Cyclic Dependencies in Modular Performance Analysis [*]

Bengt Jonsson
Dept. Information Technology
Uppsala University
Sweden
bengt@it.uu.se

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

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

Wang Yi
Dept. Information Technology
Uppsala University
Sweden
yi@it.uu.se

ABSTRACT

The Modular Performance Analysis based on Real-Time Calculus (MPA-RTC), developed by Thiele et al., is an abstraction for the analysis of component-based real-time systems. 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.

Categories and Subject Descriptors

C.3 [**Computer Systems Organization**]: Special-Purpose and Application-Based Systems—*Real-time and Embedded Systems*; C.4 [**Computer Systems Organization**]: Performance of Systems—*Modeling techniques*

General Terms

Performance, Design, Theory

Keywords

Performance Analysis, Real-Time Calculus, Fixpoint Iteration

[*]Work supported in part by the ARTIST2 Network of Excellence and by the projects COMBEST and CREDO.

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 [9] 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 computation, 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 interaction.

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. The work does not provide results on the dependence of the fixpoint on initial conditions.

The general problem of analyzing cycles in the RTC has also been considered by Schiøler et al. [5]. Their treatment assumes that the time has an initial point 0, and hence must use weaker equations than usually considered in the RTC (such as in [7]), in order to establish correct results. They show that, under some assumptions, an optimal fixpoint can be obtained by iteration from an initial approximation of long-term rates, which can be derived analytically (see Section 4.1). They do not provide an explicitly defined operational model of system behaviors. Their statement about correctness of fixpoints appears to rely on unstated assumptions about causality or absence of zero-delay cycles. In our paper, we give an explicitly defined operational system model, and state explicit conditions under which fixpoints are correct, also for the case where time goes unboundedly into the past. Our results are stated in a general setting of any specification formalism, in which system components are viewed 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 [11]) 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. The MPA-RTC abstraction has shown to provide valuable analysis results in several industrial case studies [6, 10].

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 $R[s,t)$ for $s \leq t$ denotes the sum of events that arrive in the time interval $[s,t)$ (including s but not t), with $R[s,s) = 0$. If the time domain has an initial point 0, then of course $0 \leq s$.

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 pair 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

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

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 $[s,t)$ with $C[s,s) = 0$. Continuing the analogy, the

Real-Time Calculus abstracts from concrete service functions C and uses a pair 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

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

2.3 Component Model

Components are the basic building blocks of a system. They are the implementation of tasks that process event streams and run on shared resources, 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. We assume that component behavior is deterministic. 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 Figure 1(a).

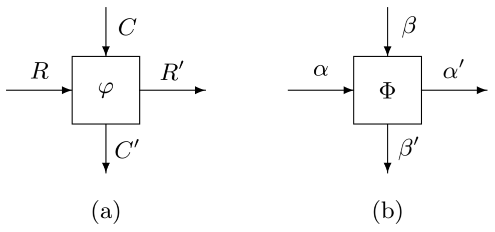

Figure 1: Concrete and abstract component

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 Figure 1(b). In the figure, we use a function Φ to denote the transformation which represents the abstract component. This function provides, for each input arrival and service curves α, β, the 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 a GPC component, the following internal relations between the incoming arrival curves $[\alpha^l, \alpha^u]$ and service curves $[\beta^l, \beta^u]$ and the outgoing arrival curves $[\alpha^{l'}, \alpha^{u'}]$ and service curves $[\beta^{l'}, \beta^{u'}]$ are stated and proven in [8][1]:

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

An important observation here is that these relations are correct for RTC with unbounded past (under the assumption of bounded buffer occupancy), but they are not correct for RTC with an initial time point. In particular, the equations for β'^u and α'^l are too tight. In this case, the relations derived in [7] must be used: $\alpha'^l = \alpha^l \otimes \beta^l$ and $\beta'^u(\Delta) = \sup_{0\leq\lambda\leq\Delta}\{\beta^u(\lambda) - \alpha^l(\lambda)\}$.

