

Cyclic Dependencies in Modular Performance Analysis

Bengt Jonsson
Dept. Information Technology
Uppsala University
Box 337, 751 05 Uppsala
bengt@it.uu.se

Lothar Thiele
Computer Engineering and
Networks Lab.
ETH Zurich, Switzerland
thiele@tik.ee.ethz.ch

Simon Perathoner
Computer Engineering and
Networks Lab.
ETH Zurich, Switzerland
perathoner@tik.ee.ethz.ch

Wang Yi
Dept. Information Technology
Uppsala University
Box 337, 751 05 Uppsala
yi@it.uu.se

ABSTRACT

In the analysis of component-based real-time systems, common questions are whether a particular workload can be processed in time on a given architecture, or whether a task set is schedulable given a certain availability of resources. An abstraction for modeling and analyzing such problems is the Modular Performance Analysis based on Real-Time Calculus (MPA-RTC), developed by Thiele et al. The formalism uses an abstract stream model to characterize both workload and availability of computation and communication resources. Components can then be viewed as stream transformers. The Real-Time Calculus has been used successfully on systems where dependencies between components, via either workload or resource streams, are acyclic. For systems with cyclic dependencies the foundations and performance of the formalism are less well understood.

In this paper, we develop a general operational semantics underlying the Real-Time Calculus, and use this to show that the behavior of systems with cyclic dependencies can be analyzed by fixpoint iterations. We characterize conditions under which such iterations give safe results, and also show how precise the results can be.

1. INTRODUCTION

Complex embedded systems very often consist of parallel or distributed computing elements that are exchanging data via some communication system. In addition, the applications that are running on these platforms can be described as communicating processes. In the case of real-time requirements, it is necessary to investigate the interaction of the processes and the communication tasks with the resources they are using. Because of the interference caused by the concurrent use of resources the evaluation of the timing behavior of the whole embedded system is a challenging task.

One approach to handling the associated computational complexity is to adopt a component-based approach where the verification and validation can be done in an incremental process. Based on appropriate abstractions, the interaction of each individual component with its environment and the available resources is analyzed and abstracted into a corresponding interface representation. The overall system behavior is then determined by an appropriate composition mechanism.

For example this approach can be successfully applied in a situation with static scheduling, in which each task is allocated pre-determined timing slots of CPU time. This approach, however, suffers problems of inflexibility. The paradigm of static fixed-priority scheduling offers more flexibility. In this case, the classical schedulability analysis uses maximal blocking time as a measure of available computation resources. In its standard form, however, it has problems to handle variabilities in task parameters, e.g., highly varying computation times and jitter.

The approach of Real-Time Calculus [7] provides a framework for a compositional analysis. Composition is possible in terms of processes (functional composition), scheduling policies (interaction) and resources (distributed operation). The underlying abstraction is able to handle complex event and resource patterns and therefore, can handle situations with a high degree of non-determinism. The approach characterizes each component by a time-invariant transfer function, which operates on upper and lower bounds on the number of events and computation resources available in all time intervals. Despite of the fact that the approach has been proven to be useful in case studies and investigations on benchmark applications, fundamental issues in linking the abstractions to an operational approach are not yet investigated.

A particular problem of great practical relevance is the handling of cyclic networks of components. The natural approach would be to compute the system characteristics captured in the Real-Time Calculus by means of a fixpoint computations, starting from some initial approximation. However, it is not known to what extent the resulting fixpoint is faithful to an underlying operational behavior of the system, or how to best choose initial approximations.

In this paper, we propose a simple operational model of distributed systems of processes and relate it to characteristics in the Real-Time Calculus. On this basis, we prove central properties about the faithfulness of fixpoints computed using the abstractions in Real-Time Calculus. The main result is a method that leads to the optimal fixpoint, i.e. makes best usage of the underlying abstraction. In addition, the approach presented in this paper is not restricted to Real-Time Calculus. The results can easily be transferred to related abstractions of streams, resources and their interactions.

The paper is organized as follows. In the next section, we introduce the basics of the framework for Modular Performance Analysis with Real-Time Calculus, accompanied by a motivating simple example. In Section 3, we present a general operational model of components and systems, and of specifications in some formalism for analyzing resource and timing properties, and prove correctness results for fixpoints. In Section 4, we specialize these results to the Real-Time Calculus, and discuss two techniques for obtaining initial approximations for iterations. Section 5 contains a simple example to illustrate sensitivity of fixpoints on parameters in boundary cases. Thereafter, we show the practical usefulness of our models on a simple case study and present concluding remarks.

Related Work. Jersak et al. [2] have considered a special case of cyclic dependencies in their periodic-with-jitter event model, in which there are functional cycles of events for task activation. The approach is limited in terms of the underlying abstraction and only informally makes statements about convergence properties. In particular, no results on the dependence of the fixpoint on initial conditions are provided.

The general problem of analyzing cycles in the RTC has been considered by Schiøler et al. [5]. They consider a similar framework as in our paper under the constraint that the time domain starts at 0. Hence they must use weaker equations than usually considered in the RTC (e.g., [7]) in order to establish guaranteed sound results. Their treatment relies on several assumptions. For instance, the result about the correctness of any fixpoint of the RTC equations ignores the potential problem of zero-delay cycles. These zero-delay cycles can occur if a continuous flow model is used for the system modeling. In addition, further properties of the obtained fixpoint solutions are assumed, e.g. “non-blocking”. The present paper gives a comprehensive and more detailed treatment of correctness of fixpoints. We also state the results in a more general setting of any specification formalism, which is based on characterizing system components as stream transformers.

The general topic of modeling and specifying systems specified as stream transformers goes back to Kahn networks [3] (extension to real-time, e.g., in [?]) and recurs in many works, e.g., on synchronous languages. We consider behaviors where the time domain is the set of real numbers. Furthermore, we do not compute fixpoints in order to obtain some actual behavior of the system, as e.g., in [3], where the ordering used reflects how much of a behavior is defined. Rather, we consider fixpoints of equations over *constraints* on such behaviors, for which the ordering reflects the strength of the obtained constraints.

2. REAL-TIME CALCULUS

In this section, we describe the framework of Modular Performance Analysis based on Real-Time Calculus. In order to introduce the problem, we also describe a simple system architecture for which the performance analysis is complicated by a cyclic dependency.

MPA-RTC is a compositional abstraction for the modeling and analysis of distributed real-time systems. The formalism has its roots in Network Calculus [4] developed to reason about timing properties of event and resource streams that flow through a network of computation and communication components. There are slightly different treatments of Real-

Time Calculus, depending on whether the time domain has an initial point (taken to be 0) or whether it goes unboundedly into the past. The presentation in this section will cover both cases.

2.1 Event Stream Model

A stream of events can be characterized by a *differential arrival function* $R : \mathbb{R} \times \mathbb{R} \mapsto \mathbb{N}$, where $R[s, t]$ denotes the sum of events that arrive in the time interval $s \leq \tau < t$ with $R[s, s] = 0$.

The Real-Time Calculus abstracts from concrete execution traces described by arrival functions R and bounds all possible traces of an event stream with a tuple of *arrival curves* $\alpha(\Delta) = [\alpha^l(\Delta), \alpha^u(\Delta)]$ where $\alpha^l(\Delta)$ denotes the lower arrival curve and $\alpha^u(\Delta)$ the upper arrival curve of the event stream.

Informally, a lower arrival curve $\alpha^l : \mathbb{R}^{\geq 0} \mapsto \mathbb{N}$ is a monotone superadditive function with $\alpha^l(0) = 0$, which states that in any half-open time interval of length Δ at least $\alpha^l(\Delta)$ events will arrive. An upper arrival curve $\alpha^u : \mathbb{R}^{\geq 0} \mapsto \mathbb{N}$ is a monotone subadditive function, with $\alpha^u(0) = 0$, which states that in any half-open time interval of length Δ at most $\alpha^u(\Delta)$ events will arrive.

