] \rightsquigarrow S_1[1]|S_2[2]$. It must be that $\alpha = \alpha_1[1] \oplus \alpha_2[2]$, where $S_1 \overrightarrow{\alpha} S_1', S_2 \overrightarrow{\alpha_2} S_2'$ and $\alpha \cap_R \alpha_2 = \emptyset$. Since $S_1 \models T_1$, $S_2 \models T_2$, we have $T_1 \beta_1 \Rightarrow_1 T_1'$, $S_1' \models T_1'$, and similarly $T_2 \beta_2 \Rightarrow_2 T_2', S_2' \models T_2'$. In fact, for all $T_1 \beta_1 \Rightarrow_1 T_1'$, $\beta_1 \preceq_T \alpha_1$, it holds that $S_1' \models T_1'$, and for all $T_2 \beta_2 \Rightarrow_2 T_2'$, $\beta_2 \preceq_T \alpha_2$, it holds that $S_2' \models T_2'$. This implies that for all $T_1[1]|T_2[2] \beta \Rightarrow_T T_1'[1]|T_2'[2]$, $\beta \preceq_T T_1[1]|T_2[2] \alpha$, $(T_1'[1]|T_2'[2], S_1'[1]|S_2'[2]) \in R$ and there exists at least one such $\alpha$-transition. This completes the proof.

However, note that even if $S_1 || S_2$ deadlocks, it is still possible that the supplies $S_1$ and $S_2$ can be combined to produce a supply 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.

**Definition 3.13.** Given supplies $S_1$ and $S_2$ we define their product $S_1 \times S_2$ by

$$S_1 \times S_2 = \begin{cases} S_1 & \text{if } S_2 = \text{FIN} \\ S_2 & \text{if } S_1 = \text{FIN} \\ (\alpha \cup \beta):(S_1' \times S_2') & \text{if } S_1 = \alpha:S_1', S_2 = \beta:S_2', \alpha \cap_R \beta = \emptyset, S_1' \times S_2' \neq \emptyset \\ \delta & \text{if } S_1 = \alpha:S_1', S_2 = \beta:S_2', \alpha \cap_R \beta = \emptyset \text{ or } S_1' \times S_2' = \emptyset \\ \Sigma_{i \in I, j \in J}(S_1' \times S_2') & \text{if } S_1 = \Sigma_{i \in I} S_1', S_2 = \Sigma_{j \in J} S_2' \end{cases}$$
Given supplies $S_1$ and $S_2$ the set of equations which arise through $S_1 \times S_2$ has a greatest fixed point.

**Lemma 3.14.** Given supplies $S_1$ and $S_2$ the set of equations which arise through $S_1 \times S_2$ has a greatest fixed point.

**Proof:** Consider the term $S_1 \times S_2$. Let $S_{S_1,S_2} = \{S \mid S_1 \Rightarrow S \text{ or } S_2 \Rightarrow S\}$ and $S_{S_1,S_2}^X = S_{S_1,S_2} \cup [S_{S_1,S_2} \times S_{S_1,S_2}]$. For finite-state processes, $S_{S_1,S_2}^X$ is finite.

Consider the set of relations on $S_{S_1,S_2}^X$: $\mathcal{W} = \{W \mid W \subseteq S_{S_1,S_2}^X \times S_{S_1,S_2}^X\}$, ordered by set inclusion. Because every element of $\mathcal{W}$ is a subset of a finite set, $\mathcal{W}$ is a complete lattice of finite height.

Consider relation $W$ where, if $(w_1, w_2) \in W$, then $w_1 \in [S_{S_1,S_2} \times S_{S_1,S_2}]$ and $w_2 \in S_{S_1,S_2}^X$ appears on the right-hand side of the equation for $w_1$, disregarding the recursive part of the third clause and the fourth clause, so that

- if $S_1 = \text{FIN}$ then $(S_1 \times S_2, S_{(i+1) \mod 2}) \in W$,
- if $S_1 = \alpha; S_1', S_2 = \beta; S_2'$ and $(S_1 \times S_2, S_1' \times S_2') \in W$, then $\alpha \cap \beta \neq \emptyset$,
- if $S_1 = \Sigma_{i \in I} S_1^i, S_2 = \Sigma_{j \in J} S_2^j$, then $(S_1 \times S_2, S_1^i \times S_2^j) \in W$ then $S_1^i = S_2^j$ for some $i \in I$ and $S_2^j = S_1^i$ for some $j \in J$.

Thus, such a relation $W$ relates a product $S_1 \times S_2$ with some of its possible derivatives according to the selected part of the definition. Further, such $W$ represent solutions to the set of equations defining $S_1 \times S_2$ if whenever $w \in [S_{S_1,S_2} \times S_{S_1,S_2}]$ and there exists $w_1$ such that $(w_1, w) \in W$, then there also exists $w_2$ such that $(w, w_2) \in W$. This is because, according to the complete definition, $S_1 \times S_2$ has some derivative $w$, if and only if $w$ has a derivative (i.e., $w \neq \delta$).