2.4 System Performance Model

In order to analyze the performance of a distributed system, we need to build a system performance model. This model has to reflect the flow of data in the system and also needs to represent its hardware architecture. In particular, it has to model the mapping of tasks to computation or communication resources, as well as the scheduling policies adopted on shared resources.

In the MPA-RTC framework such a performance model is constructed by composing abstract components to a network. The performance model depicted in Figure 2 shows a simple example of component composition. It represents a system consisting of two tasks $T1$ and $T2$ which serially process an incoming event stream, and share a processing resource according to a preemptive fixed priority policy, with task $T2$ having higher priority than task $T1$. Task $T1$ is triggered by input events from the environment. It produces output events which trigger task $T2$. Task $T2$ outputs events to the environment. In the performance model we represent the two tasks with two abstract Greedy Processing Components. We reflect the dataflow described above by connecting the abstract event stream output of component $T1$ with the abstract event stream input of component $T2$. Similarly, we connect the abstract resource stream output of component $T2$ with the abstract resource stream input of component $T1$ in order to model the scheduling policy for the shared processing resource; task $T2$ has full access to the processor, while task $T1$ receives only the processing resources left over by task $T2$.

In general, scheduling policies for shared resources are modeled by the way the abstract resources β are distributed among the different abstract components. For some scheduling policies, such as EDF or TDMA abstract components with appropriate transfer functions have been introduced.

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

2.5 A Simple Example

We use the system represented in Figure 2 as simple motivating example for our work, since it contains a cyclic dependency which inhibits the modular performance analysis.

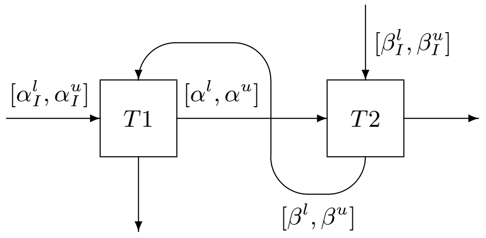

Figure 2: Performance model for example system

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 $[\alpha_I^l, \alpha_I^u]$, and that the incoming stream of resources to $T2$ is constrained by the service curves $[\beta_I^l, \beta_I^u]$. We do not consider any overhead for the context switching between the two tasks.

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

$$\begin{array}{lcl} \alpha^u & = & \min\left\{(\alpha_I^u \otimes \beta^u) \oslash \beta^l, \beta^u\right\} \\ \alpha^l & = & \min\left\{(\alpha_I^l \oslash \beta^u) \otimes \beta^l, \beta^l\right\} \\ \beta^u(\Delta) & = & \max\{\inf_{\Delta \leq \lambda}\{\beta_I^u(\lambda) - \alpha^l(\lambda)\}, 0\} \\ \beta^l(\Delta) & = & \sup_{0 \leq \lambda \leq \Delta}\{\beta_I^l(\lambda) - \alpha^u(\lambda)\} \end{array} \tag{4}$$

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 appoximation, i.e. by starting from some tuple Σ^0, and computing the sequence $\Sigma^0, \Sigma^1, \Sigma^2, \ldots$ 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 $\Sigma^0, \Sigma^1, \Sigma^2, \ldots$ 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 $\beta_I^l(\Delta) = \beta_I^u(\Delta) = \Delta$, i.e., giving $T2$ full access to the resource, and let an event arrive every second time unit, i.e.,

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

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

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

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

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

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 [2]. 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 $\mathbb{R}$ of all real numbers.

Streams. We assume that the behavior of a system or a component is observed at a set of *streams*, which are the observable connections between components and between components and the environment. A *trace* on a set V of streams is a function $\sigma : V \mapsto ((\mathbb{R} \times \mathbb{R}) \mapsto \mathbb{R}^{\geq 0})$, which for each stream $v \in V$ provides a function $\sigma(v)$ from time-intervals to observations. 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 $[s,t)$. We assume $\sigma(v)(s,t) + \sigma(v)(t,u) = \sigma(v)(s,u)$ for $s \leq t \leq u$. If time starts at 0, we assume $0 \leq s$ of course. We let $Tr(V)$ denote the set of traces on a set V of streams. The restriction of a trace σ on V to a subset $V' \subseteq V$ 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. We let its behavior be characterized by a *behavior mapping* $\varphi : Tr(V_I) \mapsto Tr(V_O)$, which maps any input trace σ_I on the input streams V_I to an output trace $\varphi(\sigma_I)$ on the output streams V_O.