We introduce the notation $R \models [\alpha^l, \alpha^u]$ to denote that $\alpha^l(\Delta) \leq R[s, s + \Delta] \leq \alpha^u(\Delta)$ for all $s \in \mathbb{R}$ and $\Delta \in \mathbb{R}^{\geq 0}$. For a given event stream described by an arrival function R , the tightest arrival curves α^l_R, α^u_R that model the stream are given by

$$\begin{aligned} \alpha^l_R(\Delta) &= \inf_{s \in \mathbb{R}} R[s, s + \Delta] \\ \alpha^u_R(\Delta) &= \sup_{s \in \mathbb{R}} R[s, s + \Delta] \end{aligned} \quad (1)$$

2.2 Resource Model

In analogy with differential arrival functions, the availability of a computation or communication resource is represented by a resource stream, which can be described by a *differential service function* $C : \mathbb{R} \times \mathbb{R} \mapsto \mathbb{R}^{\geq 0}$, where $C[s, t]$ denotes the sum of available resource units in the time interval $s \leq \tau < t$ with $C[s, s] = 0$. Continuing the analogy, the Real-Time Calculus abstracts from concrete service functions C and uses a tuple of lower and upper *service curves* $\beta(\Delta) = [\beta^l(\Delta), \beta^u(\Delta)]$ to model the availability of a resource. Informally, a lower service curve $\beta^l : \mathbb{R}^{\geq 0} \mapsto \mathbb{R}^{\geq 0}$ is a monotone superadditive function with $\beta^l(0) = 0$, which gives a lower bound $\beta^l(\Delta)$ on the number of available computation units in any interval of length Δ . An upper service curve $\beta^u : \mathbb{R}^{\geq 0} \mapsto \mathbb{R}^{\geq 0}$ is a monotone subadditive function with $\beta^u(0) = 0$, which gives an upper bound $\beta^u(\Delta)$ on the number of available resource units in any interval of length Δ .

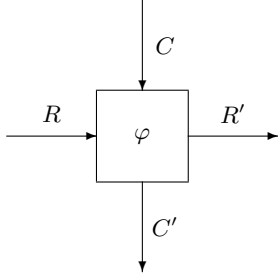
Again we use the notation $C \models [\beta^l, \beta^u]$ to denote that $\beta^l(\Delta) \leq C[s, s + \Delta] \leq \beta^u(\Delta)$ for all $s \in \mathbb{R}$ and $\Delta \in \mathbb{R}^{\geq 0}$. For a given concrete resource availability described by a service function C , the tightest service curves β^l_C, β^u_C that model the resource availability are given by

$$\begin{aligned} \beta^l_C(\Delta) &= \inf_{s \in \mathbb{R}} C[s, s + \Delta] \\ \beta^u_C(\Delta) &= \sup_{s \in \mathbb{R}} C[s, s + \Delta] \end{aligned} \quad (2)$$

2.3 Component Model

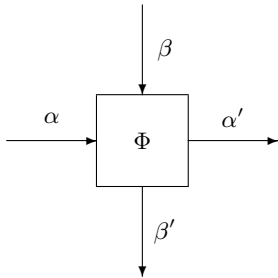
Components are the basic building blocks of a system. They are the implementation of tasks that process an event

stream and run on a (shared) resource, e.g. a computing or communication subsystem. An external view of a component is that it receives a stream of input events, and a stream of available resources, from which it produces a stream of outgoing events and a stream of remaining resources. The incoming and outgoing event streams can be modeled by differential arrival functions, R and R' , respectively, and the available and remaining resource by differential service functions, C and C' , respectively. This view is illustrated in the figure below.



In the figure, we use φ to denote the transformation which represents the behavior of the component. We can view φ as a transfer function from input to output, i.e., $(R', C') = \varphi(R, C)$.

In the MPA-RTC framework such a concrete component is modeled by an abstract component in the domain of arrival and service curves. When doing so, we use an arrival curve α and a service curve β to denote constraints on R and C . For any R and C with $R \models \alpha$ and $C \models \beta$, the component produces outputs R' and C' with $R' \models \alpha'$ and $C' \models \beta'$. The abstract component is illustrated in the figure below.



In the figure, we have used a function Φ to denote the behavior of the component. This function provides, for each input α, β , the tightest possible output arrival and service curves α', β' . The function Φ is different for different types of components with different behaviors.

A typical example for an abstract component in the context of the MPA-RTC modeling framework is a so-called *Greedy Processing Component* (GPC). It models a task that is triggered by the events of the incoming event stream which queue up in a FIFO buffer. The task processes the events in a greedy fashion, while being restricted by the availability of resources. The processing is performed within some specified execution time. For simplicity, we assume that it takes exactly one computational unit to process one event. This assumption can be relaxed by appropriate rescalings, which we will not consider further.

For this type of component, the following internal relations between the incoming arrival curves $[\alpha^l, \alpha^u]$ and service curves $[\beta^l, \beta^u]$ and the corresponding outgoing arrival

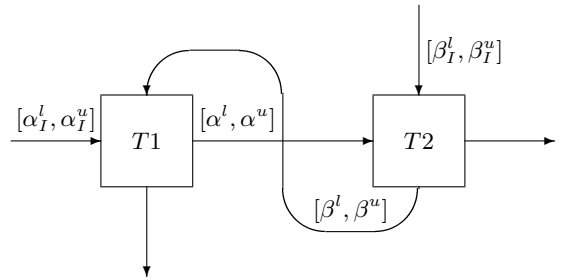
curves $[\alpha^l, \alpha^u]$ and service curves $[\beta^l, \beta^u]$ are stated and proved in [6]¹:

$$\begin{aligned} \alpha^u &= \min \{ (\alpha^u \otimes \beta^u) \oslash \beta^l, \beta^u \} \\ \alpha^l &= \min \{ (\alpha^l \oslash \beta^u) \otimes \beta^l, \beta^l \} \\ \beta^{lu}(\Delta) &= \max \{ \inf_{\Delta \leq \lambda} \{ \beta^u(\lambda) - \alpha^l(\lambda) \}, 0 \} \\ \beta^{lu}(\Delta) &= \sup_{0 \leq \lambda \leq \Delta} \{ \beta^l(\lambda) - \alpha^u(\lambda) \} \end{aligned} \quad (3)$$

An important observation here is that these relations are correct for $\text{RTC}_{-\infty}$ (under the assumption of bounded buffer occupancy), but they are not correct for RTC_0 . In particular, the equations for β^{lu} and α^l are too tight. In this case, the relations derived in [?] must be used: $\alpha^l = \alpha^l \otimes \beta^l$ and $\beta^{lu}(\Delta) = \sup_{0 \leq \lambda \leq \Delta} \{ \beta^u(\lambda) - \alpha^l(\lambda) \}$.

2.4 A Simple Example

Let us now consider a simple system for which the analysis is inhibited by a cyclic dependency. The structure of the system is shown in the figure below.



It consists of two components, $T1$ and $T2$. Component $T1$ is triggered by input events from the environment and produces output events for $T2$. However, $T1$ receives only the computation resources which are left over by $T2$. Component $T2$ receives input events from $T1$ and outputs events to the environment. It receives computation resources from the environment, and passes on the remaining computation resources to $T1$. For simplicity, we assume that both $T1$ and $T2$ need exactly 1 unit of resources to service an event. Further we assume that the incoming stream of events from the environment to $T1$ is constrained by the arrival curves $[\alpha_T^l, \alpha_T^u]$, and that the incoming stream of resources to $T2$ is constrained by the service curves $[\beta_T^l, \beta_T^u]$. The system architecture implements a preemptive fixed-priority scheduling scheme, where $T2$ has a higher priority than $T1$. We do not consider any overhead for the context switching.

