Causality Problem in Real-Time Calculus

Karine Altisen · Matthieu Moy

Formal Methods for System Design, Springer, 2016

Abstract Real-Time Calculus (RTC) [37] is a framework to analyze heterogeneous, real-time systems that process event streams of data. The streams are characterized by pairs of curves, called arrival curves, that express upper and lower bounds on the number of events that may arrive over any specified time interval. A well-known limitation of RTC is that it cannot model systems with states and several works [1, 4, 5, 11, 18, 19, 22, 23, 30, 24, 31, 38] studied how to interface RTC curves with state-based models. Doing so, while trying, for example to generate a stream of events that satisfies some given pair of curves, we faced a causality problem [33]: it can be the case that, after generating a finite prefix of an event stream, the generator deadlocks, since no extension of the prefix can satisfy the curves afterwards.

This paper formally defines the problem; it states and proves algebraic results that characterize causal pairs of curves, *i.e.* curves for which the problem cannot occur. We consider the general case of infinite curve models, either discrete or continuous time and events. The paper provides an analysis on how causality issues appear when using arrival curves and how they could be handled. It also provides an overview of algorithms to compute causal curves in several models. These algorithms compute a canonical representation of a pair of curves, which is the best pair of curves among the curves equivalent to the ones they take as input.

1 Introduction

The increasing complexity of modern embedded systems makes their design more and more difficult. Modeling and analysis techniques have been developed and help taking or validating decisions about the conception of a system as early as possible in the design process. There exist many techniques among which we can distinguish two families. *Computational* approaches study fine-grain models of the system and represent its complete behavior. The validation of the system using such a model may involve simulation, testing and verification. As opposed, *analytical* techniques, such

Univ. Grenoble Alpes, VERIMAG, F-38000 Grenoble, France
CNRS, VERIMAG, F-38000 Grenoble, France
E-mail: `Matthieu.Moy@imag.fr`, `Karine.Altisen@imag.fr`

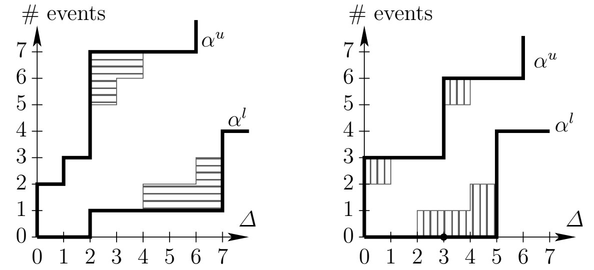
as Real-Time Scheduling (founded with [27]) and Real-Time Calculus [37], use purely analytical models, based on mathematical equations that can be solved efficiently. They usually compute quantitative information, such as worst case performances.

Both families of approaches have their advantages and drawbacks. Simulating precisely an embedded system gives very precise results, but only for one execution, and one instance of a system. Analytical approaches, on the other hand, provide, *e.g.*, worst case execution times, but only for cases that the theory can encompass. For example, Real-Time Calculus cannot handle the notion of state in the modeling of a system. Several studies try to combine the approaches to take the best of both [1, 4, 5, 11, 18, 19, 22, 23, 30, 24, 31, 38, 36]. The work we present in this paper fully takes its root and motivation in one of those studies, while working on the combination of Real-Time Calculus, state-based models and abstract interpretation, using synchronous languages [1].

Real-Time Calculus (RTC) [37] is a framework to model and analyze heterogeneous system in a compositional manner. It relies on the modeling of timing properties of event streams with curves called *arrival curves* (and service curves, which count available resources instead of events in a similar fashion). A component is described with curves for its input streams and available resources and some other curves for the outputs. Atomic RTC components, such as greedy processing component (GPC), are usually defined as a set of equations that relates their input and output streams. Now, given input arrival and service curves, RTC computes an abstraction of the behavior of the component expressed as output arrival and service curves. Then, several components can be composed in order to model a complex RTC system (see *e.g.*, [26]). For instance, inputs of the first component model constraints from the environment; the output curves of the first component can then be used as input for the next component, and so on. The analysis of the complete system is static, *i.e.*, does not need to execute the system.

Arrival curves are functions which use relative time and constrain the number of events that can occur time intervals. Precisely, the pair of arrival curves (α^u, α^l) *explicitly* defines the lower $\alpha^l(\Delta)$ and upper $\alpha^u(\Delta)$ bounds on the number of events that can occur in any sliding window of time of length Δ. For example (see Figure 1), $\alpha^u(2) = 3$ means that at most 3 events can occur during *any* time interval of length 2. Arrival curves may also contain *implicit constraints* which can be inferred from explicit ones. This paper studies those implicit constraints and provides algorithms to make them explicit.

Motivation. Implicit constraints cause problems in several contexts. For simulation purpose [21], it is typical to produce a stream of events that satisfies some given arrival curves using a *generator of events*: such generators produces streams of events which satisfy the curves. There are multiple ways to write such generators [21, 4, 1, 18, 38] but many faced the problem of implicit constraints. For example, let us consider a straightforward generator in discrete time. At each point in time, this generator computes the lower and upper bounds on the number of events which are allowed to be emitted, based on the events that were previously emitted. It then emits an number of events picked at random between those bounds. Imagine, now, using the curves in Figure 1.(b), that the generator has emitted no event during the first four time units. Then, after five time units, the generator is in a deadlock state, since it has to emit at least 4 but at most 3 events. In other words, it may happen, due to implicit constraints, that some upper bound becomes strictly lower than the



(a) Unreachable Regions (b) Forbidden regions

Fig. 1 Implicit and explicit constraints on arrival curves

corresponding lower bound, leading the generator to deadlock. Curves exhibiting this kind of problems are said to be non-*causal*.

Another case where implicit constraints are problematic is the case of *formal verification* of a system with inputs and outputs characterized by arrival curves. Such techniques usually handle stateful systems and analyze them using tools such as model-checking for timed automata [11, 22, 23, 24, 18, 38], model-checking for event count automata [31], abstract interpretation and bounded model-checking of synchronous data-flow programs [1]. In these works, one may want to prove a property like "If the input complies with the pair of arrival curves α_I, then the output satisfies the pair of arrival curves α_O". But verification tools based on reachability analysis (see *e.g.* [16]) usually allow the expression of "If the input complies with α_I *up to some time* t, then the output complies with α_O *up to time* t", only. Of course, those two properties are not equivalent and it can happen that the tools exhibit some *spurious counter-example*, namely, a counter-example which violates the latter property while the first one is satisfied. Precisely, this would be a finite counter-example which would violate α_O without violating α_I *up to some time* t, but which could not be extended into an infinite execution that still satisfies α_I. The problem, here, is the same as for generator: trying to extend the execution deadlocks due to contradictory constraints between lower and upper bounds of α_I; again the curve is said to be non-causal.

Back to verification tools that are used to analyse stateful systems specified with arrival curves, one may wonder, for each tool: "what is the behavior of the tool when used with non-causal curves?" and also "does the tool output non-causal curves?". Actually, except [1], the papers do not answer those questions. We will see that [22, 23, 24, 18, 38] can not create non-causal curves while [31] could at least in theory. Nevertheless, each of the tools would badly behave with non-causal input curves. Getting rid of this, inside those tools, may require, in general, heavyweight state exploration techniques. This paper gives a way to get rid of non-causal curves before using any verification tool.

Implicit constraints on arrival curves. We distinguish two kinds of implicit constraints that we call informally "unreachable regions" and "forbidden regions". The first one is a well-studied phenomenon within the Real-Time Calculus community [25] and the second, which may produce deadlocks in generators and spurious counter-examples in verification is the goal of this paper. Let us discover the two kinds of implicit constraints using a pair of arrival curves (α^u, α^l) (see Figure 1 for an example).

Firstly, by splitting some time interval into smaller ones, we can get additional constraints. As shown in Figure 1.(a), in an interval of size $\Delta = 6$, the curve says explicitly that the lower bound on the number of events is 1, but splitting this interval into three intervals of size 2, one can deduce a better bound, which is 3. Although the curve explicitly specifies the bounds $\alpha^l(6)$ and $\alpha^u(6)$ as 1 and 7, the number of events in a window of size 6 can actually never be equal to $\alpha^l(6) = 1$. In other words, the actual implicit lower bound is greater than $\alpha^l(6)$: this means that the curve is equivalent to a tighter curve. A well-known result [25] is that the upper (resp. lower) curve does not have this kind of implicit constraints if it is sub-additive (resp. super-additive). The transformation of an arbitrary curve into an equivalent sub-additive (resp. super-additive) curve making those constraints explicit is called *sub-additive closure* (resp. *super-additive closure*). In this paper, we call the region between the curves and its sub-additive (resp. super-additive) closure *unreachable regions* (since a, *e.g.*, generator can never visit such regions). Unreachable regions are due to constraints of a single curve on itself, and can be computed at some point by looking only at the past, *i.e.* smaller Δ.

The second case of implicit constraints can be found by looking at both curves toward the future. We use again Figure 1.(b) for an example of such a case: if no event occurs during a time window of size 3 (this is allowed since $\alpha^l(3) = 0$), then the upper curve forbids emitting more than 3 events in the next 2 units of time, while the lower curve forces to emit at least 4. It is therefore impossible to emit no event for 3 time units. We call the regions that contain such points *forbidden regions*. No execution can cross a forbidden region without getting locked some time later, due to some contradiction between lower and upper constraints. Borrowing the vocabulary used in [33], we call this kind of implicit constraints *causality constraints*. A pair of curves for which the beginning of any execution never prevents the execution from continuing is called *causal*. Intuitively, this is the same as having no forbidden region (but we will see that the relationship between absence of forbidden region and causality is only a one-way implication).