Define a function $\mathcal{F} : \mathcal{W} \rightarrow \mathcal{W}$ as $\mathcal{F}(W) = W - \{(w_1, w_2) \mid s_2 \in [S_{S_1,S_2} \times S_{S_1,S_2}] \land \forall w, (w_2, w) \notin W\}$. Since $\mathcal{F}$ can only remove elements from $W$, $\mathcal{F}(W) \leq W$. Furthermore, if $W_1 \leq W_2$, then $\mathcal{F}(W_1) \leq \mathcal{F}(W_2)$; that is, $\mathcal{F}$ is monotonic. Let us construct the set $W_0$ using the non-recursive version of the $S_1 \times S_2$ definition; that is, by omitting $S_1^i \times S_2^j \neq \delta$ from clause 3 and $S_1^i \times S_2^j = \delta$ and clause 4. Clearly, a solution to $S_1 \times S_2$, $W_{S_1 \times S_2}$ satisfies $W_{S_1 \times S_2} \leq W_0$ since fewer terms are set to $\delta$ in $W_0$. Since $W$ is a finite complete lattice, by the constructive version of the Tarski-Knaster theorem, the greatest fixed point can be computed starting from $W_0$ and iteratively applying $\mathcal{F}$ until the fixed point is reached.

It is easy to see that, if $S_1 \parallel S_2$ 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 $S_1 \parallel S_2$ that does not lead to deadlocked states. For example, consider $S_1 \overset{\text{def}}{=} \{\tau\}; \text{FIN} + \emptyset; \{\tau\}; \text{FIN}$ and $S_2 \overset{\text{def}}{=} \emptyset; \{\tau\}; \text{FIN} + \{\tau\}; \emptyset; \text{FIN}$. Then, although $S_1 \parallel S_2 \overset{\text{def}}{=} \{\tau\}; \text{FIN} \parallel \{\tau\}; \text{FIN} = \delta$, $S_1 \times S_2 = \{\tau\}; \{\tau\}; \text{FIN} \parallel \emptyset; \text{FIN}$, and $(\{\tau\}; \{\tau\}; \text{FIN} \parallel \emptyset; \text{FIN}) = \{\tau\}; \{\tau\}; \text{FIN}$. 

13
Lemma 3.15. 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 Lemma 3.12.

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 3.16. If $w \in \text{traces}(T_1[1]||T_2[2])$ then there exists a trace $w' \in \text{traces}(\text{demand}(T_1[1]) \times \text{demand}(T_2[2]))$ such that $w \leq w'$.

PROOF: Suppose that the $w \in \text{traces}(T_1[1]||T_2[2])$. It is easy to see that $w[1]$ and $w[2]$ give rise to traces of $T_1[1]$ and $T_2[2]$. Then, by Lemma 3.11, there exist $w_1$ and $w_2$ such that $w[1] \leq w_1$ and $w[2] \leq w_2$ such that $w_1 \in \text{traces}(\text{demand}(T_1[1]))$ and $w_2 \in \text{traces}(\text{demand}(T_2[2]))$. This implies that $w_1 \cup w_2$ is a trace of $\text{demand}(T_1[1]) \times \text{demand}(T_2[2])$, where $w \leq w_1 \cup w_2$, as required.

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

Theorem 3.17. $T_1||T_2$ is schedulable if and only if $\text{demand}(T_1[1]) \times \text{demand}(T_2[2]) \neq \delta$. Moreover, if it is schedulable, then it is schedulable by $\text{demand}(T_1[1]) \times \text{demand}(T_2[2])$.

PROOF: Suppose $T_1[1]||T_2[2]$ is schedulable. Then, by Lemma 3.7, it has a live $w$-cylinder. Let $w_1$ be a trace associated with the cylinder. Then, by Lemma 3.16, there is a trace $w_2$, $w_1 \leq w_2$ such that $w_2$ is a trace of $\text{demand}(T_1[1]) \times \text{demand}(T_2[2])$. This implies that $\text{demand}(T_1[1]) \times \text{demand}(T_2[2]) \neq \delta$. On the other hand, if $\text{demand}(T_1[1]) \times \text{demand}(T_2[2]) \neq \delta$, then, since, additionally, $\text{demand}(T_1[1])$ schedules $T_1[1]$ and $T_2[2]$ schedules $T_2[2]$, then, by Lemma 3.15, $T_1[1]||T_2[2]$ is schedulable by $\text{demand}(T_1[1]) \times \text{demand}(T_2[2])$.

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 $D_1 = \text{demand}(T_1) \times \text{demand}(T_2)$, $D_2 = D_2 \times \text{demand}(T_3), \ldots$. 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.