We want to compute tight bounds for the event streams from $T1$ to $T2$ and the resource stream from $T2$ to $T1$. However, the characterization of these streams is not straightforward since there is a cyclic dependency between them ($T1$ triggers $T2$ while $T2$ preempts $T1$). A particular estimate of these streams is a quadruple $(\alpha^l, \alpha^u, \beta^l, \beta^u)$. The transfer functions in equation (3) imply the following relationships between the curves:

$$\begin{aligned} \alpha^u &= \min \{ (\alpha_T^u \otimes \beta^u) \oslash \beta^l, \beta^u \} \\ \alpha^l &= \min \{ (\alpha_T^l \oslash \beta^u) \otimes \beta^l, \beta^l \} \\ \beta^{lu}(\Delta) &= \max \{ \inf_{\Delta \leq \lambda} \{ \beta_T^u(\lambda) - \alpha^l(\lambda) \}, 0 \} \\ \beta^{lu}(\Delta) &= \sup_{0 \leq \lambda \leq \Delta} \{ \beta_T^l(\lambda) - \alpha^u(\lambda) \} \end{aligned} \quad (4)$$

¹See appendix for the definition of the operators \otimes and \oslash .

Let us denote the quantities $(\alpha_I^l, \alpha_I^u, \beta_I^l, \beta_I^u)$ by Σ_I , and the quantities $(\alpha^l, \alpha^u, \beta^l, \beta^u)$ by Σ_h ; let us ignore the specification of the output streams. We denote the pair (Σ_I, Σ_h) with Σ and we use Ψ to represent the mapping from one specification Σ to another Σ' by means of the equations (4)

It is then natural to expect that a specification of the behavior of the system can be obtained as a fixpoint of Ψ , i.e. as a solution of the equation $\Sigma = \Psi(\Sigma)$. A natural way to compute such a fixpoint is to start from some initial approximation, i.e. by starting from some tuple Σ^0 , and computing the sequence $\Sigma^0, \Sigma^1, \Sigma^2, \dots$ where $\Sigma^{k+1} = \Psi(\Sigma^k)$, in the hope that the sequence will converge to a limit Σ^* .

However, the correctness of such a fixpoint iteration in the context of MPA-RTC has not been formally justified so far. In particular several questions need to be answered:

- Will any fixpoint of Ψ correctly characterize all possible behaviors of the system?
- Can there be several fixpoints?
- If so, is there an optimal fixpoint (i.e. one that provides tighter bounds than all others)?
- Can an (optimal) fixpoint be computed as the limit of a sequence $\Sigma^0, \Sigma^1, \Sigma^2, \dots$ of approximations?
- Will the iteration always converge to a limit Σ^* ?
- If so, how does Σ^* correspond to the behavior of the system?
- How to choose the initial approximation Σ_0 ?

To illustrate that fixpoints are not in general unique, consider again the above system. Let $\beta_I^l(\Delta) = \beta_I^u(\Delta) = \Delta$, and let an event arrive every second time unit, i.e.,

$$\alpha_I^l(\Delta) = \lfloor \frac{\Delta}{2} \rfloor \quad \text{and} \quad \alpha_I^u(\Delta) = \lceil \frac{\Delta}{2} \rceil$$

The optimal (and correct) fixpoint is the one which forces the same behavior on the internal event stream as on the input event stream, i.e., $[\alpha^l, \alpha^u] = [\alpha_I^l, \alpha_I^u]$. However, a much worse fixpoint is one which gives no information at all, given by a specification of the event stream $[\alpha^l, \alpha^u]$ such that

$$\alpha^l(\Delta) = 0 \quad \text{and} \quad \alpha^u(\Delta) = \lceil \Delta \rceil$$

and a specification of the resource stream $[\beta^l, \beta^u]$ such that

$$\beta^l(\Delta) = 0 \quad \text{and} \quad \beta^u(\Delta) = \Delta$$

To investigate these questions, in the next section we provide a more general framework for specifying quantitative properties in component-based systems, in which we prove results about the above questions under certain assumptions.

3. STREAMS AND FIXPOINTS

In this section, we prove results about correctness of fixpoint calculations, which are valid for a class of formalisms that specify quantitative properties of component-based systems, such as MPA or Symta/S, see [?]. We provide an abstract description of such a formalism, and thereafter define conditions under which fixpoints are correct. We here give a treatment which considers both the case where system behavior starts at an initial time point, taken to be 0, and when it has no initial time point, i.e., the time domain is the set from \mathbb{R} of all real numbers.

Streams. We assume that the behavior of a system or a component is observed at a set V of “streams”. At each stream $v \in V$, a behavior σ realizes a function $\sigma(v) : (\mathbb{R} \times \mathbb{R}) \mapsto \mathbb{R}^{\geq 0}$ from time-intervals to observations, which is monotone, i.e., decreasing in its first argument and increasing in its second. We take $\mathbb{R}^{\geq 0}$ as the range of observations, since we intend to model accumulation of some resource or observable. For instance $\sigma(v)(s, t)$ could denote the number of events that have passed or the amount of CPU resources that have been available in the interval $[s, t]$. If time starts at 0, we assume $0 \leq s$ of course. For a set V of streams, let $Beh(V)$ denote the set of behaviors on streams V . The restriction of a behavior σ to a set V of streams is denoted $\sigma|_V$

Component Behavior. A component is equipped with a set of input streams V_I and a set V_O of output streams. Its behavior can be characterized by a *behavior mapping* $\varphi : Beh(V_I) \mapsto Beh(V_O)$. The idea is that for any input behavior σ_I in $Beh(V_I)$, the result $\varphi(\sigma_I)$ gives a behavior $\varphi(\sigma_I)(v)$ on output stream v .

System Behavior. Let us consider a system of components. The system has a set V_I of external input streams, and a set V_O of internal streams and external output streams. Each stream in V_O is an output stream of some components, some of which go to other components and others to the environment. By putting together all component behavior transformers, we get a *system behavior transformer* $\psi : Beh(V_I \cup V_O) \mapsto Beh(V_I \cup V_O)$ which maps any behavior on the streams of the system to another behavior on system streams. The mapping ψ will preserve the behavior on input streams, and will generate behaviors on internal and output streams according to the behavior mappings of components.

When forming a system from deterministic components, it is reasonable to expect that any behavior σ_I on input streams V_I induces a unique behavior on the output and internal streams. We will now establish this result under an assumption that the system is *simulatable*, as defined in the following definition. This property is analogous to, e.g., the property of having no zero-cycles in synchronous languages. We first consider the case when time starts at 0, thereafter indicate how things change for unbounded past.

Let a *time vector* on V be a mapping $\tau : V \mapsto (\mathbb{R}^{\geq 0} \cup \{-\infty\})$ from streams in V to nonnegative time values or the special value $-\infty$. For a value $t \in \mathbb{R} \cup \{-\infty\}$, let \bar{t} denote the time vector which is t at all streams. We say $\tau \leq \tau'$ if $\tau(v) \leq \tau'(v)$ for all streams v . We say that two behaviors σ and σ' *agree upto* τ , denoted $\sigma \simeq_\tau \sigma'$, if $\sigma(v)(t) = \sigma'(v)(t)$ whenever $t \leq \tau(v)$ for all streams $v \in V$ and $\sigma(v) = \sigma'(v)$ for all streams $v \notin V$. Intuitively, time vectors will be used to denote time points at which the simulation will make a snapshot. A time vector needs not to map all streams to the same time value, since it may be possible to know the state of some streams further in time than other streams. The idea is that a simulation should produce snapshots of system state at successive time vectors.