System Behavior. A system of components 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 output streams go to other components (they can be thought of as internal), and some to the environment. By putting together all component behavior transformers, we get a *system behavior transformer* $\psi : \mathit{Tr}(V_I \cup V_O) \mapsto \mathit{Tr}(V_I \cup V_O)$ which maps any trace on the streams of the system to another trace on system streams. The mapping ψ will preserve the traces on input streams, and will generate traces 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 trace σ_I on input streams V_I induces a unique trace on the output streams. To establish this formally, some form of causality between inputs and outputs of the system must be assumed. We will define a property, called *simulatability*, which roughly means that the system has no zero-delay cycles. We first consider the case when time starts at 0, thereafter indicate how to adapt the results to an unbounded past.

Let a *time vector* on a set V of streams be a mapping $\tau : V \mapsto \mathbb{R}$ from streams in V to time values. For a value $t \in \mathbb{R}$, let $\bar{t}$ denote the time vector which maps all streams in its domain to t. For two time vectors τ, τ' on V, let $\tau \leq \tau'$ denote that $\tau(v) \leq \tau'(v)$ for all streams $v \in V$. For a time vector τ on V and set of streams V' with $V \subseteq V'$, we say that two traces σ and σ' on V' *agree upto* τ, denoted $\sigma \simeq_\tau \sigma'$, if $\sigma(v)(s,t) = \sigma'(v)(s,t)$ whenever $s \leq t \leq \tau(v)$ for all streams $v \in V$ and $\sigma(v) = \sigma'(v)$ for all streams $v \in V' \backslash V$. Intuitively, time vectors will be used to denote how much of a system trace has been constructed in a simulation. A time vector need not 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.

Definition 1. A system, represented by behavior transformer ψ, is *simulatable* if for each input trace $\sigma_I \in \mathit{Tr}(V_I)$ there is an increasing sequence $\tau_0, \tau_1, \ldots$ of time vectors on V_O, called the *simulation sequence for* σ_I, with $\tau_0 = \overline{0}$ and $\lim_{i \to \infty} \tau_i(v) = \infty$ for all $v \in V_O$, such that for all traces $\sigma, \sigma' \in \mathit{Tr}(V_I \cup V_O)$ with $\sigma|_{V_I} = \sigma'|_{V_I} = \sigma_I$ we have

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

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 trace can be obtained in ω steps.

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

Let us consider how to adapt this to the situation when time goes unboundedly into the past. Then there is no initial time point for defining traces. For a time vector τ, say that σ^0 is a *possible system trace up to* τ if $\sigma^0 \simeq_\tau \psi(\sigma^0)$. Let τ be an arbitrary real-valued time vector on V_O. We say that the system is *simulatable from* τ if whenever σ^0 is a possible system trace up to τ with $\sigma^0|_{V_I} = \sigma_I$, there is an increasing sequence $\tau_0, \tau_1, \ldots$ of time vectors on V_O, starting with $\tau_0 = \tau$, called the *simulation sequence for* σ^0 *from* τ, with $\lim_{i \to \infty} \tau_i(v) = \infty$ for all $v \in V_O$, such that for all traces $\sigma, \sigma' \in \mathit{Tr}(V_I \cup V_O)$ 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.

If the system is now simulatable from τ, we can construct an actual system trace $\psi^\omega(\sigma^0)$ in the same way as above, starting from σ^0. We say that $\psi^\omega(\sigma^0)$ is the (uniquely defined) actual system trace on input σ_I, which agrees with σ^0 upto τ.