Surprisingly, this question has received very little attention and to the best of our knowledge, no transformation has been published before [2] to make these implicit constraints explicit. One may wonder if this is a “true” problem, *i.e.* if such non causal curves can be encountered in practice. Indeed, part of the answer is that they cannot come from measurements on a real system, since curves derived from execution or simulation of real systems are always well-formed (i.e. are sub-additive/super-additive and causal). The common practice is to use causal curves for the inputs of RTC models. As RTC computations preserve the causality of the curves, non-causal curves were not considered as a problem so far. This may explain why no studies have been published yet on the subject. Things are different when instead of using RTC operators, one uses other tools for deriving output arrival curves, given some input arrival curves. Those tools (*e.g.* model-checking of timed automata [15] on abstracted models, abstract interpretation of Lustre programs [16]) may compute non-causal arrival curves, even when the input is causal. Since the output of one analysis can be used as input for the next analysis, this non-causality can be problematic.

Additionally, non-causal curves contain implicit constraints that could be made explicit. If the output of a computation gives the curve in Figure 1, then making the implicit constraints explicit gives tighter bounds (for example, a tighter bound on the number of events in a window of size 4). We encountered the case, when merging the output of several computations for the same inputs [4] using different approximation methods. This provides several pairs of curves, each of them being a valid over-approximation of the expected result. The basic combination of these curves (point-wise minimum and maximum) can contain implicit constraints, and making them explicit gives more precise results from the same analysis.

Contributions. To solve these issues, this paper formally defines the causality problem and proposes several solutions.

- We give an algebraic characterization of a causal pair of arrival curves.
- Combining this property with existing ones, we give a definition of a *canonical representation* of a pair of arrival curves, which is causal and sub-additive/super-additive. We show that it is also the *tightest representation* of the original pair of curves.
- We propose a general transformation called *causality closure* which computes the causal representation of an arbitrary pair of arrival curves.

We provide an overview about causality issues that can appear when using arrival curves and how they could be handled. We also outline algorithms which compute causality closure, dedicated to several important classes of arrival curves.

The paper is an extended version of [2] and contains all the proofs of the results. The theory is developed for dense-time or discrete-time arrival curves on the one hand, discrete-event or fluid-event models on the other hand. The implementation part has been developed for discrete-time discrete-event models, since this was our context of use [1], but it could be adapted to other contexts. Furthermore, although all along the paper we talk about arrival curves, the reader should be convinced that results also apply to service curves in Real-Time Calculus, and strict service curves in Network Calculus [8].

Again, the results were published in short in [2], and they are starting to gain acceptance in the Real-Time Calculus community. Indeed, [10] considers the problem of conformance of a stream to an RTC specification, and relies on [2] to define whether a finite valid sequence can be extended into an infinite valid one (finite sequences that cannot be extended are called *locking sequences*). [12] proposes Finitary RTC to solve some performance issues with traditional RTC. The authors consider causality as a part of RTC Basics, assume the curves to be causal, and rely on it (using Theorem 4 of this paper) in proofs of their own theorems. [5] proposes an approach to generate random sequences of events based on RTC specifications and also refers to [2] for the definition of causality.

However, the problem is still not well understood. [10] detects *locking sequences* using a SAT solver and [23] proposes to detect deadlocks using model-checking on timed automata. Those methods are usually more expensive, algorithmically, than the analytical solution proposed in this paper. [5] defines an algorithm to turn RTC specifications into an automaton and prunes the states of this automaton, *a priori*, to avoid forbidden states. Again, applying the causality closure algorithm proposed in this paper would have been simpler and would have completely avoided the problem.

Now, new approaches based on [2], *e.g.* [12], rely on the validity of theorems they use. Note that up to now, proofs of these theorems have not been published while the proofs are non-trivial. The main goal of this paper is therefore to enhance the results of [2] providing complete proofs. We provide, in addition, new pedagogic material to help the reader to better understand our results, new experimental results and, also, new results like Theorem 10 which does not appear in previous work.

Roadmap. Section 2 defines *arrival curves* and some algebraic operators; the reader familiar with RTC may jump to Section 3, which defines *causality* and gives a *suffi-*

cient condition to guarantee that a pair of arrival curves is causal. Section 4 extends this result to a *characterization* (necessary and sufficient condition), valid for a wide class of curves. Section 5 defines *causality closure* which computes the *tightest causal representation* of a pair of arrival curves. Section 6 provides a discussion about *causality issues* and their solutions. Section 7 gives an overview on *how to compute causality closure* for many classes of curves used in RTC applications. Algorithms are evaluated experimentally in Section 8.

2 Arrival Curves

Usually in RTC, arrival curves are given by pairs and express upper and lower bounds on the number of events that can occur in any window of time. This section defines arrival curves. It first gives the context — the study allows either discrete or continuous time and discrete or fluid event counts — and recalls some basic notions of Min/Max-plus algebra that will be used later in the paper.

2.1 Basic Notions in Min-plus and Max-plus Algebra

Notations: $\mathcal{R}^+$ denotes the set of non-negative real numbers, $\overline{\mathcal{R}^+} \overset{\text{def}}{=} \mathcal{R}^+ \cup \{\infty\}$, $\mathcal{N}$ the set of natural numbers and $\overline{\mathcal{N}} \overset{\text{def}}{=} \mathcal{N} \cup \{\infty\}$.

To define arrival curves, we use functions that measure the number of events occurring before a given time. We assume either a discrete-event or a fluid-event model: the number of events may be discrete (*i.e.* in $\mathcal{N}$) or fluid (*i.e.* in $\mathcal{R}^+$); we give bounds in $\overline{\mathcal{N}}$ or $\overline{\mathcal{R}^+}$, ∞ being used to denote the absence of constraint on upper bounds. Also, we allow time to be either discrete or continuous. We denote by $\mathcal{T}$ the time and $\mathcal{E}$ the event count; $\mathcal{T}$ and $\mathcal{E}$ independently represent either $\mathcal{N}$ or $\mathcal{R}^+$ ($\overline{\mathcal{E}}$ is either $\overline{\mathcal{N}}$ or $\overline{\mathcal{R}^+}$). We consider functions from $\mathcal{T}$ to $\overline{\mathcal{E}}$ whatever be the value of $\mathcal{T}$ and $\mathcal{E}$.

Definition 1 Let f be such a function from $\mathcal{T}$ to $\overline{\mathcal{E}}$. f is *wide-sense increasing* iff(def) $\forall x, y \in \mathcal{T} \;.\; x \leq y \implies f(x) \leq f(y)$.

In the definition, $\leq$ is naturally extended to $\overline{\mathcal{E}}$. We denote by $\mathcal{F}$ the set of functions f such that f is a function from $\mathcal{T}$ to $\overline{\mathcal{E}}$, f is wide-sense increasing and $f(0) = 0$. $\mathcal{F}_{finite}$ represents the set of functions in $\mathcal{F}$ restricted to functions from $\mathcal{T}$ to $\mathcal{E}$. We use the usual point-wise order on $\mathcal{F}$:

Definition 2 Let $f, g \in \mathcal{F}$, $f \leq g \overset{\text{def}}{\iff} \forall x \in \mathcal{T} \;.\; f(x) \leq g(x)$.

We recall the usual operators $\otimes$, $\overline{\otimes}$, $\oslash$, $\overline{\oslash}$:

Definition 3 Let f, g be functions from $\mathcal{T}$ to $\overline{\mathcal{E}}$ and $x \in \mathcal{T}$,

$$(f \otimes g)(x) \overset{\text{def}}{=} \inf_{t \in [0,x]} \{f(x-t) + g(t)\} \qquad \text{((min, +) convolution)}$$

$$(f \,\overline{\otimes}\, g)(x) \overset{\text{def}}{=} \sup_{t \in [0,x]} \{f(x-t) + g(t)\} \qquad \text{((max, +) convolution)}$$

$$(f \oslash g)(x) \overset{\text{def}}{=} \sup_{t \geq 0} \{f(x+t) - g(t)\} \qquad \text{((min, +) deconvolution)}$$

$$(f \,\overline{\oslash}\, g)(x) \overset{\text{def}}{=} \inf_{t \geq 0} \{f(x+t) - g(t)\} \qquad \text{((max, +) deconvolution)}$$

Note that if $f, g \in \mathcal{F}$, $(f \otimes g)$ and $(f \overline{\otimes} g)$ are in $\mathcal{F}$, but $(f \oslash g)$ and $(f \overline{\oslash} g)$ may not (since, for instance, $(f \oslash g)(0)$ and $(f \overline{\oslash} g)(0)$ may not be equal to zero). The following lemma gives conditions for $(f \oslash g)$ and $(f \overline{\oslash} g)$ to be in $\mathcal{F}$.