3.3. Examples

3.3.1. Scheduling periodic tasks

We first define a simple periodic task with period $p$ and execution time $w$, $\text{Task}_{w,p} = T_{0,0,w,p}$, as follows:

$$T_{e,t,w,p} = \begin{cases} \emptyset : T_{e,t+1,w,p} & \text{if } e = w, t < p \\ T_{0,0,w,p} & \text{if } e = w, t = p \\ \emptyset : T_{e,t+1,w,p} + \{(r, \pi)\} : T_{e+1,t+1,w,p} & \text{if } e < w, w - e < p - t \\ \{(r, \pi')\} : T_{e+1,t+1,w,p} & \text{if } e < w, w - e = p - t \end{cases}$$
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 3.1). Let us consider an instance of a classical scheduling problem for a set of periodic tasks running on a single processor resource: \(\text{Task}_{2,3}\|\text{Task}_{2,7}\|S\), where \(S = \{r\} : S\). 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 \(\{r, pr\}\) for the appropriate priorities \(pr\).

The tasks are schedulable according to the Definition 3.1 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 \(\text{Task}_{2,3}\) as it has the smallest period. Indeed, to consider schedulability under a specific scheduling policy, we would simply need to specify the priorities and check for the schedulability of the system within the new transition system.

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, \(\text{demand}(\text{Task}_{w,p}) = X_{0,0,w,p}\) is defined below:

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

It is easy to check that \(\text{demand}(\text{Task}_{2,3})\|=\text{demand}(\text{Task}_{2,7})\) does not deadlock and thus can schedule the two tasks according to Lemma 3.12.

Let us now consider a task with variable execution time which takes between \(b\) and \(w\) time units to complete: \(\text{Task}_{b,w,p} = \text{Task}_{b,p} + \text{Task}_{b+1,p} + \ldots + \text{Task}_{w,p}\). One can see that \(\text{demand}(\text{Task}_{b,w,p}) = \text{demand}(\text{Task}_{w,p})\). 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].

### 3.3.2. Scheduling with partial supplies

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 $P$:

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

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 [22, 23]. 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 $PRM_{w, P} = \text{demand}(Task_{w, P})$. 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 $T_1 = Task_{1,3} || Task_{1,5} || PRM_{3,5}$ where all priorities of resource requests are fixed to 1. 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 $\rightarrow r$, actions pertaining to resource requests, abbreviated by $\rightarrow$, 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 3.1.

If instead we wish to consider schedulability of the tasks under an EDF policy, we would have to repeat our analysis for periodic tasks with priorities defined as below.

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

Figure 1: Scheduling with a periodic resource
where $D_{max}$ is a number exceeding the largest period in the task set. In this new setting, the composition $T_2 = Task_{1,3} \parallel Task_{1,5} \parallel PRM_{3,5}$ is schedulable as shown in Figure 2. In the figure, preempted transitions are crossed out.

4. Hierarchies on tasks and supplies

In the previous section we defined an approach for scheduling a set of tasks via analysis of their demand processes which are supply processes capturing the precise resource allocation required by tasks to complete their execution. In this section we proceed to provide machinery that may allow us to reason about hierarchical approaches to scheduling that rely on approximating the necessary supply, making it more generous than necessary, in exchange to a simple representation. Specifically, we define an ordering relation between tasks and two ordering relations between supplies which describe when a task/supply requires/offers greater resource allocation than another.

4.1. Task demands

We proceed to consider the notion of task demand and we define a relation on tasks which characterizes when a task is more “demanding” than another in the sense that it places more requirements on the available supply.

**Definition 4.1.** A relation $\mathcal{D} \in \mathcal{T} \times \mathcal{T}$ is a demand relation if for all $(T_1, T_2) \in \mathcal{D}$, if $T_1 \xrightarrow{\alpha}$ then

1. there exist $T_2 \xrightarrow{\beta} T_2'$ with $\beta \preceq T_2 \alpha$, and $T_1 \xrightarrow{\alpha} T_1'$, such that $(T_1', T_2') \in \mathcal{D}$ and
2. if $T_2 \xrightarrow{\beta} T_2', \beta \preceq T_2 \alpha \cup \beta$ and $\alpha \preceq T_1 \alpha \cup \beta$, then there exists $T_1 \xrightarrow{\alpha} T_1'$ such that $(T_1', T_2') \in \mathcal{D}$.

We write $\preceq_\mathcal{D}$ for the largest demand relation and we say that a task $T_1$ is more demanding than a task $T_2$, $T_1 \preceq_\mathcal{D} T_2$, if there exists a demand relation $\mathcal{D}$ with $(T_1, T_2) \in \mathcal{D}$.