Component Specifications. Let us now consider specifications of traces and behaviors. Abstractly, a specification Σ on a set V of streams is a mapping $V \mapsto 2^{((\mathbb{R} \times \mathbb{R}) \mapsto \mathbb{R}^{\geq 0})}$, which maps each stream $v \in V$ to a set $\Sigma(v)$ of traces 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 $\mathit{Spec}(V)$ 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* $\Phi : \mathit{Spec}(V_I) \mapsto \mathit{Spec}(V_O)$ from specifications on the set V_I of input streams to specifications on the set V_O of output streams. This mapping should be *correct wrp. to* φ, i.e., have the property that whenever $\sigma_I(v_i) \models \Sigma_I(v_i)$ 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* $\Psi : \mathit{Spec}(V_I \cup V_O) \mapsto \mathit{Spec}(V_I \cup V_O)$. From correctness of component specification mappings, it follows that Ψ is correct wrp. to the system behavior transformer ψ, i.e., whenever $\sigma \models \Sigma$, then $\varphi(\sigma) \models \Phi(\Sigma)$ for $\sigma \in \mathit{Tr}(V_I \cup V_O)$ and $\Sigma \in \mathit{Spec}(V_I \cup V_O)$. The question we want to study is the following: Assume Ψ, and a specification Σ_I on V_I, let σ_I be any trace on V_I with $\sigma_I \models \Sigma_I$. Will the actual trace $\psi^\omega(\sigma_I)$ of the system on input σ_I satisfy the "limit" of a sequence $\Sigma^0, \Psi(\Sigma^0), \Psi(\Psi(\Sigma^0)), \ldots$? 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 trace σ *satisfies a specification* Σ *upto* τ, denoted $\sigma \models_{\leq \tau} \Sigma$ if there is a trace σ' with $\sigma \simeq_\tau \sigma'$ such that $\sigma' \models \Sigma$.

In the rest of this section, we assume a simulatable system with input streams V_I and output streams V_O, 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 on V_I and a trace σ_I on V_I with

$\sigma_I \models \Sigma_I$. Let $\tau_0, \tau_1, \tau_2, \ldots$ be the simulation sequence for σ_I, and let $\psi^\omega(\sigma_I)$ be the actual system trace on input σ_I.

We shall assume that for an actual system trace 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. In MPA-RTC, this strongest specification can be obtained from Equations (1) and (2).

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 $\tau_0, \tau_1, \tau_2, \ldots$ be the simulation sequence for σ_I. Let $\psi^\omega(\sigma_I)$ be the actual system trace given σ_I, obtained as the limit of the sequence $\sigma^0, \sigma^1, \sigma^2, \ldots$, where σ^0 is any trace 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 $\sigma^0 \models_{\leq \overline{0}} \Sigma^0$ 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 trace 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. □

Convergence. Theorem 1 does not say anything about convergence of the sequence $\Sigma^0, \Psi(\Sigma^0), \Psi^2(\Sigma^0), \ldots$. Let us formalize 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 $Spec(V)$ of specifications over V, by defining $\Sigma \sqsubseteq \Sigma'$ iff $\sigma \models \Sigma$ implies $\sigma \models \Sigma'$ for any $\sigma \in Tr(V)$. We shall assume that $Spec(V)$ with the partial order $\sqsubseteq$ constitutes a cpo, i.e., that any chain $\Sigma^0 \sqsubseteq \Sigma^1 \sqsubseteq \Sigma^2 \sqsubseteq \cdots$ 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 trace $\sigma \in Tr(V)$ 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 $\Sigma^0 \sqsubseteq \Sigma^1 \sqsubseteq \cdots$.