Lemma 1 *Let f, g be two functions. If $f, g \in \mathcal{F}$, then*

1. *$f \otimes g \in \mathcal{F}$; $f \overline{\otimes} g \in \mathcal{F}$;*
2. *$f \oslash g$ and $f \overline{\oslash} g$ are wide-sense increasing;*
3. *$f \oslash g \in \mathcal{F} \iff f \leq g$*
4. *$f \overline{\oslash} g \in \mathcal{F} \iff g \leq f$*

Proof 1. $(f \otimes g)(0) = \inf_{0 \leq t \leq 0}\{f(0-t) + g(t)\} = f(0) + g(0) = 0$; let $x, y \in \mathcal{T}$ such that $x \leq y$, for all t: $f(x-t) + g(t) \leq f(y-t) + g(t)$ since f is wide-sense increasing, this implies that $f \otimes g(x) \leq f \otimes g(y)$. The proof is similar for $\overline{\otimes}$.
2. Let $x, y \in \mathcal{T}$ such that $x \leq y$, then for all $t \geq 0$: $f(x+t) - g(t) \leq f(y+t) - g(t)$ since f is wide-sense increasing and this implies that $f \oslash g(x) \leq f \oslash g(y)$. The proof is exactly the same to prove that $f \overline{\oslash} g(x) \leq f \overline{\oslash} g(y)$.
3. $f \oslash g(0) = \sup_{t \geq 0}\{f(t) - g(t)\}$. Note that $f \leq g \iff \sup_{t \geq 0}\{f(t) - g(t)\} \leq 0$. If $f \leq g$, as $f(0) = g(0) = 0$, we have $f \oslash g(0) = 0$. Conversely, if $f \oslash g(0) = 0$, this means that $\sup_{t \geq 0}\{f(t) - g(t)\} \leq 0$, hence $f \leq g$.
4. The proof for $\overline{\oslash}$ is the same as for $\oslash$.

□

We now give the formal definition of the intuitive notion of "unreachable regions" (see Figure 1). Curves have no "unreachable region" if they are sub-additive/super-additive:

Definition 4 Let $f \in \mathcal{F}$, f is *sub-additive* iff(def) $\forall s, t \in \mathcal{T} \,.\, f(t+s) \leq f(t) + f(s)$. Well-known results (see [25] for example) characterize sub-additivity:

1. $f \in \mathcal{F}$ and f sub-additive $\iff f \otimes f = f$
2. Let $f \in \mathcal{F}$. Among all the sub-additive functions $g \in \mathcal{F}$ that are smaller than f ($g \leq f$) there exists an upper bound called the *sub-additive closure* of f given by:

$$\overline{f} \overset{\text{def}}{=} \inf_{n \geq 1} \otimes^n f$$

where $\otimes^1 f = f, \otimes^{n+1} f = f \otimes (\otimes^n f)$.

Definition 5 Let $f \in \mathcal{F}$, f is *super-additive* iff(def) $\forall s, t \in \mathcal{T} \,.\, f(t+s) \geq f(t) + f(s)$. Again, well-known results (see [25] for example) characterize super-additivity:

1. $f \in \mathcal{F}$ and f super-additive $\iff f \overline{\otimes} f = f$
2. Let $f \in \mathcal{F}$. Among all the super-additive functions $g \in \mathcal{F}$ that are greater than f ($g \geq f$) there exists a lower bound called the *super-additive closure* of f given by:

$$\underline{f} \overset{\text{def}}{=} \sup_{n \geq 1} \overline{\otimes}^n f.$$

2.2 Arrival Curves

2.2.1 Definition of Arrival Curves

Arrival curves define lower and upper bounds on the number of events that can occur in a window of time. It defines a set of event streams that satisfy all the bounds: as usual, we represent such an event stream with a cumulative curve.

Definition 6 A *cumulative curve* of an event stream is a curve $R \in \mathcal{F}_{finite}$ such that $R(t)$ represents the (finite) number of events that occur in the interval of time $[0, t]$.

Definition 7 A *pair of arrival curves* is a pair of functions (α^u, α^l) in $\mathcal{F} \times \mathcal{F}_{finite}$, such that $\alpha^l \leq \alpha^u$.

Equivalently, a pair of arrival curves is a pair of positive wide-sense increasing functions (α^u, α^l) such that $\alpha^l(0) = \alpha^u(0) = 0$, $\alpha^l \leq \alpha^u$ and $\forall t > 0 \ . \ \alpha^l(t) \neq \infty$ (since α^l is a lower bound, it cannot have an infinite value).

Let R be a cumulative curve and (α^u, α^l) a pair of arrival curves. *R satisfies* (α^u, α^l) (we also use "*R complies with* (α^u, α^l)"), noted $R \models (\alpha^u, \alpha^l)$, iff(def)

$$\forall x \in \mathcal{T}, \forall \delta \in \mathcal{T}, \quad R(x + \delta) - R(x) \in [\alpha^l(\delta), \alpha^u(\delta)]$$

We say that an arrival curve (α^u, α^l) is *satisfiable* iff(def) there exists a cumulative curve R that satisfies (α^u, α^l).

Definition 8 Let (α^u, α^l) and $(\alpha^{u\prime}, \alpha^{l\prime})$ be two arrival curves. (α^u, α^l) and $(\alpha^{u\prime}, \alpha^{l\prime})$ are *equivalent* iff(def) for all cumulative curves $R \in \mathcal{F}_{finite}$,

$$R \models (\alpha^u, \alpha^l) \iff R \models (\alpha^{u\prime}, \alpha^{l\prime})$$

2.2.2 Sub-additivity and Super-additivity

As shown in the introduction, a pair of arrival curves can have unreachable regions. A curve with no unreachable regions is sub-additive if it is an upper curve α^u, and super-additive if it is a lower curve α^l. When the curve has unreachable regions, it is possible to remove them, using the associated closure operation.

Definition 9 A pair of arrival curves (α^u, α^l) is *Sub-Additive-Super-Additive* (*SA-SA* for short) iff(def) α^u is sub-additive and α^l super-additive.

We denote by $(\overline{\alpha^u}, \underline{\alpha^l})$ the SA-SA closure of (α^u, α^l).

Lemma 2 *Let (α^u, α^l) be a pair of arrival curves. $(\overline{\alpha^u}, \underline{\alpha^l})$ is a SA-SA pair of arrival curves* equivalent to (α^u, α^l).

Proof This is a well-known result [25, 37], and a corollary of Lemma 3 and 5 presented below. □

2.2.3 Arrival Curves Satisfied "Up to T"

Real-time calculus usually deals with infinite event streams and cumulative curves which should comply with arrival curves on their entire domain $\mathcal{T}$. In this paper, we need in addition to define that a cumulative curve satisfies a pair of arrival curves *up to* a certain time T. Instead of checking event counts in any window of time, we only check windows of time ending before this time T. The same notion is used, for example, in [17].

Definition 10 Let (α^u, α^l) be a pair of arrival curves and R a cumulative curve. Let $T \in \mathcal{T}$. R satisfies (α^u, α^l) *up to* T, denoted by $R \models_{\leq T} (\alpha^u, \alpha^l)$, iff(def)

$$\forall t \leq T, \forall \delta \leq t, \quad R(t) - R(t-\delta) \in [\alpha^l(\delta), \alpha^u(\delta)]$$

Intuitively, this means that R did not yet violate the arrival curves at time T. A relationship between $\models_{\leq T}$ and $\models$ can be simply expressed:

Lemma 3 *Let R be a cumulative curve, and (α^u, α^l) be a pair of arrival curves. We have:*

$$\forall T \in \mathcal{T}, R \models_{\leq T} (\alpha^u, \alpha^l) \quad \iff \quad R \models (\alpha^u, \alpha^l)$$

Proof The proof is trivial once the definition of $R \models_{\leq T}$ is expanded. □

3 Causality: Definition and Sufficient Condition

We now define the notion of causality. We consider the problem of an event stream which is correct, *w.r.t* a given pair of arrival curves, up to a given time T, but "cannot be continued" without violating this pair of curves. This can be seen as a deadlock of the stream, since it can neither let time elapse nor emit any additional event. A pair of arrival curves for which this problem can never happen is called *causal*. We first give a formal definition of causality and then give a (necessary and sufficient) characterization of the definition using algebraic formulas.

3.1 Definition of Causality

Definition 11 Let (α^u, α^l) be a pair of arrival curves.

Informally, (α^u, α^l) is causal when for any time T, any cumulative curve R that satisfies (α^u, α^l) *up to* T can be indefinitely extended into a cumulative curve R' that also satisfies (α^u, α^l). Formally, (α^u, α^l) is *causal* iff(def)

$$\forall T \geq 0, \forall R, \;\; R \models_{\leq T} (\alpha^u, \alpha^l) \implies \exists R', R' \models (\alpha^u, \alpha^l) \wedge \forall t \leq T, R(t) = R'(t)$$

Note that if (α^u, α^l) is a causal pair of arrival curves then it is satisfiable. Indeed, as $\alpha^u(0) = \alpha^l(0) = 0$, a cumulative curve R such that $R(0) = 0$ satisfies (α^u, α^l) until $T = 0$. Using the definition, there exists a cumulative curve R' which extends R and satisfies the curves.

Unlike sub-additivity and super-additivity, causality is really a property on a *pair* of curves. It does not make sense to say that α^u alone, or α^l alone, is causal. Indeed, the only way to observe that a cumulative curve cannot be extended is to exhibit a contradiction between a bound from the upper curve and a bound from the lower curve.