According to this definition, if $T_1$ is more demanding than $T_2$ then for every action $\alpha$ enabled by $T_1$, there is a move of $T_2$ which can be matched by some $\alpha$-move of $T_1$ and, furthermore, assuming that $\alpha$ is a maximal move of $T_1$ with respect to $\alpha \cup \beta$, all
moves $\beta$ of $T_2$ which are maximal with respect to $\alpha \cup \beta$ can be matched by some $\alpha$-move of $T_1$. To better understand the definition, let us first consider the point relating to the existence of an $\alpha$ move of $T_1$ (instead of universality): let

$$
T_1 \overset{\text{def}}{=} \{(r_1, 1)\} : \{(r_2, 0)\} : \text{FIN} + \{(r_1, 1)\} : \{(r_3, 0)\} : \text{FIN}
$$

$$
T_2 \overset{\text{def}}{=} \{(r_1, 1)\} : \{(r_2, 0)\} : \text{FIN}
$$

Although, $T_2$ cannot match the second summand of $T_1$, it is intuitive that $T_1$ should be considered as more demanding than $T_2$. This is because for $T_1$ to be scheduled successfully it is imperative that after being offered $r_1$ it will be offered simultaneously both $r_2$ and $r_3$. Thus, it is sufficient for $T_2$ to match one of the $\{(r_1, 1)\}$ actions of $T_1$.

Moving on to the second clause of the definition, we note that it aims to ensure that if a supply offers the combination of resources $\alpha \cup \beta$ where both $\alpha$ and $\beta$ are maximal matching actions of $T_1$ and $T_2$, respectively, then if the supply is able to schedule the $\alpha$-continuation of $T_1$ it should also be able to schedule the $\beta$ continuation of $T_2$, that is, $T_1$ should continue to be more demanding than $T_2$. For example, for

$$
T_1 \overset{\text{def}}{=} \{(r_1, 1), (r_2, 1)\} : \{(r_2, 0)\} : \text{FIN} + \{(r_1, 1), (r_3, 1)\} : \{(r_3, 0)\} : \text{FIN}
$$

$$
T_2 \overset{\text{def}}{=} \{(r_2, 1)\} : \{(r_2, 0)\} : \text{FIN} + \{(r_1, 1), (r_3, 1)\} : \{(r_3, 0)\} : \text{FIN}
$$

we may check that, according to the definition, $T_1$ is more demanding than $T_2$. Note that supply $S \overset{\text{def}}{=} \{\tau_1, \tau_2, \tau_3\} : \{\tau_3\} : \text{FIN}$, schedules both tasks and, furthermore, the $\{(r_2, 1)\}$ transition of $T_2$ need not concern us since it is not a maximal response of $T_2$ to $\{\tau_1, \tau_2, \tau_3\}$, and it will be preempted in the composition $T_2 \parallel S$.

Some further examples follow:

**Example 4.2.** Consider the following tasks.

$$
T_1 \overset{\text{def}}{=} \{(\text{cpu}, 1)\} : \emptyset : T_1
$$

$$
T_2 \overset{\text{def}}{=} \{(\text{cpu}, 1)\} : \emptyset : T_2 + \emptyset : \{(\text{cpu}, 1)\} : T_2
$$

$$
T_3 \overset{\text{def}}{=} \{(\text{cpu}, 1)\} : \emptyset : T_3 + \emptyset : \{(\text{cpu}, 1)\} : T_3
$$

$$
T_4 \overset{\text{def}}{=} \{(\text{cpu}, 1)\} : \emptyset : T_4 + \emptyset : \{(\text{cpu}, 1)\} : T_4
$$

$$
T_5 \overset{\text{def}}{=} \emptyset : \{(\text{cpu}, 1)\} : T_5
$$

$$
T_6 \overset{\text{def}}{=} \emptyset : \{(\text{cpu}, 1)\} : T_6 + \{(\text{cpu}, 1)\} : \{(\text{cpu}, 1)\} : T_6
$$

$T_1$ and $T_2$ request resource $\text{cpu}$ once in every two time units with the distinction that $T_1$ requires the resource during the first time unit whereas $T_2$ is satisfied with an allocation during either time units. We may verify that $T_1$ is more demanding than $T_2$. Note that action $T_2 \overset{\emptyset}{\rightarrow}$ need not be matched by $T_1$ since, according to the definition, it is not a maximal move of $T_2$ with respect to $\emptyset \cup \{(\text{cpu}, 1)\}$.

Moving on to tasks $T_3$ and $T_4$ we observe that they both require resource $\text{cpu}$ once in every three time units but they pose slightly different nondeterministic requirements. We may check that $T_1$ is more demanding than both tasks $T_3$ and $T_4$ which demand
cpu once every three time units. In addition, $T_2$ is more demanding than $T_4$ but not of $T_3$ since $T_3$ may choose to respond to an initial $\emptyset$ action with the third summand which is not less demanding than $T_2$ given that it requests resource cpu during the third time unit. A comparison between $T_3$ and $T_4$ shows that $T_3$ is more demanding than $T_4$. Finally, note that task $T_5$ is not more demanding than task $T_6$. Intuitively, we can see that task $T_5$ can be scheduled by supply $S = \{\text{cpu}\} : \emptyset : \{\text{cpu}\} : S$ but task $T_6$ cannot. Furthermore, according to the definition, action $T_0 \overset{(\{\text{cpu},1\})}{\rightarrow}$ needs to be examined as it is a maximal action of $T_0$ with respect to $\emptyset \cup \{\{\text{cpu},1\}\}$ and clearly one that illustrates the absence of a demand relation between the two tasks. This example brings out the subtle treatment required for the actions of the less demanding task.