Definition 1. A system, represented by behavior transformer ψ , is *simulatable* if for each input behavior $\sigma_I \in Beh(V_I)$ there is an increasing sequence τ_0, τ_1, \dots of time vectors on V_O , called the *simulation sequence* for σ_I , with $\tau_0 = -\infty$ and $\lim_{i \rightarrow \infty} \tau_i(v) = \infty$ for all $v \in V_O$, such that for

all behaviors $\sigma, \sigma' \in \text{Beh}(V_I \cup V_O)$ with $\sigma|_{V_I} = \sigma'|_{V_I} = \sigma_I$ we have

for all $i \geq 0$: $\sigma \simeq_{\tau_i} \sigma'$ implies $\psi(\sigma) \simeq_{\tau_{i+1}} \psi(\sigma')$ \square

Note that $\sigma \simeq_{\tau_i} \sigma'$ implies that σ and σ' agree on V_I . Intuitively, a system is simulatable if one can advance time vectors stepwise, at each step compute new outputs from previously known inputs for some longer time, and such that the whole behavior can be obtained in ω steps.

For a simulatable system, with behavior transformer ψ , with given input behavior σ_I , we can construct the resulting system behavior as the limit of a sequence of behaviors, as follows. Let $\tau_0, \tau_1, \tau_2, \dots$ be the simulation sequence for σ_I . Define the sequence of behaviors $\sigma^0, \sigma^1, \sigma^2, \dots$, where σ^0 is any behavior with $\sigma^0|_{V_I} = \sigma_I$ and $\sigma^{i+1} = \psi(\sigma^i)$. We have that the sequence $\sigma^0, \sigma^1, \sigma^2, \dots$ converges, since $\sigma^i \simeq_{\tau_i} \sigma^{i+j}$ for any $i, j \geq 0$ by simulatability. The limit σ , which we can denote by $\psi^\omega(\sigma_I)$, is the actual behavior.

Let us finally consider how to adapt this to the situation when time goes unboundedly into the past. Then there is no initial time point for defining behaviors. Here, let τ be a real-valued time vector on V_O . We say that the system is *simulatable from* τ if for each input behavior $\sigma_I \in \text{Beh}(V_I)$ there is an increasing sequence τ_0, τ_1, \dots of time vectors on V_O , starting with $\tau_0 = \tau$, called the *simulation sequence for* σ_I *from* τ , with $\lim_{i \rightarrow \infty} t_i(v) = \infty$ for all $v \in V_O$, such that for all behaviors $\sigma, \sigma' \in \text{Beh}(V_I \cup V_O)$ with $\sigma|_{V_I} = \sigma'|_{V_I} = \sigma_I$ we have that $\sigma \simeq_{\tau_i} \sigma'$ implies $\psi(\sigma) \simeq_{\tau_{i+1}} \psi(\sigma')$ for all $i \geq 0$. We say that the system is *simulatable* if it is simulatable from any real-valued time vector.

To construct system behavior for some given input behavior $\sigma_I \in \text{Beh}(V_I)$, we assume that system behavior is defined up to a specific time vector τ by some behaviour σ^0 with $\sigma^0|_{V_I} = \sigma_I$. We are interested in σ^0 up to time vector τ : this requires that σ^0 is a possible system behavior up to τ , i.e., that $\sigma^0 \simeq_\tau \psi(\sigma^0)$. If the system is now simulatable from τ , we can construct the system behavior, starting from σ^0 , to obtain the actual behavior $\psi^\omega(\sigma^0)$ in the same way as above. Intuitively $\psi^\omega(\sigma^0)$ is the uniquely defined behavior of the system, which agrees with σ^0 upto time vector τ (and with σ_I on V_I).

Component Specifications. Let us now consider specifications of behaviors. Abstractly, a specification Σ of the behavior on a set V of streams is a mapping $V \mapsto 2^{(\mathbb{R}^{\geq 0} \mapsto \mathbb{R}^{\geq 0})}$, which maps each stream $v \in V$ to a set $\Sigma(v)$ of behaviors on stream v (of course, it should satisfy some properties of being “well-behaved”). As one of many possibilities, a specification can prescribe bounds on the number of events transmitted over a stream in certain time intervals. For a set V of streams, let $\text{spec}(V)$ denote the set of specifications on streams V . If both σ and Σ use the set V of streams, we use $\sigma \models \Sigma$ to denote $\forall v \in V. \sigma(v) \in \Sigma(v)$.

A component with input streams V_I and output streams V_O , characterized by the behavior mapping φ , can be specified by a *specification mapping* $\Phi : \text{spec}(V_I) \mapsto \text{spec}(V_O)$ from specifications of input streams to specifications of output streams. For any specification Σ_I of input behavior the result $\Phi(\Sigma_I)$ gives a specification $\Phi(\Sigma_I)(v)$ of behavior on each output stream v . This mapping should be *correct wrp. to* φ , i.e., have the property that if $\sigma_I(v_i) \models \Sigma_I(v_i)$ for all input streams $v_i \in V_I$, then $\varphi(\sigma_I)(v_O) \models \Phi(\Sigma_I)(v_O)$ for all

output streams $v_O \in V_O$.

System Specifications. Just as for system behaviors, we can put together all component specification mappings into a *system specification transformer* $\Psi : \text{spec}(V_I \cup V_O) \mapsto \text{spec}(V_I \cup V_O)$. The question we want to study is the following: Assume Ψ , and a specification Σ_I of behaviors on V_I , let σ_I be any behavior on V_I with $\sigma_I \models \Sigma_I$. Will the behavior $\psi^\omega(\sigma_I)$ of the system, uniquely determined by σ_I , satisfy the “limit” of a sequence $\Sigma^0, \Psi(\Sigma^0), \Psi(\Psi(\Sigma^0)), \dots$? Questions include:

- How can we choose Σ^0 ?
- When will it converge?
- If it converges, does it specify $\psi^\omega(\sigma_I)$?

Let us make some definitions. For a time vector τ on V , we say that a behavior σ *satisfies a specification* Σ *upto* τ , denoted $\sigma \models_{\leq \tau} \Sigma$ if there is a behavior σ' with $\sigma \simeq_\tau \sigma'$ such that $\sigma' \models \Sigma$.

In the rest of this section, we assume a simulatable system represented by system behavior transformer ψ . We let Ψ be a system specification transformer which is correct wrp. to ψ . We assume a specification Σ_I of behaviors on input streams V_I . We assume σ_I to be a behavior on input streams with $\sigma_I \models \Sigma_I$. Let $\tau_0, \tau_1, \tau_2, \dots$ be the simulation sequence for σ_I , and let $\psi^\omega(\sigma_I)$ be the actual behavior of the system.

We shall assume that for an actual system behavior σ with $\sigma|_{V_I} \models \Sigma_I$, there is a strongest specification Σ which agrees with Σ_I on V_I , such that $\sigma \models \Sigma$. We denote this specification by Σ_σ

THEOREM 1. *If the specification Σ^0 is satisfiable and also agrees with Σ_I on V_I , then*

$$\psi^\omega(\sigma_I) \models_{\leq \tau_i} \Psi^i(\Sigma^0)$$

for all $i \geq 0$.

PROOF. Let $\tau_0, \tau_1, \tau_2, \dots$ be the simulation sequence for σ_I . Let $\psi^\omega(\sigma_I)$ be the actual system behavior given σ_I , obtained as the limit of the sequence $\sigma^0, \sigma^1, \sigma^2, \dots$, where σ^0 is any behavior with $\sigma^0|_{V_I} = \sigma_I$, and where $\sigma^{i+1} = \psi(\sigma^i)$. We shall prove by induction over i that $\sigma^i \models_{\leq \tau_i} \Psi^i(\Sigma^0)$. The base case $\sigma^0 \models_{\leq -\infty} \Sigma^0$ follows from the assumption that $\sigma^0|_{V_I} = \sigma_I$ and that Σ^0 agrees with Σ_I on V_I and is satisfiable. For the inductive step, assume $\sigma^i \models_{\leq \tau_i} \Psi^i(\Sigma^0)$, i.e., that there is a behavior σ' with $\sigma^i \simeq_{\tau_i} \sigma'$ such that $\sigma' \models \Psi^i(\Sigma^0)$. By correctness of Ψ with respect to ψ , we get that $\psi(\sigma') \models \Psi^{i+1}(\Sigma^0)$. Since the system represented by ψ is simulatable, we get $\psi(\sigma^i) \simeq_{\tau_{i+1}} \psi(\sigma')$. Hence $\psi(\sigma^i) \models_{\leq \tau_{i+1}} \Psi^{i+1}(\Sigma^0)$. The conclusion of the theorem follows by noting that $\psi^\omega(\sigma_I) \simeq_{\tau_i} \sigma^i$ for all i . \square

Convergence. Theorem 1 does not say anything about convergence of the sequence $\Sigma^0, \Psi(\Sigma^0), \Psi^2(\Sigma^0), \dots$. Let us formalize the conditions under which this can be attained.