Some approaches use one curve only. For example, [35, 11] use an upper bound only; similarly, [19] works with some lower bound service curve. Such approaches *do not* face the causality problem at all. However, it is a common practice in Real-Time Calculus to manipulate pairs of curves, from one of the founding works [9] to more recent applications (*e.g.* [22, 30, 23, 24, 31, 10, 5, 12, 17, 36, 26]). Including both an upper and a lower bound increases precision of the analysis. Also, it can be part of the specification (*e.g.* an event generator can be specified with both upper and lower bounds [5] to avoid generating irrelevant traces).

Causality reveals new implicit constraints (see Figure 1). When curves are not causal, there exists another pair of curves defining the same set of cumulative curves, but with those constraints made explicit. Just like SA-SA closure computes an equivalent pair of curves where the unreachable regions are made explicit, we will build a new pair of curves where implicit constraints due to causality are made explicit. This new pair of curves will be equivalent to the first one; it will be called *causality closure*. The goal of the following results is to build properties about causality, in order, next, to be able to characterize and then compute this causality closure.

3.2 Sufficient Conditions for Causality: Intuitions and Schema of the Results

The formal definition of a causal pair of curves enables to derive sufficient conditions to detect whether a pair of curves is causal or not. This condition will be directly used to build the causality closure of the curves. We first give some intuitions about the results and explain the schema of proofs in this paragraph (3.2); next paragraph (3.3) details theorems and their proofs.

Informally, we call *forbidden regions* the points between α^u and α^l that are reachable by finite cumulative curves, but for which the cumulative curves can trivially not be extended into infinite ones.

Let us consider the curve α^l, and define ${\alpha^l}^*$ informally as "α^l without its forbidden regions": we give here some intuition on how to compute it. ${\alpha^l}^*(\Delta_0)$ is the smallest value above $\alpha^l(\Delta_0)$ for which a cumulative curve R verifying $\forall t \in [0, T], \Delta \in [0, \Delta_0], R(t+\Delta) - R(t) \geq {\alpha^l}^*(\Delta)$ is guaranteed to be extensible infinitely beyond T by emitting the maximum number of events allowed by α^u, without violating α^l. This the same as saying that if $R(t+\Delta) - R(t) < {\alpha^l}^*(\Delta)$ for some t, Δ, then R cannot be extended without violating either α^u or α^l, which means that the region below ${\alpha^l}^*$ is forbidden. Computing the forbidden region of α^l at abscissa Δ_0 means therefore computing the lowest N for which $\alpha^u(\Delta) + N$ would not cross $\alpha^l(\Delta_0 + \Delta)$ for some $\Delta \geq 0$. This is equivalent to finding the supremum of N for which the curves would intersect. Formally, this can be written as ${\alpha^l}^*(\Delta_0) = \sup_{\Delta \geq 0} \left\{ \alpha^l(\Delta_0 + \Delta) - \alpha^u(\Delta) \right\}$: this is deconvolution $\alpha^l \oslash \alpha^u$. A similar reasoning would lead to the curve $\alpha^u \overline{\oslash} \alpha^l$ for the forbidden regions of α^u.

This explanation is far from being a proof, but gives a piece of intuition, and exhibits a formula using deconvolution operators $\oslash$ $\overline{\oslash}$. Following this intuition, we can therefore define forbidden regions as the area between a curve α^u (resp. α^l), and $\alpha^u \overline{\oslash} \alpha^l$ (resp. $\alpha^l \oslash \alpha^u$). Intuitively, computing $\alpha^u \overline{\oslash} \alpha^l$ means "removing forbidden regions from α^u", and computing $\alpha^l \oslash \alpha^u$ means "removing forbidden regions from α^l". When $\alpha^u = \alpha^u \overline{\oslash} \alpha^l$ and $\alpha^l = \alpha^l \oslash \alpha^u$, we can say that the curves have no

$$\begin{tikzcd}
\begin{array}{c} \alpha^l = \alpha^l \oslash \alpha^u \\ \text{and} \\ \alpha^u = \alpha^u \,\overline{\oslash}\, \alpha^l \end{array} \arrow[r, Rightarrow, "\mathbf{(e)}"] \arrow[d, Rightarrow, "\mathbf{(d)}"] & (\alpha^u, \alpha^l) \text{ is causal} \arrow[d, Leftrightarrow, "\mathbf{(b)}"] \\
\begin{array}{c} \underline{\alpha^l} = \underline{\alpha^l} \oslash \overline{\alpha^u} \\ \text{and} \\ \overline{\alpha^u} = \overline{\alpha^u} \,\overline{\oslash}\, \underline{\alpha^l} \end{array} \arrow[r, Rightarrow, "\mathbf{(a)}"] \arrow[ur, Rightarrow, "\mathbf{(c)}"] & (\overline{\alpha^u}, \underline{\alpha^l}) \text{ is causal}
\end{tikzcd}$$

Fig. 2 Overall view of theorems in this section

forbidden region. This is a sufficient condition for (α^u, α^l) being causal; it is formally expressed in implication **(e)** of Figure 2.

Next section provides a proof of this result. The intermediate results of the proof are summarized in Figure 2. Implications **(c)** (Theorem 2), and **(e)** (Theorem 4), give algebraic sufficient conditions for causality. Intuitively, they state that a pair of curves is causal if either the pair of curves **(e)** or it SA-SA closure **(c)** has no forbidden region.

One could have expected the converse of **(e)**, *i.e.* that a causal pair of arrival curves has no forbidden region. This implication is indeed false in general, because a causal pair of curves can have unreachable forbidden regions (see counter-example in Section 3.3.6). Nevertheless, the converse of **(c)** and the converse of **(a)** are true in many cases. Actually, Section 4 proves them for a wide class of curves and conjecture their validity in the general case.

Precisely, intermediate results of the proof (see Figure 2) are proved in the following order:

(a) is given by Lemma 4 when applied to $(\overline{\alpha^u}, \underline{\alpha^l})$.
(b) (Theorem 1) is based on the fact that $(\overline{\alpha^u}, \underline{\alpha^l})$ and (α^u, α^l) accept the same set of cumulative curves.
(c) (Theorem 2) is obtained by transitivity of **(a)** and **(b)**.
(d) (Theorem 3) is proved using induction and continuity of deconvolution operators.
(e) (Theorem 4) is obtained by transitivity of **(c)** and **(d)**.

3.3 Sufficient Conditions for Causality: Theorems and Proofs

3.3.1 Sufficient Condition for Causality of SA-SA Curves

The following lemma gives a sufficient condition for causality of SA-SA pairs of curves. A strictly more general result will be given in Theorem 2, which uses this first one in its proof. It makes the link between forbidden regions and causality: an SA-SA pair of curves is causal if it does not have forbidden regions.

Lemma 4 (Sufficient Condition for Causality of SA-SA Curves) *Let* (α^u, α^l) *be a SA-SA pair of curves. We have:*

$$\begin{pmatrix} \alpha^l = \alpha^l \oslash \alpha^u \\ \textit{and} \\ \alpha^u = \alpha^u \,\overline{\oslash}\, \alpha^l \end{pmatrix} \Rightarrow (\alpha^l, \alpha^u) \textit{ is causal}$$

Applied to $(\overline{\alpha^u}, \underline{\alpha^l})$, this is implication **(a)** on Figure 2.

Proof Let (α^u, α^l) be a SA-SA pair of arrival curves such that $\alpha^l = \alpha^l \oslash \alpha^u$ and $\alpha^u = \alpha^u \,\overline{\oslash}\, \alpha^l$. Let $T \in \mathcal{T}$ be a time and R a cumulative curve such that $R \models_{\leq T} (\alpha^u, \alpha^l)$.

To prove the lemma, we need to show that R can be extended into some cumulative curve R' such that $R' \models (\alpha^u, \alpha^l)$. We build R' such that the lowest number of events is emitted while still complying with the lower bounds required by α^l and the prefix of R up to T. This cumulative curve, R', will indeed be valid with respect to (α^u, α^l), and infinite.

Formally, let R' be the cumulative curve defined by

$$\begin{aligned} \forall t \leq T, R'(t) =& R(t) \\ \forall t > T, R'(t) =& \sup_{x \in [0,T]} \{R(x) + \alpha^l(t - x)\} \end{aligned}$$

To show that $R' \models (\alpha^u, \alpha^l)$, we prove first that $R' \models \alpha^l$ and then $R' \models \alpha^u$.

Proof of $R' \models \alpha^l$: This part uses only the definition of R' and needs no hypothesis on (α^u, α^l). Let $t \in \mathcal{T}, \Delta \in [0, t]$. If $t \leq T$, then by definition of R', $R'(t) - R'(t - \Delta) \in [\alpha^l(\Delta), \alpha^u(\Delta)]$. We now consider $t > T$, and distinguish two cases on Δ:

- If $t - \Delta > T$, then

$$R'(t) - R'(t - \Delta) = \sup_{x \in [0,T]} \{R(x) + \alpha^l(t - x)\} - \sup_{x \in [0,T]} \{R(x) + \alpha^l(t - \Delta - x)\}$$

For all $x \in [0, T]$, we can write

$$\begin{aligned} \alpha^l(t - x - \Delta) + \alpha^l(\Delta) &\leq \alpha^l(t - x) && \text{(by super-additivity of } \alpha^l) \\ R(x) + \alpha^l(t - x - \Delta) + \alpha^l(\Delta) &\leq \alpha^l(t - x) + R(x) \end{aligned}$$

Hence,

$$\begin{aligned} \sup_{x \in [0,T]} \left\{R(x) + \alpha^l(t - x - \Delta) + \alpha^l(\Delta)\right\} &\leq \sup_{x \in [0,T]} \{R(x) + \alpha^l(t - x)\} \\ \sup_{x \in [0,T]} \left\{R(x) + \alpha^l(t - x - \Delta)\right\} + \alpha^l(\Delta) &\leq \sup_{x \in [0,T]} \{R(x) + \alpha^l(t - x)\} \\ R'(t - \Delta) + \alpha^l(\Delta) &\leq R'(t) && \text{(def. of } R') \end{aligned}$$

- If $t - \Delta \leq T$, then

$$\begin{aligned} R'(t) - R'(t - \Delta) &= \sup_{x \in [0,T]} \{R(x) + \alpha^l(t - x)\} - R(t - \Delta) \\ &\geq R(t - \Delta) + \alpha^l(t - (t - \Delta)) - R(t - \Delta) \quad \text{(with } x = t - \Delta) \\ &\geq \alpha^l(\Delta) \end{aligned}$$

In both cases, $R'(t) - R'(t - \Delta) \geq \alpha^l(\Delta)$, therefore $R' \models \alpha^l$.