We now proceed to justify our notion of more demanding. To begin with we may easily prove that $\preceq_D$ is reflexive and transitive. Furthermore, we may verify that more demanding tasks place more requirements on their supplies by proving that if task $T$ is more demanding than task $T'$ then a supply that can schedule $T$ can also schedule $T'$.

**Lemma 4.3.** Suppose that task $T_1$ is schedulable under supply $S$ and that $T_1$ is more demanding than $T_2$. Then, task $T_2$ is also schedulable by supply $S$.

**Proof:** The proof consists of showing that the relation

$$ S = \{(T_2, S) | \exists \text{ demand relation } D, \text{ supply relation } R \text{ and } \exists T \in T : (T_1, T_2) \in D, (T_1, S) \in R\} $$

is a supply simulation. Suppose $(T_2, S) \in S$ and $T_1$ is a task such that $(T_1, T_2) \in D$, where $D$ is a demand relation, and $(T_1, S) \in S$, where $S$ is a supply relation. Suppose $S \xrightarrow{\alpha} S'$. We confirm that the two clauses of Definition 3.3 are satisfied as follows:

- Since $(T_1, S) \in R$, there exists $T_1'$ with $T_1 \xrightarrow{\beta} T_1'$, sat($\beta$, $\alpha$), $\beta \preceq_{T_1} \alpha$ and $(T_1', S') \in R$. Then, by clause (1) of Definition 4.1, there exists $T_2'$, such that $T_2 \xrightarrow{\beta'} T_2'$ with $\beta' \preceq_{T_2} \beta$, and for some $T_1''$, $T_1 \xrightarrow{\beta} T_1''$, $(T_1'', T_2') \in D$. By Definition 3.3 it is also the case that $(T_1'', S') \in R$, while, clearly, sat($\beta'$, $\alpha$). This implies that $(T_2', S') \in S$ as required.

- Now suppose $T_2 \xrightarrow{\beta} T_2'$ with $\beta \preceq_{T_2} \alpha$. To begin with, we note that there exists $T_1 \xrightarrow{\gamma} T_1'$, sat($\gamma$, $\alpha$), $\gamma \preceq_{T_1} \alpha$ and $(T_1', T_2') \in D$. Now consider $\beta \cup \gamma$. It must be the case that both $\beta \preceq_{T_2} \beta \cup \gamma$ and $\gamma \preceq_{T_1} \beta \cup \gamma$, otherwise, we would have contradictions to our assumptions that $\beta \preceq_{T_2} \alpha$ and $\gamma \preceq_{T_1} \alpha$. Then, by Definition 4.1, there exists $T_1''$ such that $T_1 \xrightarrow{\gamma} T_1''$ and $(T_1'', T_2') \in D$. By Definition 3.3 it is also the case that $(T_1'', S') \in R$. This implies that $(T_2', S') \in S$ which completes the proof.

**4.2. Supply generosity**

Symmetrically to demands, we now proceed to define a hierarchy on supplies. This hierarchy is built on the basis of simulation relations that capture when a supply is more "generous" than another, where the intended meaning of “generosity” is that the more generous a supply the more tasks it can schedule. Below we define two such notions.
4.2.1. Strong generosity

**Definition 4.4.** A relation \( R \in S \times S \) is a strong generosity relation if for all \( (S_1, S_2) \in R \),

1. If \( S_2 \rightarrow \) then \( S_1 \rightarrow \),
2. If \( S_2 \rightarrow \) and \( S_1 \overset{\alpha}{\rightarrow} S'_1 \) then we have that \( S_2 \overset{\alpha}{\rightarrow} S'_2 \) and \( (S'_1, S'_2) \in R \).

We write \( \preceq_S \) for the largest strong generosity relation and we say that supply \( S_1 \) is strongly more generous than supply \( S_2 \), \( S_2 \preceq_S S_1 \), if there exists \( R \) with \( (S_1, S_2) \in R \).

According to the definition, \( S_1 \) is strongly more generous than \( S_2 \) if: (1) whenever \( S_2 \) is not deadlocked then \( S_1 \) is also not deadlocked, and (2) whenever \( S_2 \) is not deadlocked, then any action enabled by \( S_1 \) is also enabled by \( S_2 \). Intuitively, this definition aims to establish that any task scheduled by the less generous supply, \( S_2 \), can also be scheduled by more general supply, \( S_1 \). To implement this, \( S_1 \) is required to offer a subset of the behaviors of \( S_2 \), in this way it is guaranteed that each of \( S_1 \)'s executions is also possible in \( S_2 \) and, thus, any task schedulable by \( S_2 \) will be schedulable by \( S_1 \). Thus, in Example 3.10, \( X_1 \) is a strongly more generous supply than \( X_3 \).