For a set V of streams, the relation \models introduces a natural partial order \sqsubseteq on the set $\text{spec}(V)$ of specifications over V , by defining $\Sigma \sqsubseteq \Sigma'$ iff $\sigma \models \Sigma$ implies $\sigma \models \Sigma'$ for any $\sigma \in \text{Beh}(V)$. We shall assume that $\text{spec}(V)$ with the partial order \sqsubseteq constitutes a cpo, i.e., that any chain $\Sigma^0 \sqsubseteq \Sigma^1 \sqsubseteq \Sigma^2 \sqsubseteq \dots$ has a least upper bound $\bigsqcup_{i \geq 0} \Sigma^i$. We finally assume

that any specification Σ is a *safety property*, which means that if $\forall \tau. \sigma \models_{\leq \tau} \Sigma$, then also $\sigma \models \Sigma$.

We shall assume that for any behavior $\sigma \in \text{Beh}(V)$ there is a least (strongest) specification which is satisfied by σ .

A specification mapping Φ is *monotone* if $\Sigma \sqsubseteq \Sigma'$ implies $\Phi(\Sigma) \sqsubseteq \Phi(\Sigma')$. It is *continuous* if

$$\Phi(\bigsqcup_{i \geq 0} \Sigma^i) = \bigsqcup_{i \geq 0} \Phi(\Sigma^i)$$

for any chain $\Sigma^0 \sqsubseteq \Sigma^1 \sqsubseteq \dots$.

We can now strengthen Theorem 1, so that it also guarantees convergence.

THEOREM 2. *Assume, in addition to previous assumptions, that Ψ is monotone and continuous. Then, among all specifications which agree with Σ_I on V_I , and are satisfied by at least one actual system behavior σ with $\sigma|_{V_I} \models \Sigma_I$, Ψ has a unique smallest fixpoint Σ^* which is satisfied by all such behaviors. Furthermore Σ^* can be obtained as the limit of the sequence $\Sigma^0, \Psi(\Sigma^0), \Psi^2(\Sigma^0), \dots$ of approximations if Σ^0 is a specifications which agree with Σ_I on V_I , and is satisfied by at least one actual system behavior σ with $\sigma|_{V_I} \models \Sigma_I$, and is such that $\Sigma^0 \sqsubseteq \Sigma^*$.*

PROOF. We continue using the previously introduced notation. Let σ be some behavior of the system such that $\sigma|_{V_I} \models \Sigma_I$. Let Σ_σ be the strongest specification, which agrees with Σ_I on V_I , such that $\sigma \models \Sigma_\sigma$. By correctness of Ψ we have $\Sigma_\sigma \sqsubseteq \Psi(\Sigma_\sigma)$. By monotonicity of Ψ we then get $\Psi^k(\Sigma_\sigma) \sqsubseteq \Psi^{k+1}(\Sigma_\sigma)$ for all $k \geq 0$. This means that the sequence $\Sigma_\sigma, \Psi(\Sigma_\sigma), \Psi^2(\Sigma_\sigma), \dots$ converges to a fixpoint Σ^* . By Theorem 1, and the assumption that all properties are safety properties, $\sigma'' \models \Sigma^*$ for any behavior σ'' of the system such that $\sigma''|_{V_I} \models \Sigma_I$.

Since σ was an arbitrary initial behavior, we get the same fixpoint Σ^* if we repeat the above procedure with any other behavior σ' with $\sigma'|_{V_I} \models \Sigma_I$, and repeat the above procedure.

We have proven the theorem for initial approximations of form Σ_σ for some system behavior. It remains to consider the case where $\Sigma_\sigma \sqsubseteq \Sigma^0 \sqsubseteq \Sigma^*$ for some σ . The theorem then follows by noting that by monotonicity $\Psi^k(\Sigma_\sigma) \sqsubseteq \Psi^k(\Sigma^0) \sqsubseteq \Sigma^*$ for all $k \geq 0$, implying that also $\Sigma^0, \Psi(\Sigma^0), \Psi^2(\Sigma^0)$ converges to Σ^* . \square

Intuitively, Theorem 2 states that among all specifications which agree with Σ_I on V_I , and are satisfied by at least one actual system behavior σ with $\sigma|_{V_I} \models \Sigma_I$, there is a unique least fixpoint which is satisfied by all such behaviors. The theorem furthermore states that this fixpoint can be obtained by iteration from an initial approximation, which characterizes one possible system behavior. Thus, if we can construct one system behavior, we can get a specification of all possible system behaviors by iterating from this one behavior to a fixpoint.

The case where time starts at $-\infty$. Let us indicate how the results in this section work out when time has no initial point, and how Theorems 1 and 2 adapt.

We use the notation introduced previously. Let τ be a timevector. Let σ^0 be a behavior with $\text{beh}^0|_{V_I} \models \Sigma_I$ such that $\sigma^0 \simeq_\tau \psi(\sigma^0)$, and let $\tau_0, \tau_1, \tau_2, \dots$ be the simulation sequence for σ^0 from τ . Define $\text{Cont}(\sigma^0, \tau)$ as the set of behaviors σ such that $\sigma|_{V_I} \models \Sigma_I$ and $\sigma \simeq_\tau \sigma^0$. We can now derive the following two theorems.

THEOREM 3. *If the specification Σ^0 is satisfied by σ^0 , then*

$$\psi^\omega(\sigma^0) \models_{\leq \tau_i} \Psi^i(\Sigma^0)$$

for all $i \geq 0$.

PROOF. Analogous to that for Theorem 1. \square

THEOREM 4. *Assume that Ψ is monotone and continuous. Then, among all specifications which agree with Σ_I on V_I , and are satisfied by at least one actual system behavior in $\text{Cont}(\sigma^0, \tau)$, the transformer Ψ has a unique smallest fixpoint Σ^* which is satisfied by all behaviors $\sigma \in \text{Cont}(\sigma^0, \tau)$ with $\sigma|_{V_I} \models \Sigma_I$. Furthermore Σ^* can be obtained as the limit of the sequence $\Sigma^0, \Psi(\Sigma^0), \dots$ of approximations if Σ^0 is a specification which agrees with Σ_I on V_I , and is satisfied by at least one actual system behavior in $\text{Cont}(\sigma^0, \tau)$, and satisfies $\Sigma^0 \sqsubseteq \Sigma^*$.*

PROOF. Analogous to that for Theorem 2. \square

4. FIXPOINTS IN RTC

In this section, we transfer the results of the preceding section to RTC. We must then first show that the general assumptions introduced in Section 3 hold for RTC. This will concern both the version of RTC with 0 as initial time point and the version where time starts at $-\infty$.

Consider a system with set V of streams. In RTC, a behavior σ of the system maps each event stream $v \in V$ to an arrival function R , and maps each resource stream $v \in V$ to a service function C . To transfer results from Section 3, we should check that the system is simulatable. Not all systems need to be simulatable, but sufficient conditions are given in the following proposition.