Proof of $R' \models \alpha^u$: We distinguish the same two cases as above on $t - \Delta$:

- If $t - \Delta > T$, then
  For all $x \in [0, T]$, we can write

$$\begin{aligned} \alpha^l(t - x - \Delta) + \alpha^u(\Delta) &\geq \alpha^l(t - x) && \text{(since } \alpha^l = \alpha^l \oslash \alpha^u\text{)} \\ R(x) + \alpha^l(t - x - \Delta) + \alpha^u(\Delta) &\geq \alpha^l(t - x) + R(x) \end{aligned}$$

  Hence,

$$\begin{aligned} \sup_{x \in [0,T]} \left\{R(x) + \alpha^l(t - x - \Delta) + \alpha^u(\Delta)\right\} &\geq \sup_{x \in [0,T]} \{\alpha^l(t - x) + R(x)\} \\ \sup_{x \in [0,T]} \left\{R(x) + \alpha^l(t - x - \Delta)\right\} + \alpha^u(\Delta) &\geq \sup_{x \in [0,T]} \{\alpha^l(t - x) + R(x)\} \\ R'(t - \Delta) + \alpha^u(\Delta) &\geq R'(t) && \text{(def. of } R'\text{)} \end{aligned}$$

- If $t - \Delta \leq T$, then, by definition of R',

$$R'(t) - R'(t - \Delta) = R'(t) - R(t - \Delta) = \sup_{x \in [0,T]} \{R(x) - R(t - \Delta) + \alpha^l(t - x)\}$$

  For all $x \in [t - \Delta, T]$ we can write

$$\begin{aligned} R(x) - R(t - \Delta) \leq& \alpha^u(x - t + \Delta) && \text{(since } R \models_{\leq T} (\alpha^u, \alpha^l)\text{)} \\ \alpha^l(t - x) + R(x) - R(t - \Delta) \leq& \alpha^u(x - t + \Delta) + \alpha^l(t - x) \end{aligned}$$

  As $\alpha^u = \alpha^u \mathbin{\overline{\oslash}} \alpha^l$, $\alpha^u(x - t + \Delta) + \alpha^l(t - x) \leq \alpha^u(\Delta)$. Using the line above, we get, for all $x \in [t - \Delta, T]$

$$\alpha^l(t - x) + R(x) - R(t - \Delta) \leq \alpha^u(\Delta)$$

  Now, for all $x \in [0, t - \Delta]$ we can write

$$\begin{aligned} R(t - \Delta) - R(x) \geq& \alpha^l(t - \Delta - x) && \text{(since } R \models_{\leq T} (\alpha^u, \alpha^l)\text{)} \\ \alpha^l(t - x) + R(x) - R(t - \Delta) \leq& \alpha^l(t - x) - \alpha^l(t - \Delta - x) \end{aligned}$$

  As $\alpha^l = \alpha^l \oslash \alpha^u$, $\alpha^l(t - x) - \alpha^l(t - \Delta - x) \leq \alpha^u(\Delta)$. Hence, for all $x \in [0, t - \Delta]$,

$$\alpha^l(t - x) + R(x) - R(t - \Delta) \leq \alpha^u(\Delta)$$

  The two sub-cases lead to the same conclusion:

$$\begin{aligned} \forall x \in [0, T], \quad \alpha^l(t - x) + R(x) - R(t - \Delta) &\leq \alpha^u(\Delta) \\ \sup_{x \in [0,T]} \{\alpha^l(t - x) + R(x)\} - R(t - \Delta) &\leq \alpha^u(\Delta) \\ R'(t) - R'(t - \Delta) &\leq \alpha^u(\Delta) \end{aligned}$$

  Indeed, $t - \Delta \leq T$ and $R(t - \Delta) = R'(t - \Delta)$). So, R' is valid w.r.t. α^u. This concludes the proof.

□

Notice that the proof uses the fact that α^l is super-additive; nevertheless we show later that the implication actually holds for any pair of arrival curves.

3.3.2 Causality Preservation by SA-SA Closure

The following theorem is the equivalence **(b)** in Figure 2.

Theorem 1 *Let (α^u, α^l) be a pair of arrival curves.*

$$(\alpha^u, \alpha^l) \text{ is causal} \Longleftrightarrow (\overline{\alpha^u}, \underline{\alpha^l}) \text{ is causal}$$

The proof uses the following lemma:

Lemma 5 *For any pair of arrival curves (α^u, α^l), any $T \geq 0$, and any cumulative curve R, we have:*

$$R \models_{\leq T} (\alpha^u, \alpha^l) \Longleftrightarrow R \models_{\leq T} (\overline{\alpha^u}, \underline{\alpha^l})$$

Proof (Lemma 5) To simplify the notations, we write $\alpha^{l(n)} \overset{\text{def}}{=} \overline{\otimes}^n \alpha^l$ and $\alpha^{u(n)} \overset{\text{def}}{=} \otimes^n \alpha^u$. The notation is ambiguous, but we use this shortcut only when it is clear from the context (lower bounds α^l always use the $\overline{\otimes}$ operator, while upper bounds α^u always use the $\otimes$ operator).

We first focus on the direct implication ($\Longrightarrow$). We show by induction that $\forall N \geq 1, R \models_{\leq T} \alpha^{u(N)}$. The base case $N = 1$ is exactly the hypothesis. We now assume that $R \models_{\leq T} \alpha^{u(N)}$:

$$\begin{aligned}
&\forall t \leq T, \forall \Delta \in [0, t], \forall s \in [t - \Delta, t] \\
&\qquad R(t) - R(t - \Delta) = R(t) - R(s) + R(s) - R(t - \Delta) \\
&\qquad\qquad \leq \alpha^{u(N)}(t - s) + \alpha^u(s - (t - \Delta))
\end{aligned}$$

Therefore, $\forall t \leq T$,

$$R(t) - R(t - \Delta) \leq \inf_{t - s \in [0, \Delta]} \left\{ \alpha^{u(N)}(t - s) + \alpha^u(s - (t - \Delta)) \right\} = \alpha^{u(N+1)}(\Delta)$$

Which concludes the induction:

$$\forall N \in \mathcal{N}, \forall t \geq T, \forall \Delta \in [0, t] \quad R(t) - R(t - \Delta) \leq \alpha^{u(N)}(\Delta)$$

Hence,

$$\forall t \geq T, \forall \Delta \in [0, t], \quad R(t) - R(t - \Delta) \leq \inf_{N \geq 1} \{\alpha^{u(N)}(\Delta)\} = \overline{\alpha^u}(\Delta)$$

This concludes the proof for α^u. The proof for α^l would be the same.

The reverse implication ($\Longleftarrow$) is trivial since $\underline{\alpha^l} \geq \alpha^l$ and $\overline{\alpha^u} \leq \alpha^u$. □

Proof (Theorem 1) The definition of causality states that:

$$(\alpha^u, \alpha^l) \text{ causal} \iff \left(\begin{array}{c} \forall T \geq 0, \forall R,\ R \models_{\leq T} (\alpha^u, \alpha^l) \\ \Longrightarrow \\ \exists R' \mid R' \models (\alpha^u, \alpha^l) \text{ and } \forall t \leq T, R(t) = R'(t) \end{array} \right)$$

Lemma 2 applied to some R' gives $R' \models (\alpha^u, \alpha^l) \iff R' \models (\overline{\alpha^u}, \underline{\alpha^l})$, and Lemma 5 gives $\forall T \geq 0, R \models_{\leq T} (\alpha^u, \alpha^l) \iff R \models_{\leq T} (\overline{\alpha^u}, \underline{\alpha^l})$. Therefore, the equation above can be rewritten as

$$\begin{aligned}(\alpha^u, \alpha^l) \text{ causal } &\iff \begin{pmatrix} \forall T \geq 0, \forall R, \big(R \models_{\leq T} (\overline{\alpha^u}, \underline{\alpha^l})\big) \\ \Longrightarrow \\ \exists R' \mid R' \models (\overline{\alpha^u}, \underline{\alpha^l}) \text{ and } \forall t \leq T, R(t) = R'(t) \end{pmatrix} \\ &\iff (\overline{\alpha^u}, \underline{\alpha^l}) \text{ is causal}\end{aligned}$$

□

3.3.3 Sufficient Condition for Causality

The result given below is a variant of Lemma 4, which does not require the curves to be SA-SA. This is implication **(c)** in Figure 2 which is obtained by transitivity of Theorem 1 and Lemma 4.