It is interesting to note that the existence of more alternatives of granting resources in supplies diminishes their potential of scheduling tasks. For example, if

\[
T \overset{\text{def}}{=} \emptyset : \{(r, 1)\} : \text{FIN} + \{(r, 1)\} : \{(r, 2)\} : \{(r, 1)\} : \text{FIN}
\]

and

\[
S_1 \overset{\text{def}}{=} \emptyset : \{T\} : \text{FIN} + \{\overline{T}\} : \{T\} : \text{FIN}, \quad S_2 \overset{\text{def}}{=} \emptyset : \{\overline{T}\} : \text{FIN}
\]

although \( S_2 \) can schedule \( T \), this is not the case with \( S_1 \). The same is true in the case that we allow a supply to offer a wider range of resources. For example, \( S_1 \overset{\text{def}}{=} \{\overline{T}\} : \{T\} : \text{FIN} \) also fails to schedule task \( T \).

It is easy to show that \( \preceq_S \) is reflexive and transitive. Furthermore, the following result establishes that generosity preserves schedulability.

**Lemma 4.5.** If task \( T \) is schedulable under supply \( S_2 \) and \( S_1 \) is strongly more generous than supply \( S_2 \) then \( T \) is also schedulable under supply \( S_1 \).

**Proof:** The proof consists of showing that the relation

\[
S = \{(T, S_1)\} \exists S_2 \in S, \text{ supply relation } R \text{ and strong generosity relation } G : (S_1, S_2) \in G \text{ and } (T, S_2) \in R
\]

is a supply relation. Suppose \( (T, S_1) \in S \) and \( S_2 \) is a supply such that \( (T, S_2) \in R \), where \( R \) is a supply relation and \( (S_1, S_2) \in G \) where \( G \) is a strong generosity relation. Suppose \( S_1 \overset{\alpha}{\rightarrow} S'_1 \). By Definition 4.4(2), \( S_2 \overset{\alpha}{\rightarrow} S'_2 \) with \( (S'_1, S'_2) \in G \). Thus:

1. There exists \( T \overset{\beta}{\rightarrow} T', \beta \subseteq \alpha \) with \( (T', S'_2) \in R \). By definition, \( (T', S'_1) \in R \) as required.
2. Suppose \( T \overset{\beta}{\rightarrow} T', \beta \preceq_T \alpha \). Again we have \( (T', S'_2) \in R \) and \( (T', S'_1) \in R \) which completes the proof. \( \Box \)
In fact, we may also show that:

**Lemma 4.6.** $S_1$ is strongly more generous than $S_2$ if and only if each task schedulable under supply $S_2$ is also schedulable under supply $S_1$.

**Proof:** The ‘⇒’ direction follows by the previous lemma. To demonstrate the ‘⇐’ direction we will show that the following relation is a strong generosity relation.

$$R = \{ (S_1, S_2) \mid \forall T \cdot T \text{ schedulable by } S_2 \implies T \text{ schedulable by } S_1 \}$$

Suppose $(S_1, S_2) \in R$. We have the following:

1. Suppose $S_2 \xrightarrow{\alpha} S'_2$ and consider the set of tasks $\{ \alpha : T \mid T \text{ schedulable by } S'_2 \}$. Then, this set, being schedulable by $S_2$, is also schedulable by $S_1$, which implies that $S_1 \xrightarrow{\alpha}$ by Definition 3.3, as required.

2. Suppose $S_1 \xrightarrow{\alpha} S'_1$ and in order to reach a contradiction suppose further that $S_2 \not\xrightarrow{\alpha}$. Consider task $T \overset{\text{def}}{=} \sum_{S_2 \xrightarrow{\alpha_i}} \alpha_i : \text{FIN} + \alpha : T'$ where $T'$ is not schedulable by $S'_1$ not by any of $S_2$’s derivatives. Then $T$ is schedulable by $S_2$ but not $S_1$, resulting in a contradiction. This implies that $S_2 \xrightarrow{\alpha} S'_2$ and $(S'_1, S'_2) \in R$ as required.

As an example for strong generosity consider supplies $S_1$ and $S_2$ below

$$S_1 \overset{\text{def}}{=} \{ \text{cpu} \} : \{ \text{cpu} \} : \emptyset : S_1$$

$$S_2 \overset{\text{def}}{=} \{ \text{cpu} \} : (\{ \text{cpu} \} : \emptyset : S_2 + \emptyset : \{ \text{cpu} \} : S_2) + \emptyset : \{ \text{cpu} \} : \{ \text{cpu} \} : S_2$$

where $S_1$ offers supply $\text{cpu}$ during the first two time out of every three units of execution and $S_2$ offers the cpu for two out of every three time units where the precise timing of the offerings is nondeterministic. We may easily verify that $S_1$ is more generous than $S_2$ and such it may schedule at least as many tasks as $S_1$. Thus, the deterministic nature of $S_1$ makes it more generous than $S_2$.