PROPOSITION 1. *If there is a nonzero lower bound on the time needed to process an event by a component, then a system which does not have any cycle of resource streams is simulatable.*

PROOF. (Sketch) For any component, its future output can be foreseen until the next point in time when it receives an event, or the status of its resource input is changed. These points in time form an increasing sequence of time points that converges to ∞ . Furthermore, the next such time point is always coinciding with the arrival of an event at some component (either from the environment or from another component). These arrivals happen at an increasing sequence of time points, which can be uniquely inferred from the current time point and the given behavior on input streams. \square

Concerning specifications, we should check that, for a set V of streams the set of specifications $\text{spec}(V)$ on V forms a cpo. Recall that a specification $\Sigma \in \text{spec}(V)$ maps each event stream $v \in V$ to an arrival curve of form $[\alpha^l, \alpha^u]$, and maps each resource stream $v \in V$ to a service curve of form $[\beta^l, \beta^u]$. In order to make $\text{spec}(V)$ a cpo, we augment the set of arrival curves by curves $[\alpha^l, \alpha^u]$ in which $\alpha^u(\Delta) = \infty$ for all Δ with $\Delta \geq \Delta_0$ for some $\Delta_0 > 0$. Upper service curves are bounded by the constraint $\beta^u(\Delta) \leq \Delta$, so we need not to make this extension for service curves. With this augmentation ($\text{spec}(V), \sqsubseteq$) forms a cpo. It is also easy to see that all specifications in $\text{spec}(V)$ are safety properties.

We should also check that specification transformers are correct, monotone and continuous: this is checked for each case. For instance, the equations (3) are monotone and continuous. Finally, for each behavior σ , there is a strongest specification which is satisfied by σ : this is obtained as in Equations (1) and (2) in Section 2.

We can now transfer the results from the previous section. Assume a system with input streams V_I and output streams V_O , which has no cycle formed by only resource streams. Assume a nonzero lower bound on the time needed to process an event by a component. Let Ψ be the established RTC equations for the streams V of the system: there are correct versions of these both for the case where time starts at 0, and where it starts at $-\infty$. Let Σ_I be a specification of input streams V_I . Then Theorem 2 holds for RTC where time starts at 0. For RTC where time starts at $-\infty$, we can even prove a slightly stronger version, namely the following.

THEOREM 5. *Assume the Σ_I allows at least one system behavior which is eventually periodic; this happens, e.g., if the system is not fully loaded. Then, among all satisfiable specifications which agree with Σ_I on V_I , Ψ has a unique smallest fixpoint Σ^* . The fixpoint Σ^* is satisfied by all actual system behaviors σ with $\sigma|_{V_I} \models \Sigma_I$. Furthermore Σ^* can be obtained as the limit of the sequence $\Sigma^0, \Psi(\Sigma^0), \Psi^2(\Sigma^0), \dots$ of approximations if Σ^0 is any satisfiable specification which agrees with Σ_I on V_I , and satisfies $\Sigma^0 \sqsubseteq \Sigma^*$.*

PROOF. The difference, in comparison with Theorem 2, is that it is enough to start the iteration with a satisfiable specification Σ^0 ; it need not be satisfied by any actual system behavior. To prove this extension, let σ_p be an eventually periodic actual system behavior. We note that by Theorem 1, using that σ_p is eventually periodic, there is for each Δ an integer k such that σ_p satisfies $\Psi^i(\Sigma^0)$ on all intervals smaller than Δ whenever $i \geq k$. By further considering the structure of RTC equations, we conclude that

$$\lim_{i \rightarrow \infty} \Psi^i(\Sigma^0)(\Delta) \geq \lim_{i \rightarrow \infty} \Psi^i(\Sigma_{\sigma_p})(\Delta)$$

for all Δ . The theorem follows. \square

For RTC, in which the time domain goes to $-\infty$, we can prove a stronger analogue of Theorem 4.

THEOREM 6. *Let τ be a time vector, and let σ^0 be a behavior with $\sigma^0|_{V_I} \models \Sigma_I$ such that $\sigma^0 \simeq_\tau \psi(\sigma^0)$. Assume that Σ_I is satisfied by at least one system behavior σ_p which is eventually periodic with $\sigma^0 \simeq_\tau \sigma_p$. Then, among all specifications Σ which agree with Σ_I on V_I , and satisfy $\sigma^0 \models_{\leq \tau} \Sigma$, there is a unique smallest fixpoint Σ^* of Ψ . This fixpoint Σ^* is satisfied by all system behaviors $\sigma \in \text{Cont}(\sigma^0, \tau)$ with $\sigma|_{V_I} \models \Sigma_I$. Furthermore Σ^* can be obtained as the limit of the sequence $\Sigma^0, \Psi(\Sigma^0), \dots$ of approximations if Σ^0 is a specification which agrees with Σ_I on V_I , and satisfy $\sigma^0 \models_{\leq \tau} \Sigma^0$ and $\Sigma^0 \sqsubseteq \Sigma^*$.*

PROOF. Analogous to the previous theorem. \square

Theorems 5 and 6 suggest a methodology for finding the optimal fixpoint of Ψ .

1. Construct some behavior σ of the system, such that $\sigma|_{V_I} \models \Sigma_I$, and such that σ satisfies the optimal fixpoint of Ψ . Such a behavior could be found, by constructing a simulation which is an actual system behavior. The task is made easier by the observation that

we need to find only one behavior, and can choose one which is as regular as possible, e.g., as an infinitely repeating periodic behavior.

2. Let Σ_σ be a tightest specification of the above behavior, and use Σ_σ as an initial approximation Σ^0 .
3. An alternative way to construct an initial approximation Σ^0 is by analytic techniques using long-term rates, as described in the next section.
4. Perform fixpoint iteration by computing the sequence $\Sigma^0, \Psi(\Sigma^0), \dots$. This is guaranteed to converge to a limit Σ^* which is the optimal fixpoint of Ψ .

To summarize: the recommended way to compute fixpoints by iteration is to start from a strong initial approximation, which is included in the sought optimal fixpoint and thereafter iterate towards a fixpoint. We observe that the dual approach – starting from a very weak initial approximation and iterate towards a stronger solution – will in most cases yield quite poor precision.

4.1 Obtaining an Initial Approximation

Theorem 5 provides a guarantee that a fixpoint solution will exist in RTC, provided that there is some solution to the equations. An initial approximation can be constructed by simulation. In this section, we will further develop another technique for constructing an initial approximation, presented by Schiøler et al. [5].

We consider the example system from Section 2.4, and assume that the externally provided streams of events and resources are constrained by monotone curves $[\alpha_I^l, \alpha_I^u]$ and $[\beta_I^l, \beta_I^u]$ which are superadditive and subadditive, respectively. We assume that these curves are consistent, i.e., that they allow at least one flow. We go on to deduce some properties.

By a lemma of Michael Fekete [1], if a monotone function f is subadditive or superadditive, then the limit $\lim_{t \rightarrow \infty} \frac{f(t)}{t}$ exists. So, define

$$\begin{aligned} A^l &= \lim_{t \rightarrow \infty} \frac{\alpha_I^l(t)}{t} & A^u &= \lim_{t \rightarrow \infty} \frac{\alpha_I^u(t)}{t} \\ B^l &= \lim_{t \rightarrow \infty} \frac{\beta_I^l(t)}{t} & B^u &= \lim_{t \rightarrow \infty} \frac{\beta_I^u(t)}{t} \end{aligned} \quad (5)$$

Intuitively, these are bounds on long-term rates.²

We conclude that $A^l \leq A^u$ and $B^l \leq B^u$, otherwise the curves would be inconsistent. It follows that the specification allows behaviors which do not generate overflow if and only if $2A^l \leq B^u$. If so, then the specification Σ^0 , which on the internal event stream has the arrival curve $[\alpha^l, \alpha^u]$ with

$$\alpha^l(\Delta) = \lfloor A^l \Delta \rfloor \quad \text{and} \quad \alpha^u(\Delta) = \lceil A^l \Delta \rceil$$

and on the internal resource stream has the service curve $[\beta^l, \beta^u]$ with

$$\beta^l(\Delta) = \beta^u(\Delta) = \Delta(B^u - A^l)$$

is satisfiable. It is not difficult to show that Σ^0 is stronger than the tightest possible specification of a behavior which does not generate overflow. Thus, Σ^0 satisfies the conditions

²Schiøler et al. do not prove the existence of these limits like we do, but rather assume that they exist, and remark that they do so in several common situations

for an initial approximation in Theorem 5, so that we can use it in iterative generation of the optimal fixpoint.