Theorem 2 *Let (α^u, α^l) be a pair of arrival curves.*

$$\begin{pmatrix} \underline{\alpha^l} = \underline{\alpha^l} \oslash \overline{\alpha^u} \\ \textit{and} \\ \overline{\alpha^u} = \overline{\alpha^u} \mathbin{\overline{\oslash}} \underline{\alpha^l} \end{pmatrix} \Longrightarrow (\alpha^u, \alpha^l) \textit{ is causal}$$

3.3.4 Absence of Forbidden Regions Implies Causality

The main result of the section states that if a pair of curves has no forbidden region, then its SA-SA closure has none either. This is stated by implication **(d)** in Figure 2 and by Theorem 6:

Theorem 3 *Let (α^u, α^l) be a pair of arrival curves.*

$$\begin{pmatrix} \alpha^l = \alpha^l \oslash \alpha^u \\ \textit{and} \\ \alpha^u = \alpha^u \mathbin{\overline{\oslash}} \alpha^l \end{pmatrix} \Rightarrow \begin{pmatrix} \underline{\alpha^l} = \underline{\alpha^l} \oslash \overline{\alpha^u} \\ \textit{and} \\ \overline{\alpha^u} = \overline{\alpha^u} \mathbin{\overline{\oslash}} \underline{\alpha^l} \end{pmatrix}$$

We prove this theorem in several steps, given by the following lemmas:

Lemma 6 *Let (α^u, α^l) be a pair of arrival curves.*

$$\begin{pmatrix} \alpha^l = \alpha^l \oslash \alpha^u \\ \textit{and} \\ \alpha^u = \alpha^u \mathbin{\overline{\oslash}} \alpha^l \end{pmatrix} \Rightarrow \begin{pmatrix} \alpha^l = \alpha^l \oslash \overline{\alpha^u} \\ \textit{and} \\ \alpha^u = \alpha^u \mathbin{\overline{\oslash}} \underline{\alpha^l} \end{pmatrix}$$

Lemma 7 *Let (α^u, α^l) be a pair of arrival curves.*

$$\begin{pmatrix} \alpha^l = \alpha^l \oslash \alpha^u \\ \textit{and} \\ \alpha^u = \alpha^u \mathbin{\overline{\oslash}} \alpha^l \end{pmatrix} \Rightarrow \begin{pmatrix} \underline{\alpha^l} = \underline{\alpha^l} \oslash \alpha^u \\ \textit{and} \\ \overline{\alpha^u} = \overline{\alpha^u} \mathbin{\overline{\oslash}} \alpha^l \end{pmatrix}$$

For each lemma/theorem, we show only the implication of the first equality, the proof of the second would be similar.

Proof (Lemma 6) Let (α^u, α^l) be a pair of arrival curves such that $\alpha^l = \alpha^l \oslash \alpha^u$. We first prove by induction that

$$\forall N \geq 1, \quad \alpha^l \oslash \left(\alpha^{u(N)}\right) = \alpha^l$$

The base case is given by the hypothesis of the lemma. For the inductive step, assuming $\alpha^l \oslash \alpha^{u(N)} = \alpha^l$, we have:

$$\begin{aligned}
\alpha^l \oslash \alpha^{u(N+1)} &= \alpha^l \oslash \left[\alpha^{u(N)} \otimes \alpha^u\right] && \text{(by definition of } \alpha^{u(N)}\text{)} \\
&= \left[\alpha^l \oslash \alpha^{u(N)}\right] \oslash \alpha^u && \text{(since } (f \oslash g) \oslash h = f \oslash (g \otimes h)\text{,} \\
& && \text{see [25] p. 123)} \\
&= \alpha^l \oslash \alpha^u && \text{(by induction hypothesis)} \\
&= \alpha^l && \text{(hypothesis of the lemma)}
\end{aligned}$$

This concludes the induction. Now, we can write, $\forall t \geq 0$,

$$\begin{aligned}
\alpha^l(t) &= \sup_{N \geq 1} \left\{ \left(\alpha^l \oslash \left[\alpha^{u(N)}\right]\right)(t) \right\} \\
&= \sup_{N \geq 1} \left\{ \sup_{\Delta \geq 0} \left\{ \alpha^l(t + \Delta) - \left[\alpha^{u(N)}(\Delta)\right] \right\} \right\} && \text{(definition of } \oslash\text{)} \\
&= \sup_{\Delta \geq 0} \left\{ \alpha^l(t + \Delta) - \inf_{N \geq 1} \left\{ \alpha^{u(N)}(\Delta) \right\} \right\} \\
&= \sup_{\Delta \geq 0} \left\{ \alpha^l(t + \Delta) - \overline{\alpha^u}(\Delta) \right\} && \text{(definition of } \overline{\alpha^u}\text{)} \\
&= (\alpha^l \oslash \overline{\alpha^u})(t) && \text{(definition of } \oslash\text{)}
\end{aligned}$$

□

Proof (Lemma 7) Let (α^u, α^l) be a pair of arrival curves such that $\alpha^l = \alpha^l \oslash \alpha^u$. We first show, by induction, that:

$$\forall N \geq 1, \quad \alpha^{l(N)} = \alpha^{l(N)} \oslash \alpha^u$$

The base case is given by the hypothesis of the lemma. For $N \geq 1$, we assume $\alpha^{l(N)} = \alpha^{l(N)} \oslash \alpha^u$. We already have $\alpha^{l(N+1)} \leq \alpha^{l(N+1)} \oslash \alpha^u$. To prove $\alpha^{l(N+1)} = \alpha^{l(N+1)} \oslash \alpha^u$, we will show the converse inequality, *i.e*:

$$\begin{aligned}
&\forall t \geq 0, \forall \Delta \geq 0, \quad \alpha^{l(N+1)}(t) + \alpha^u(\Delta) \geq \alpha^{l(N+1)}(t + \Delta) && \textit{i.e:} \\
&\forall t \geq 0, \forall \Delta \geq 0, \quad \alpha^{l(N+1)}(t) + \alpha^u(\Delta) \geq \sup_{x \in [0, t+\Delta]} \{\alpha^{l(N)}(x) + \alpha^l(t + \Delta - x)\} && \textit{i.e:}
\end{aligned}$$

$$\forall t \geq 0, \forall \Delta \geq 0, \forall x \in [0, t + \Delta], \quad \alpha^{l(N+1)}(t) + \alpha^u(\Delta) \geq \alpha^{l(N)}(x) + \alpha^l(t + \Delta - x)$$

Let $t \geq 0$, $\Delta \geq 0$ and $x \in [0, t + \Delta]$. We distinguish two cases on x:

- If $x \geq t$, then we can write:

$$\alpha^u(\Delta) - \alpha^l(\Delta - (x-t)) \geq \alpha^u(x-t) \quad (\text{since } \alpha^u = \alpha^u \,\overline{\oslash}\, \alpha^l, \text{ hypothesis of lemma})$$
$$\alpha^{l(N)}(x) - \alpha^u(x-t) \leq \alpha^{l(N)}(t) \quad (\text{since } \alpha^{l(N)} = \alpha^{l(N)} \oslash \alpha^u, \text{ the induction hypothesis})$$

The rest of the proof follows from the combination of these equations:

$$\alpha^u(\Delta) - \alpha^l(\Delta - x + t) \geq \alpha^u(x-t) \geq \alpha^{l(N)}(x) - \alpha^{l(N)}(t)$$
$$\alpha^u(\Delta) + \alpha^{l(N)}(t) \geq \alpha^{l(N)}(x) + \alpha^l(\Delta - x + t)$$

Since $\alpha^{l(N+1)}(t) \geq \alpha^{l(N)}(t)$,

$$\alpha^u(\Delta) + \alpha^{l(N+1)}(t) \geq \alpha^u(\Delta) + \alpha^{l(N)}(t) \geq \alpha^{l(N)}(x) + \alpha^l(\Delta - x + t)$$
$$\alpha^{l(N+1)}(t) + \alpha^u(\Delta) \geq \alpha^{l(N)}(x) + \alpha^l(t + \Delta - x) \quad (\text{reordering})$$

- If $x < t$, then we can apply a similar reasoning, swapping $x - t$ and $t - x$:

$$\alpha^l(t + \Delta - x) - \alpha^u(\Delta) \leq \alpha^l(t-x) \quad (\text{since } \alpha^l = \alpha^l \oslash \alpha^u, \text{ hypothesis of lemma})$$
$$\alpha^{l(N+1)}(t) \geq \alpha^{l(N)}(x) + \alpha^l(t-x) \quad (\text{by definition of } \alpha^{l(N+1)})$$

The rest of the proof follows from the combination of these equations:

$$\alpha^l(t + \Delta - x) - \alpha^u(\Delta) \leq \alpha^l(t-x) \leq \alpha^{l(N+1)}(t) - \alpha^{l(N)}(x)$$
$$\alpha^{l(N+1)}(t) + \alpha^u(\Delta) \geq \alpha^{l(N)}(x) + \alpha^l(t + \Delta - x) \quad (\text{reordering})$$

Both cases prove that $\alpha^{l(N+1)}(t) + \alpha^u(\Delta) \geq \alpha^{l(N)}(x) + \alpha^l(t + \Delta - x)$, and thus $\forall t \geq 0, \Delta \geq 0, \alpha^{l(N+1)}(t) + \alpha^u(\Delta) \geq \alpha^{l(N+1)}(t + \Delta)$. This implies by definition of $\oslash$ that $\forall t \geq 0, \alpha^{l(N+1)}(t) \geq (\alpha^{l(N+1)} \oslash \alpha^u)(t)$. As for any functions f and g, $f \oslash g \geq f$, the converse inequality holds and the induction goal is proved.