Generalizing this example, we may also see that a periodic time partition with period $P$, duration $D \leq P$, and relative start time $t_0$, $\text{Part}_{t_0, D, P}$, defined in Section 3.3.2, is strongly more generous than the periodic resource model $\text{PRM}_{D, P}$ that guarantees $D$ time units of resource usage within every period $P$. The former presents one of the possible executions of the latter, this making it more generous, and able to schedule at least as many tasks.

### 4.2.2. Weak generosity

It turns out that the definition of strong generosity prevents us from comparing other supply models which one might be interested in comparing. For instance, supply $S_1$ above which offers a resource during the first two out of every three time units, would be intuitively considered as being more generous than supply $S_3 \overset{\text{def}}{=} \{ \text{cpu} \} : \emptyset : S_3$. However, $S_1$ is not strongly more generous than $S_3$, according to our definition and, for instance, although $S_1$ offers more resources than $S_3$ it fails to schedule task $T$ below which is in fact schedulable by the more stingy $S_3$:

$$T \overset{\text{def}}{=} \{ (\text{cpu}, 0) \} : [\emptyset : \text{FIN} + \{ (\text{cpu}, 0) \} : \{ (\text{cpu}, 0) \} : \text{FIN}]$$

21
Nonetheless, we would like to relax the notion of supply generosity to encompass a wider set of supplies at the expense of Lemma 4.5. Specifically, below we define a weaker notion of generosity which is considered within a restricted class of tasks. This definition is as follows.

**Definition 4.7.** A relation $R \in S \times S$ is a weak generosity relation if for all $(S_1, S_2) \in R$,

1. If $S_2 \rightarrow$ then $S_1 \rightarrow$.
2. If $S_2 \rightarrow$ and $S_1 \xrightarrow{\alpha} S'_1$ then we have that $S_2 \xrightarrow{\beta} S'_2$, $\beta \subseteq \alpha$ and $(S'_1, S'_2)$.

We write $\preceq_W$ for the largest weak generosity relation and we say that supply $S_1$ is weakly more generous than supply $S_2$, $S_2 \preceq_W S_1$, if there exists a weak generosity relation $R$ with $(S_1, S_2) \in R$.

It is now straightforward to see the following:

- Supplies $S_1$ and $S_3$ considered above are such that $S_1$ is weakly more generous than $S_3$.
- A partial supply $Part_{t_0, D, P}$ is weakly more generous than a partial supply $Part_{t_0, D', P}$, where $D' \leq D$.
- A periodic resource mode $PRM_{w, p}$, defined in Section 3.3.2, is weakly more generous than a periodic resource mode $PRM_{w', p}$, $w' \leq w$.
- A periodic resource mode $PRM_{2, 4}$ is not weakly more generous than a periodic resource mode $PRM_{1, 2}$. We may confirm this by considering the execution $PRM_{2, 4} \xrightarrow{[\tau]} PRM_{2, 4} \xrightarrow{[\theta]} PRM_{2, 4}$ and observing that it cannot be matched by $PRM_{1, 2}$ as required by the definition of weak generosity. Note that task $Task_{1, 2}$ is schedulable by supply $PRM_{1, 2}$ but it is not schedulable by $PRM_{2, 4}$.

Regarding the ability of weakly more generous supplies to schedule tasks we have the following result. Consider the class of periodic tasks $C$ with period $p$ and execution time $w$, $Task_{w, p}$, defined in Section 3.3.1. We may prove that:

**Lemma 4.8.** If task $T \in C$ is schedulable under supply $S_2$ and $S_1$ is weakly more generous than supply $S_2$ then $T$ is also schedulable under supply $S_1$.

**Proof:** The proof consists of showing that the following relation is a supply relation.

$$S = \{(T, t, w, p, S_1) \mid \exists S_2 \in S, \text{ supply relation } R \text{ and weak generosity relation } W \cdot (S_1, S_2) \in W \text{ and } (T, t, w, p, S_2) \in R, \text{ for some } e' \leq e\}$$

So, consider $(T, t, w, p, S_1) \in S$ and suppose there exist a supply $S_2$, a supply relation $R$ and a weak generosity relation $W$, such that $(S_1, S_2) \in W$ and $(T, t, w, p, S_2) \in R$ for some $e' \leq e$. Suppose $S_1 \xrightarrow{\alpha} S'_1$. We will show that $T \xrightarrow{\beta} T$ where $sat(\beta, \alpha)$, $\beta \leq_{T, t, w, p} \alpha$ and $(T, S'_1) \in R$. First note that since $S_1 \xrightarrow{\alpha} S'_1$ and $(S_1, S_2) \in W$, $S_2 \xrightarrow{\gamma} S'_2$, $\gamma \subseteq \alpha$ and $(S'_1, S'_2) \in W$. The following cases exist:
Finally, if \( e = w \), then \( t < p \), therefore \( T_{e',t,w,p} \rightarrow T_{e',t+1,w,p}, T_{e,t,w,p} \rightarrow T_{e,t+1,w,p}, (T_{e',t,w,p}, S'_2) \in \mathcal{S} \), and thus, \( (T_{e,t+1,w,p}, S'_1) \in \mathcal{R} \) as required.