For systems with many components, and a complicated structure, the initial approximation can be obtained by solving a system of linear equations to get a long-term rate for each stream; however, the essential idea is the same as we just described for this small system. This way of treating complex system structures is detailed by Schiöler et al. [5].

5. ON PRECISION OF FIXPOINTS

The results in Sections 3 and 4 show that fixpoints of equations in RTC are correctly satisfied by system behaviors. However, so far we have not considered to analyze preciseness of obtained fixpoints.

In this subsection, we consider a very simple example to illustrate that the difference in precision between equations correct for time starting at 0, and for time starting at $-\infty$ can in some cases make a big difference.

We will consider the example system in Section 2.4. Let $\beta_T^l(\Delta) = \beta_T^u(\Delta) = \Delta$, and let an event arrive every n th time unit with no jitter, i.e.,

$$\alpha_T^l(\Delta) = \lfloor \frac{\Delta}{n} \rfloor \quad \text{and} \quad \alpha_T^u(\Delta) = \lceil \frac{\Delta}{n} \rceil$$

We assume that $n \geq 2$. If $n = 2$, the system is fully loaded, and if $n > 2$, the system is (more or less) underloaded. We know that the real system behavior will quickly stabilize to a situation where each event is processed first by $T1$, then by $T2$, so that end-to-end latency is 2.

As a measure of the ‘‘precision’’ of fixpoints, let us focus on two quantities characterized by specifications of events from $T1$ to $T2$, i.e., arrival curves $[\alpha^l, \alpha^u]$:

- *maxburst*, which is the maximum possible number of consecutive events that are allowed to be emitted with exactly one time unit distance, i.e., without any pause in the processing activity of $T1$.

The quantity *maxburst* can be obtained as

$$\sup \{ \Delta \in \mathbb{R} : \alpha^u(\Delta) \geq \Delta \} .$$

- *maxgap*, which is the maximum time interval during which no processing activity is required from $T2$. In RTC which starts at time 0, this quantity is represented as $\sup \{ \Delta \in \mathbb{R} : \alpha^l(\Delta) = 0 \}$, but with unbounded past, the quantity is

$$\sup \{ \Delta \in \mathbb{R} : \alpha^l(\Delta + 1) = 0 \} ,$$

since there is always a previous event, which makes $T2$ occupied for one time unit.

Let us now calculate lower bounds on *maxburst* and *maxgap* for a given value of n . We first consider the case where time starts at time 0. We obtain the following relations between *maxburst* and *maxgap*.

- $\text{maxgap} \geq \text{maxburst} + 1$. This is since $T2$ can be continuously busy for *maxburst* time units, meaning that $T1$ initially can be without processing resources for *maxburst* time units, implying that the first event is emitted by $T1$ after *maxburst* + 1 time units. In the case where time starts at $-\infty$, this relation should rather be $\text{maxgap} \geq \text{maxburst}$.

- *maxburst* is at least the largest value k that satisfies

$$\begin{aligned} k &\leq \text{maxgap} \\ n(k-1) &\leq \text{maxburst} + k - 1 \\ n(k-1) &\leq \text{maxburst} + \text{maxgap} - 1 \end{aligned} \quad (6)$$

To see this, note that $T1$ can be without processing resources for *maxburst* time units, thereafter it can obtain *maxgap* consecutive time units of processing resource. To emit k consecutive events, $T1$ needs to have first a queueing phase, which can have length *maxburst*, followed by k units of processing time.

Thus, we need to have

- $k \leq \text{maxgap}$, to process the k events,
- that the k th event arrives before the $k - 1$ first events have been processed; the k th event arrives at time $n(k-1)$ after start of the queueing phase, and the $k-1$ first events have arrived *maxburst* + $k - 1$ units after start of the queueing phase,
- that the k th event, which can be output at time $n(k-1) + 1$ after queueing start, does so before end of processing phase, i.e., before *maxburst* + *maxgap*.

Let us now consider the case $n = 2$. Then the largest values of *maxgap* and *maxburst* converge towards ∞ . To see this, assume that there are largest values b, g for them. Then $g \leq b + 1$ from first relation. thereafter, we also see that the second relation allows $k = b + 1$. For the case $n \geq 2$, the above equations give no such growth.

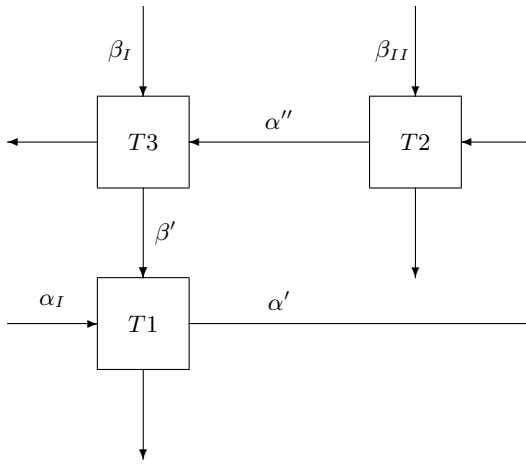
We thereafter consider the case where time starts at $-\infty$. In this case, the optimal fixpoint of the RTC equations will be a tightest characterization of actual system behavior, i.e., the fixpoint is as precise as possible. We have confirmed this by calculating the fixpoint in our MPA tool. In particular, we do not obtain the same growth in *maxburst* and *maxgap* as in the case where time starts at 0, since the equations that mutually bound b and g in terms of each other are slightly different (see above).

This example illustrates a system, where under full load we get perfect precision if we use the RTC equations that are correct for unbounded past, but no precision at all if we use RTC equations that are correct from time 0.

6. EXPERIMENTAL ILLUSTRATION

In this section, we show with a concrete example how the fixpoint iteration described in the previous sections succeeds in the analysis of a distributed system with a cyclic dependency.

Consider the system depicted in the figure below. It consists of a sequence of three tasks $T1$, $T2$ and $T3$ that process a periodic event stream α_T . The system has two computation resources with availability $\beta_T(\Delta) = \beta_{TT}(\Delta) = \Delta$. The first resource is shared by $T1$ and $T3$ according to a fixed priority scheduling policy with $T3$ having higher priority than $T1$. The system contains a cyclic dependency since $T1$ indirectly triggers $T3$ while $T3$ preempts $T1$.



We assume that the input event stream α_I is strictly periodic with a period P of 10 time units, i.e.

$$\alpha_I^l(\Delta) = \lfloor \frac{\Delta}{10} \rfloor \quad \text{and} \quad \alpha_I^u(\Delta) = \lceil \frac{\Delta}{10} \rceil$$

Further we assume that T1, T2 and T3 have constant processing times of 4, 7 and 5 time units, respectively. We want to compute tight bounds for the event streams denoted with α' , α'' and for the service stream denoted with β' .

In order to find an appropriate initial value for the fixpoint iteration we first simulate the system execution. For this simulation we use the PESIMDES simulation environment³. Since the system architecture is simple, the simulation could also be performed by hand without much effort. We observe that after an initial transitory phase, the task T1 produces outputs events for the task T2 with a recurring timing pattern. In particular it produces events with consecutive distances of 4 time units, 16 time units, 4 time units, 16 time units etc. This makes it easy to characterize the behavior σ of the event stream T1-T2 and find the tightest specification Σ_σ of this behavior in terms of upper and lower arrival curves. The corresponding arrival curves are depicted in Figure 1.