Hence, $\forall N \geq 1, \alpha^{l(N)} = \alpha^{l(N)} \oslash \alpha^u$. The rest of the proof is a simple application of the lower semi-continuity of the operator $\oslash$ with respect to its left operand, as stated in [25], page 135. To make this proof self-contained, we detail the steps:

$$\begin{aligned}
\forall t, \underline{\alpha^l}(t) &= \sup_{N \geq 1} \left\{ \alpha^{l(N)}(t) \right\} && (\text{definition of } \underline{\alpha^l}) \\
&= \sup_{N \geq 1} \left\{ \left(\alpha^{l(N)} \oslash \alpha^u \right)(t) \right\} && (\text{applying the result of the induction}) \\
&= \sup_{N \geq 1} \left\{ \sup_{\Delta \geq 0} \left\{ \alpha^{l(N)}(t + \Delta) - \alpha^u(\Delta) \right\} \right\} && (\text{definition of } \oslash) \\
&= \sup_{\Delta \geq 0} \left\{ \sup_{N \geq 1} \left\{ \alpha^{l(N)}(t + \Delta) \right\} - \alpha^u(\Delta) \right\} \\
&= \underline{\alpha^l} \oslash \alpha^u && (\text{definition of } \underline{\alpha^l} \text{ and } \oslash)
\end{aligned}$$

□

Proof (Theorem 3) Let (α^u, α^l) be a pair of arrival curves, such that $\alpha^l = \alpha^l \oslash \alpha^u$ and $\alpha^u = \alpha^u \,\overline{\oslash}\, \alpha^l$.

From Lemma 6 and 7, the following equalities also hold:

$$\alpha^l = \alpha^l \oslash \overline{\alpha^u} \tag{1}$$

$$\overline{\alpha^u} = \overline{\alpha^u} \,\overline{\oslash}\, \alpha^l \tag{2}$$

$$\alpha^u = \alpha^u \,\overline{\oslash}\, \underline{\alpha^l} \tag{3}$$

$$\underline{\alpha^l} = \underline{\alpha^l} \oslash \alpha^u \tag{4}$$

Equations 1 and 2 ensure the hypotheses which allow to apply Lemma 7 to $(\overline{\alpha^u}, \alpha^l)$ and get $\underline{\alpha^l} = \underline{\alpha^l} \oslash \overline{\alpha^u}$. Similarly, Equations 3 and 4 allow to apply Lemma 7 to $(\alpha^u, \underline{\alpha^l})$, which provides $\overline{\alpha^u} = \overline{\alpha^u} \,\overline{\oslash}\, \underline{\alpha^l}$. □

3.3.5 Sufficient Condition for Causality

The last theorem of this section gives a sufficient condition for the causality of a curve. Unlike Lemma 4, it applies to any pairs of curves, and its hypothesis is simpler than the one of Theorem 2. Informally, it states that a pair of curves without forbidden regions is causal (see implication **(e)** in Figure 2). The result is obtained by transitivity of Theorem 3 and Lemma 4.

Theorem 4 *Let (α^u, α^l) be a pair of arrival curves.*

$$\begin{pmatrix} \alpha^l = \alpha^l \oslash \alpha^u \\ \textit{and} \\ \alpha^u = \alpha^u \,\overline{\oslash}\, \alpha^l \end{pmatrix} \implies (\alpha^u, \alpha^l) \textit{ is causal}$$

3.3.6 Causality does not Imply Absence of Forbidden Regions

Theorem 4 provides a sufficient condition to ensure that a pair of arrival curves is causal. Unfortunately, this condition is not necessary, as shown in the counter-example of Figure 3. The vertically hatched region is a forbidden region, and we do not have $\alpha^l = \alpha^l \oslash \alpha^u$, but the pair of curves is still causal. Actually, the forbidden region is below $\underline{\alpha^l}$, so it is not reachable.

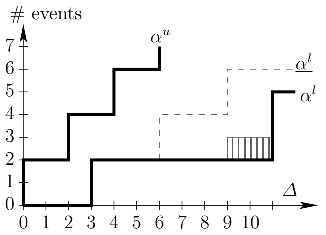


Fig. 3 Causal Curve with an Unreachable Forbidden Region

4 Characterization of Causality for Piecewise-Affine Curves and Conjecture

This section presents a characterization of causal arrival curves when they have the particular shape of piecewise affine functions. This characterization is still a conjecture in the general case — left as an open problem —, since we did not succeed in proving it nor building counter-examples.

4.1 Validity of Causal Curves with Respect to Themselves: a Conjecture

The characterization property relies on the fact that causal and SA-SA arrival curves are valid cumulative curves with respect to themselves, *i.e.* $\alpha^u \models (\alpha^u, \alpha^l)$ and $\alpha^l \models (\alpha^u, \alpha^l)$. We enunciate it as a conjecture, since we can prove it for piecewise affine curves, but it remains open in the general case. Intuitively, the conjecture means that if the arrival curves have neither "forbidden" nor "unreachable" regions, then we can follow either the lower or the upper curve to get a valid cumulative curve.

Conjecture 1 Let (α^u, α^l) be a causal pair of arrival curves. If α^l is super-additive, then $\alpha^l \models (\alpha^u, \alpha^l)$. Similarly, if α^u is sub-additive and is finite ($\in \mathcal{F}_{finite}$), then $\alpha^u \models (\alpha^u, \alpha^l)$.

This conjecture was originally presented in [2] as a theorem, but the proof given was actually valid only for discrete-event systems.

Definition 12 A function is *finitely piecewise-affine* if and only if its restriction to any finite interval of the form $[0, T]$, for $T \in \mathcal{T}$, is made of a finite number of affine pieces.

The class of finitely piecewise-affine functions includes arrival curves in the discrete-time model, in the discrete-event model (assuming $\forall \Delta, \alpha^u(\Delta) \neq +\infty$), and in the commonly used "piecewise-affine, pseudo-periodic" model [32, 6].

Lemma 8 *Conjecture 1 holds for finitely piecewise-affine curves.*

Proof We show that $\alpha^l \models (\alpha^u, \alpha^l)$. Since α^l is super-additive, it complies with itself by definition. The only way to have $\alpha^l \not\models (\alpha^u, \alpha^l)$ is to violate the constraint on α^u. The rest of the proof is done by contradiction:

Let $T = \sup_{t \geq 0} \{\alpha^l \models_{\leq t} (\alpha^u, \alpha^l)\}$. Since we assume $\alpha^l \not\models (\alpha^u, \alpha^l)$, then T exists and is finite (by Lemma 3). Two cases can happen:

- If $\alpha^l \not\models_{\leq T} (\alpha^u, \alpha^l)$, then $\exists \Delta > 0 \mid \alpha^l(T) - \alpha^l(T - \Delta) > \alpha^u(\Delta)$. Then, $\alpha^l \models_{\leq T - \Delta} (\alpha^u, \alpha^l)$ (because $T - \Delta < T$ and by definition of T), and α^l cannot be extended until t without violating α^u, which would conclude the proof.
- If $\alpha^l \models_{\leq T} (\alpha^u, \alpha^l)$, then $\forall t > T, \alpha^l \not\models_{\leq t} (\alpha^u, \alpha^l)$ (by definition of T), *i.e.* $\forall t > T, \exists \Delta_t > 0 \mid \alpha^l(t) - \alpha^l(t - \Delta_t) > \alpha^u(\Delta_t)$.

  Because (α^u, α^l) is finitely piecewise-affine, we can set t close enough to T to have only one affine piece of α^l within $(T, t]$, *i.e.* $\exists A \geq 0, B \leq 0$ such that:

  $$\forall \delta \in (T, t], \quad \alpha^l(\delta) = A\delta + B$$

If $\Delta_t \geq t - T$, then $\alpha^l \models_{\leq t-\Delta_t} (\alpha^u, \alpha^l)$, and α^l cannot be extended beyond T without violating α^u, which would conclude the proof. We can therefore concentrate on the case $\Delta_t < t - T$. Since α^l is non-decreasing, $\alpha^l(T) \leq (AT + B)$ and

$$\begin{aligned}
\alpha^l(T+\Delta_t) - \alpha^l(T) &\geq A(T+\Delta_t) + B - (AT+B) = A\Delta_t \\
&\geq (At+B) - (A(t-\Delta_t)+B) \\
&\geq \alpha^l(t) - \alpha^l(t-\Delta_t) && \text{(definition of } A \text{ and } B) \\
&> \alpha^u(\Delta_t) && \text{(definition of } \Delta_t)
\end{aligned}$$

Therefore, α^l cannot be prolonged after T without violating α^u.