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 trace σ with $\sigma|_{V_I} \models \Sigma_I$, Ψ has a unique smallest fixpoint Σ^* which is satisfied by all such traces. Furthermore Σ^* can be obtained as the limit of the sequence $\Sigma^0, \Psi(\Sigma^0), \Psi^2(\Sigma^0), \ldots$ 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 trace 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 trace 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 $\Sigma_\sigma, \Psi(\Sigma_\sigma), \Psi^2(\Sigma_\sigma), \ldots$ converges to a fixpoint Σ^*. By Theorem 1, and the assumption that all properties are safety properties, $\sigma'' \models \Sigma^*$ for any trace σ'' of the system such that $\sigma''|_{V_I} \models \Sigma_I$. Since σ was an arbitrary initial trace, we get the same fixpoint Σ^* if we repeat the above procedure with any other trace σ' with $\sigma'|_{V_I} \models \Sigma_I$.

We have proven the theorem for initial approximations of form Σ_σ for some system trace. 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 $\Sigma^0, \Psi(\Sigma^0), \Psi^2(\Sigma^0), \ldots$ converges to Σ^*. □

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 trace σ with $\sigma|_{V_I} \models \Sigma_I$, there is a unique least fixpoint which is satisfied by all such traces. The theorem furthermore states that this fixpoint can be obtained by iteration from an initial approximation, which characterizes one possible system trace. Thus, if we can construct one system trace, we can get a specification of all possible system traces by iterating from a strongest specification of this trace 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 possible system trace up to τ with $\sigma^0|_{V_I} \models \Sigma_I$, and let $\tau_0, \tau_1, \tau_2, \ldots$ be the simulation sequence for σ^0 from τ. Define $Cont(\sigma^0, \tau)$ as the set of traces σ such that $\sigma \simeq_\tau \sigma^0$. 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. □

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 trace in $Cont(\sigma^0, \tau)$, the transformer Ψ has a unique smallest fixpoint Σ^* which is satisfied by all traces $\sigma \in Cont(\sigma^0, \tau)$ with $\sigma|_{V_I} \models \Sigma_I$. Furthermore Σ^* can be obtained as the limit of the sequence $\Sigma^0, \Psi(\Sigma^0), \ldots$ 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 trace in $Cont(\sigma^0, \tau)$, and satisfies $\Sigma^0 \sqsubseteq \Sigma^*$.*

Proof. Analogous to that for Theorem 2. □

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 trace 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 trace on input streams. $\square$

Concerning specifications, we should check that, for a set V of streams the set of specifications $Spec(V)$ on V forms a cpo. Recall that a specification $\Sigma \in Spec(V)$ maps each event stream $v \in V$ to a pair of arrival curves $[\alpha^l, \alpha^u]$, and maps each resource stream $v \in V$ to a pair of service curves $[\beta^l, \beta^u]$. In order to make $Spec(V)$ 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 $(Spec(V), \sqsubseteq)$ forms a cpo. It is also easy to see that all specifications in $Spec(V)$ 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 trace 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 0, we can even prove a slightly stronger version, namely the following.

THEOREM 5. *Assume that Σ_I allows at least one system trace 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 traces σ with $\sigma|_{V_I} \models \Sigma_I$. Furthermore Σ^* can be obtained as the limit of the sequence $\Sigma^0, \Psi(\Sigma^0), \Psi^2(\Sigma^0), \ldots$ 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 needs not to be satisfied by any actual system trace. To prove this extension, let σ_p be an eventually periodic actual system trace. 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

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

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 trace 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 trace 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 satisfies $\sigma^0 \models_{\leq\tau} \Sigma$, there is a unique smallest fixpoint Σ^* of Ψ. This fixpoint Σ^* is satisfied by all system traces $\sigma \in Cont(\sigma^0, \tau)$ with $\sigma|_{V_I} \models \Sigma_I$. Furthermore Σ^* can be obtained as the limit of the sequence $\Sigma^0, \Psi(\Sigma^0), \ldots$ of approximations if Σ^0 is a specification which agrees with Σ_I on V_I, and satisfy $\sigma^0 \models_{\leq\tau} \Sigma$ 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 trace σ of the system, such that $\sigma|_{V_I} \models \Sigma_I$, and such that σ satisfies the optimal fixpoint of Ψ. Such a trace could be found, by constructing a simulation which is an actual system trace. The task is made easier by the observation that we need to find only one trace, and can choose one which is as regular as possible, e.g., as an infinitely repeating periodic trace.