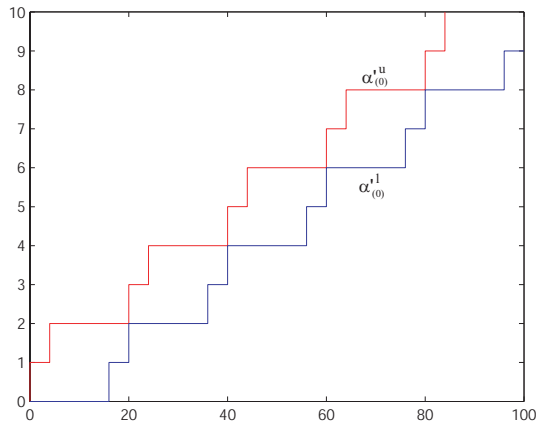


Figure 1: Tightest specification of the simulated trace T1-T2

We then use this arrival curves as initial values for the fixpoint iteration. In particular we model the tasks T1, T2

³available at <http://www.mpa.ethz.ch/PESIMDES/>

and T3 with three Greedy Processing Components GPC1, GPC2 and GPC3 and repeatedly compute their outgoing arrival and service curves in the following order: GPC2, GPC3, GPC1. After 4 iterations we reach a fixpoint, i.e. all the arrival curves and service curves in the performance model are stable. The obtained characterizations of α' , α'' and β' are depicted in the Figures 2, 3 and 4, respectively.

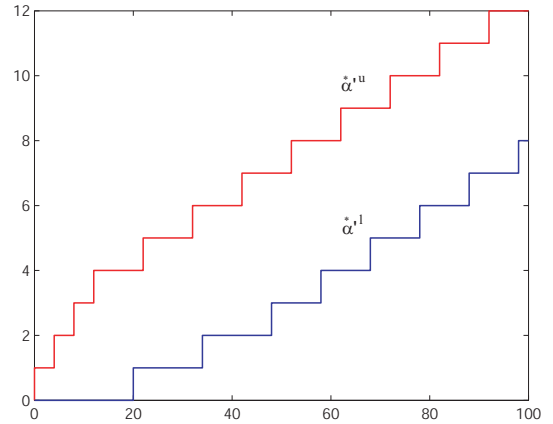


Figure 2: Characterization of the event stream T1-T2

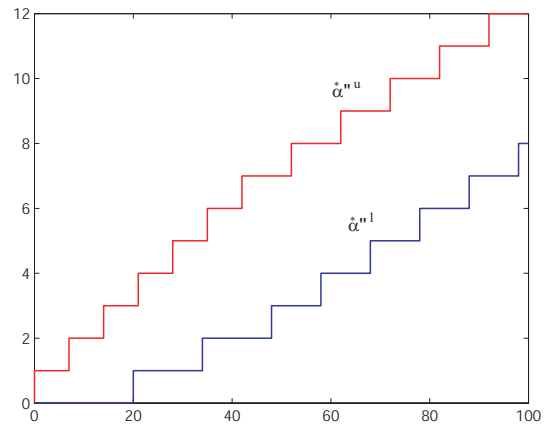


Figure 3: Characterization of the event stream T2-T3

7. CONCLUSION

We have performed a detailed study of correctness and preciseness of fixpoints obtained by solving the RTC equations for systems with cyclic dependencies. This has been performed by developing operational semantics. Under mild assumptions (“no zero-delay cycles”), we have proven that any satisfiable fixpoint correctly characterizes all allowed system behaviors. The results indicate that the recommended way to compute fixpoints by iteration is to start from a strong initial approximation, which is included in the sought optimal fixpoint and thereafter iterate towards a fixpoint, obtaining successively weaker and weaker approximations as the iteration progresses.

A major problem is to find a good initial approximation for the iteration. We consider two ways to find this approx-

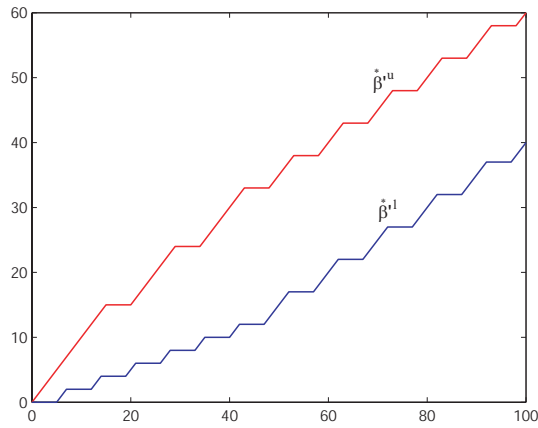


Figure 4: Characterization of the resource stream T3-T1

imation: one is to construct a simple periodic simulation of the system, from which the initial approximation is derived, another is to calculate the long-term rates of input streams, use them to calculate stable long-term rates at internal channels, and thereafter extract a strongest initial approximation.

We considered the precision obtainable by the optimal fix-point. One example illustrated that in boundary cases (full load), small variations in the setup can have large effects on precision. An experiment indicated an often typical effect: that the jitter is often overapproximated to some extent by the optimal fixpoint: this is an inherent consequence of the fact that RTC does not model phase correlations between different components. Future work includes to understand better how the obtained precision depends on parameters in the input specification.

Let us make an additional comparison with the work by Schiøler et al. [5]. An interesting idea, originated by them is to find long-term rates of streams in order to find initial specifications for the fixpoint iteration. They do not characterize when such long-term rates exist. In our paper, we have proven that such long-term rates always exist whenever a specification is satisfiable.

Appendix: Min-Max Algebra

The min-plus convolution \otimes and min-plus deconvolution \oslash of f and g are defined as:

$$(f \otimes g)(\Delta) = \inf_{0 \leq \lambda \leq \Delta} \{f(\Delta - \lambda) + g(\lambda)\}$$

$$(f \oslash g)(\Delta) = \sup_{\lambda \geq 0} \{f(\Delta + \lambda) - g(\lambda)\}$$

A curve f is *sub-additive*, if

$$f(a) + f(b) \geq f(a + b) \quad \forall a, b \geq 0$$

and *super-additive*, if

$$f(a) + f(b) \leq f(a + b) \quad \forall a, b \geq 0$$

References

[1] M. Fekete. Über die Verteilung der Wurzeln by gewissen algebraischen Gleichungen mit ganzzahligen Koeffizienten. *Mathematische Zeitschrift*, 17:228–249, 1923.

[2] M. Jersak, K. Richter, and R. Ernst. Performance analysis for complex embedded applications. *Int. J. of Embedded Systems*, 1(1/2):33–49, 2005.

[3] G. Kahn. The semantics of a simple language for parallel programming. In *IFIP 74*, pages 471–475. North-Holland, 1974.

[4] J. Y. Le Boudec and P. Thiran. *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet*. Springer-Verlag New York, Inc., 2001.

[5] H. Schiøler, J. Jessen, J. Dalsgaard, and K. Larsen. Network calculus for real time analysis of embedded systems with cyclic task dependencies. In G. Hu, editor, *Proc. 20th International Conference on Computers and Their Applications, CATA 2005, March 16-18, 2005, Louisiana*, pages 326–332. ISCA, 2005.

[6] E. Wandeler. *Modular Performance Analysis and Interface-Based Design for Embedded RealTime Systems*. PhD thesis, ETH Zürich, 2006.

[7] E. Wandeler, A. Maxiaguine, and L. Thiele. Quantitative characterization of event streams in analysis of hard real-time applications. In *IEEE Real-Time and Embedded Technology and Applications Symposium*, pages 450–461, 2004.