- If \( e = w, e = e' \) and \( t < p \), then \( T_{e',t,w,p} \rightarrow T_{e',t+1,w,p}, T_{e,t,w,p} \rightarrow T_{e,t+1,w,p}, (T_{e',t,w,p}, S'_2) \in \mathcal{S} \), and thus, \( (T_{e,t+1,w,p}, S'_1) \in \mathcal{R} \) as required.

- If \( e = w, e < e' \) and \( t < p \), then \( T_{e',t,w,p} \rightarrow T_{e'',t+1,w,p} \), where \( e'' \in \{e', e' + 1\} \), depending on whether \( p \in \gamma \). In any case, \( e'' < e, T_{e,t,w,p} \rightarrow T_{e,t+1,w,p} \), \( (T_{e',t,w,p}, S'_2) \in \mathcal{S} \), and thus, \( (T_{e,t+1,w,p}, S'_1) \in \mathcal{R} \) as required.

- If \( e = w \) and \( t = p \), then since \( T_{e',t,w,p} \) is schedulable by \( S_2 \) it must be that \( e' = w \) and the proof follows as in the next case.

- If \( e < w, w - e < p - t, w - e' < p - t \) the following cases exist. If \( p \in \gamma \), then \( \tau \in \alpha \) and \( T_{e',t,w,p} \rightarrow (r,\pi) T_{e',t+1,w,p}, T_{e,t,w,p} \rightarrow (r,\pi) T_{e+1,t+1,w,p} \), where \( (T_{e'+1,t+1,w,p}, S'_2) \in \mathcal{S} \), and thus, \( (T_{e+1,t+1,w,p}, S'_1) \in \mathcal{R} \) as required. If \( p \notin \gamma \) and \( \tau \in \alpha \) then \( T_{e',t,w,p} \rightarrow (r,\pi) T_{e',t+1,w,p}, T_{e,t,w,p} \rightarrow (r,\pi) T_{e+1,t+1,w,p} \), where \( (T_{e'+1,t+1,w,p}, S'_2) \in \mathcal{S} \), and thus, \( (T_{e+1,t+1,w,p}, S'_1) \in \mathcal{R} \) as required. Finally, if \( p \notin \gamma \) and \( \tau \notin \alpha \) then \( T_{e',t,w,p} \rightarrow (r,\pi) T_{e',t+1,w,p}, T_{e,t,w,p} \rightarrow (r,\pi) T_{e+1,t+1,w,p} \), where \( (T_{e'+1,t+1,w,p}, S'_2) \in \mathcal{S} \), and thus, \( (T_{e+1,t+1,w,p}, S'_1) \in \mathcal{R} \) as required.

- If \( e < w, w - e < p - t \) and \( w - e' = p - t \), then the proof follows similarly to the first case of the previous clause.

Finally, if \( e < w \) and \( w - e = p - t \), then, since \( T_{e',t,w,p} \) is schedulable by \( S_2 \), \( e = e', \tau \in \gamma \) and thus, \( \tau \in \alpha \) and \( T_{e',t,w,p} \rightarrow (r,\pi) T_{e',t+1,w,p}, T_{e,t,w,p} \rightarrow (r,\pi) T_{e+1,t+1,w,p} \), where \( (T_{e'+1,t+1,w,p}, S'_2) \in \mathcal{S} \), and thus, \( (T_{e+1,t+1,w,p}, S'_1) \in \mathcal{R} \) which completes the proof.

**Example 4.9.** Consider a system composed of two applications competing for the usage of a single resource, the first consisting of the task set \( \text{Task}_{1,3} \parallel \text{Task}_{1,5} \) running under an EDF scheduler and the second consisting of the task set \( \text{Task}_{1,6} \parallel \text{Task}_{1,5} \) running under a RM scheduler. We may verify that the assignment of supply \( \text{PRM}_{1,5} \) to the first application and \( \text{PRM}_{2,5} \) to the second application leads to the schedulability of the system. This can be achieved by constructing the demand-processes of the two application and verifying that

1. \( \text{PRM}_{1,5} \) is weakly more generous than demand(\( \text{Task}_{1,3} \parallel \text{Task}_{1,5} \))
2. \( \text{PRM}_{2,5} \) is weakly more generous than demand(\( \text{Task}_{1,6} \parallel \text{Task}_{1,5} \)).

**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 developing a tool which implements our techniques for schedulability analysis and compositional scheduling of real-time systems. We plan to extend our work in order to handle dependencies between tasks. Furthermore, we would like to define the notion of a residual supply which captures the supply available after a system has its resource demands satisfied and which will enable to perform incremental scheduling of systems. 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