2. Let Σ_σ be a tightest specification of the trace σ, 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 $\Sigma^0, \Psi(\Sigma^0), \ldots$. This sequence converges 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 weak initial approximation and iterating towards a stronger solution - will in many cases yield quite poor precision, as illustrated by the example at the end of Section 2.5.

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 2.5, 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 system trace. 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 $\lim_{t\to\infty} \frac{f(t)}{t}$ exists. So, define

$$\begin{array}{rclcrcl} A^l &=& \lim_{t\to\infty} \frac{\alpha_I^l(t)}{t} & \qquad & A^u &=& \lim_{t\to\infty} \frac{\alpha_I^u(t)}{t} \\ B^l &=& \lim_{t\to\infty} \frac{\beta_I^l(t)}{t} & & B^u &=& \lim_{t\to\infty} \frac{\beta_I^u(t)}{t} \end{array} \tag{5}$$

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. Continuing the particular example in Section 2.5, it follows that the specification allows traces 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

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

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 trace which does not generate overflow. Thus, Σ^0 satisfies the conditions 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 more complicated interconnection structure, the initial approximation can be obtained by solving a system of linear equations to get a long-term rate for each stream, as detailed by Schiøler et al. [5]. The essential principle is the same as we just described for this small system.

5. ON PRECISION OF FIXPOINTS

The results in Sections 3 and 4 show that fixpoints of equations in RTC are satisfied by system traces. 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.

Consider the example system in Section 2.5. Let $\beta_I^l(\Delta) = \beta_I^u(\Delta) = \Delta$, and let an event arrive every nth time unit with no jitter, i.e.,

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

[2]Schiøler 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

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 actual system trace 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 actitivity 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*.

- *maxgap* $\geq$ *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 *maxgap* $\geq$ *maxburst*.

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

  $$\begin{array}{rcl} k &\leq& \mathit{maxgap} \\ n(k-1) &\leq& \mathit{maxburst} + k - 1 \\ n(k-1) &\leq& \mathit{maxburst} + \mathit{maxgap} - 1 \end{array} \tag{6}$$

  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

  - $k \leq$ *maxgap*, to process the k events,
  - that the kth event arrives before the $k - 1$ first events have been processed; the kth even 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 kth 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 $n > 2$, 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 the tightest characterization of an actual system trace, 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 Figure 3. It consists of a sequence of three tasks T1, T2 and T3 that process a periodic event stream α_I. The system has two computation resources with availability $\beta_I(\Delta) = \beta_{II}(\Delta) = \Delta$. 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.

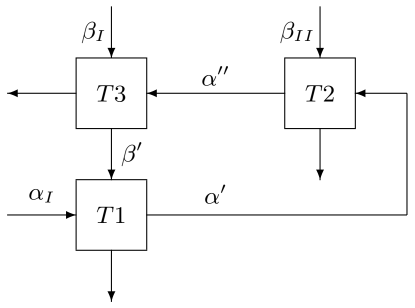

Figure 3: Performance model for experiment

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

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

Further we assume that T1, T2 and T3 have constant processing times of 4, 7 and 5 time units, respectively. We want to compute 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 simulaton environment[3]. 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 4.

[3]available at http://www.mpa.ethz.ch/PESIMDES/

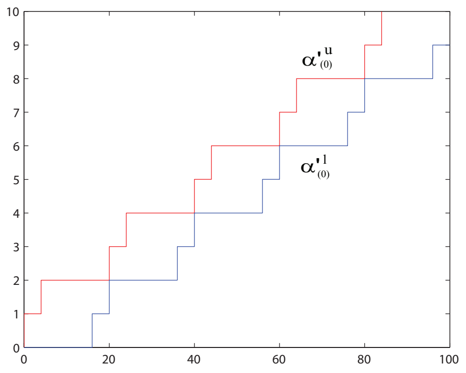

Figure 4: Tightest specification of the simulated trace T1-T2

We then use this arrival curves as initial values for the fixpoint iteration. In particular we model the tasks T1, T2 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 5, 6 and 7, respectively.

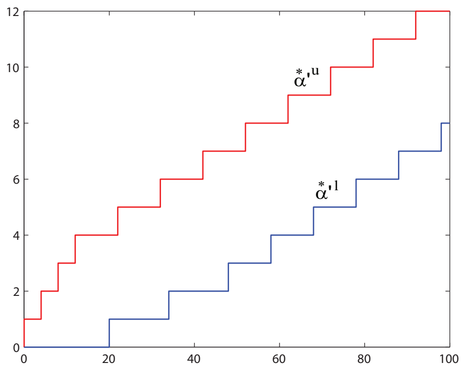

Figure 5: Bounds for the event stream T1-T2

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 an 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

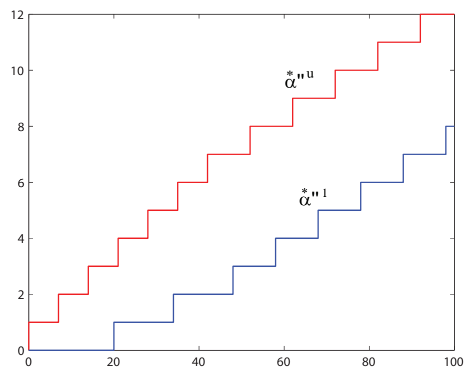

Figure 6: Bounds for the event stream T2-T3

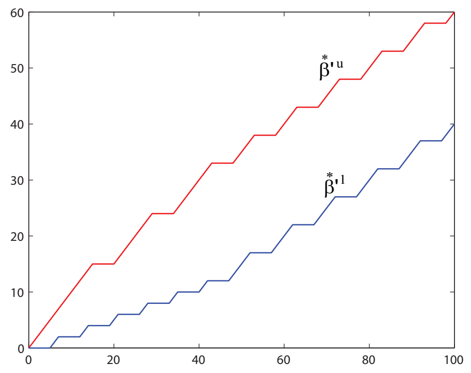

Figure 7: Bounds for the resource stream T3-T1

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 approximation: one is to construct a simple periodic simulation of the system, from which the initial approximation is derived, another (previously suggested by Schiøler et al. [5]) 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 fixpoint. 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.

Appendix: Min-Max Algebra

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

$$\begin{aligned}(f \otimes g)(\Delta) &= \inf_{0\le\lambda\le\Delta}\{f(\Delta-\lambda)+g(\lambda)\}\\ (f \oslash g)(\Delta) &= \sup_{\lambda\ge 0}\{f(\Delta+\lambda)-g(\lambda)\}\end{aligned}$$

A curve f is *sub-additive*, if

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

and *super-additive*, if

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

8. REFERENCES

[1] M. Fekete. Über die Verteilung der Wurzeln bei 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] L. Thiele, S. Chakraborty, M. Gries, and S. Künzli. Design space exploration of network processor architectures. In *Network Processor Design: Issues and Practices, Volume 1*, pages 55–89. 2002.

[7] L. Thiele, S. Chakraborty, M. Gries, and S. Künzli. A framework for evaluating design tradeoffs in packet processing architectures. In *39th Design Automation Conference (DAC 2002)*, pages 880–885, New Orleans LA, USA, June 2002. ACM Press.

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

[9] 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.

[10] E. Wandeler, L. Thiele, M. Verhoef, and P. Lieverse. System architecture evaluation using modular performance analysis - a case study. *Software Tools for Technology Transfer (STTT)*, 8(6):649–667, Oct. 2006.

[11] R. Yates. Networks of real-time processes. In Best, editor, *Proc. CONCUR '93, Theories of Concurrency: Unification and Extension*, volume 715 of *Lecture Notes in Computer Science*, pages 384–397. Springer Verlag, 1993.