□

Unfortunately, the proof does not extend to the general case in fluid-event, continuous time model. Intuitively, to prove the conjecture, one would need to show that $\alpha^l \not\models (\alpha^u, \alpha^l)$ implies the non-causality, and hence show the existence of a cumulative curve that is valid up to some point, and non-extensible infinitely. It is tempting to try to construct such cumulative curve by letting it enter a forbidden region before violating (α^u, α^l), but doing so is not trivial: not all forbidden regions are reachable (see Figure 4.(a) for an example), and the notion of “first forbidden region” may not be defined in the continuous case. See an example in Figure 4.(b), showing a pair of curves with an infinite number of forbidden region. For any forbidden region, there exists another one, closer to the origin, that makes it unreachable. However, it does not give a counter-example for the conjecture because α^l is not super-additive.

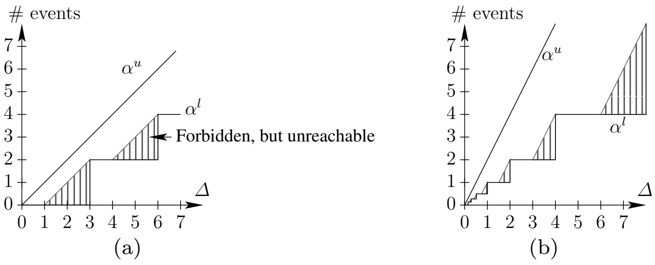


Fig. 4 Why the proof for Lemma 8 does not apply in the general case

4.2 Conjecture about the Characterization of Causality

Assuming Conjecture 1 (*i.e.* either in case of finitely piecewise-affine curves, or if a proof is found in general), we can strengthen some of the implications in Figure 2, page 11. Figure 5 shows the result. Propositions are the same, but implications **(a)** and **(c)** have been promoted to equivalences **(A)** and **(C)**.

We now prove equivalence **(A)**, previously implication **(a)**, proved in Lemma 4.

$$\begin{tikzcd}
\begin{array}{c} \alpha^l = \alpha^l \oslash \alpha^u \\ \text{and} \\ \alpha^u = \alpha^u \,\overline{\oslash}\, \alpha^l \end{array} \arrow[r, Rightarrow, "\mathbf{(e)}"] \arrow[d, Rightarrow, "\mathbf{(d)}"] & (\alpha^u, \alpha^l) \text{ is causal} \arrow[d, Leftrightarrow, "\mathbf{(b)}"] \\
\begin{array}{c} \underline{\alpha^l} = \underline{\alpha^l} \oslash \overline{\alpha^u} \\ \text{and} \\ \overline{\alpha^u} = \overline{\alpha^u} \,\overline{\oslash}\, \underline{\alpha^l} \end{array} \arrow[r, Leftrightarrow, "\mathbf{(A)}"] \arrow[ur, Rightarrow, "\mathbf{(C)}"] & (\overline{\alpha^u}, \underline{\alpha^l}) \text{ is causal}
\end{tikzcd}$$

Fig. 5 Overall view of theorems valid assuming Conjecture 1 holds

Theorem 5 *Let (α^u, α^l) be a SA-SA pair of curves. Assuming Conjecture 1, we have:*

$$\begin{pmatrix} \alpha^l = \alpha^l \oslash \alpha^u \\ \textit{and} \\ \alpha^u = \alpha^u \,\overline{\oslash}\, \alpha^l \end{pmatrix} \iff (\alpha^l, \alpha^u) \textit{ is causal}$$

Proof $\Rightarrow$: This is Lemma 4.

$\Leftarrow$: Suppose (α^u, α^l) is SA-SA and causal.

Let us show that $\forall \mathit{\Delta} \in \mathcal{T}, \alpha^l(\mathit{\Delta}) = (\alpha^l \oslash \alpha^u)(\mathit{\Delta})$. By definition, $(\alpha^l \oslash \alpha^u)(\mathit{\Delta}) = \sup_{t \geq 0}\{\alpha^l(\mathit{\Delta}+t) - \alpha^u(t)\}$. This supremum is obtained for $t = 0$ with the value $\alpha^l(\mathit{\Delta})$:

Since (α^u, α^l) is SA-SA and causal, applying Conjecture 1, we know that $\alpha^l \models (\alpha^u, \alpha^l)$. So in particular, $\forall t \geq 0, \alpha^l(\mathit{\Delta} + t) - \alpha^l(\mathit{\Delta}) \leq \alpha^u(t)$, which leads to the conclusion.

We can show that $\alpha^u = \alpha^u \,\overline{\oslash}\, \alpha^l$ in a similar way. □

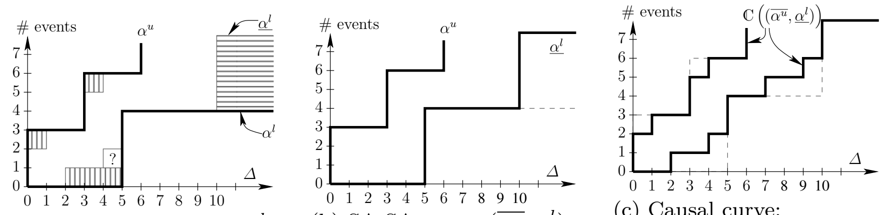
The result given below is a stronger version of Theorem 2 and can be summarized as "a pair of arrival curves is causal if and only if its SA-SA closure has no forbidden region". This is equivalence **(C)** in Figure 2. Similarly to Theorem 2, the proof is obtained by transitivity of Theorems 1 and 5.

Theorem 6 *Let (α^u, α^l) be a pair of arrival curves. Assuming Conjecture 1, we have:*

$$\begin{pmatrix} \underline{\alpha^l} = \underline{\alpha^l} \oslash \overline{\alpha^u} \\ \textit{and} \\ \overline{\alpha^u} = \overline{\alpha^u} \,\overline{\oslash}\, \underline{\alpha^l} \end{pmatrix} \iff (\alpha^u, \alpha^l) \textit{ is causal}$$

Theorems 5 and 6 give characterizations of causality for any pair of curves provided that Conjecture 1 holds. Note that by Lemma 8, those characterizations hold for any pair of finitely piecewise-affine curves.

It should be noted that although Theorems 5 and 6 are conditional (*i.e.* assume Conjecture 1 holds), the validity or invalidity of Conjecture 1 has no consequence on the following sections. Indeed, to compute causality closure, we need to show that a given pair of curves is causal (in Theorem 7), which requires one direction only of the existing implication (Lemma 4), but not the equivalence.



(a) Original curve: (α^u, α^l) (b) SA-SA curve: $(\overline{\alpha^u}, \underline{\alpha^l})$ (c) Causal curve: $\mathbb{C}(\overline{\alpha^u}, \underline{\alpha^l})$

Fig. 6 Step by step causality closure

5 Computing Causality Closure

The goal of this section is to define the causality closure of a pair of curves (α^u, α^l): it is a pair of arrival curves which is causal and equivalent to (α^u, α^l). The first step is to define the operator $\mathbb{C}$, which removes the forbidden regions from a pair of curves.

Now, removing forbidden regions transforms both curves α^u and α^l and regions removed from α^l can create new forbidden regions on the upper side, and vice versa. One natural way to solve this issue may be to iterate the forbidden region removal operator $\mathbb{C}$ until reaching some fix-point (assuming it is reached in a finite number of steps).

However, we will show that operator $\mathbb{C}$ does not create any new forbidden region *when applied on SA-SA curves* (see later Lemma 12). Hence, in cases where the SA-SA closure can be computed, it will be enough to compute it and then to apply operator $\mathbb{C}$; in other cases, a fix-point iteration of $\mathbb{C}$ will applied.

To illustrate this, an example is given in Figure 6. The original curve, in Figure 6.(a), has both forbidden regions (vertically hatched) and an unreachable region (horizontally hatched). One region of interest is the little square between $\Delta = 4$ and $\Delta = 5$, marked with a "?" in Figure 6.(a). If we consider the curves (α^u, α^l), this square does not seem to be a forbidden region. Indeed, an execution, which emits only 1 event in 4 time units, seems to be able to continue by emitting 3 events right after. Actually, this is not possible, and there are at least two ways to show this.

A first way is to apply the forbidden regions removal, $\mathbb{C}$, twice: emitting 3 events as suggested above is not possible given the leftmost forbidden region of α^u. So, the "?"-region will have to be removed, as a consequence of the forbidden region on α^u if we iterate $\mathbb{C}$ twice. After the second iteration of the forbidden region removal, we reached the fix-point, and implication (e) (Theorem 4) guarantees causality. This iterative process will be detailed in Section 7.3.

A second way to show that the "?"-region should be removed is to work on $\underline{\alpha^l}$ instead of α^l: since $\underline{\alpha^l}(10) = 8$ and $\overline{\alpha^u}(6) = 6$, an execution has to emit at least two events in 4 time units. This illustrates the approach followed in this section: we eliminate the forbidden regions with $\mathbb{C}$ (Figure 6.(c)) only after computing SA-SA closure (Figure 6.(b)).

5.1 Removing Forbidden Regions: Operator $\mathbb{C}$

We defined pairs of arrival curves as pairs (α^u, α^l) of functions for which $\alpha^u \geq \alpha^l$. In addition, we write $\bot_{AC}$ the set of pairs of functions in $\mathcal{F}$ such that the former constraint is false. To simplify notations, $\bot_{AC}$ will be used as a single element even if it represents an infinite set of objects. We note AC the set of all pairs of arrival curves *plus* $\bot_{AC}$.

Definition 13 Let (α^u, α^l) and $(\alpha^{u\prime}, \alpha^{l\prime})$ be two pairs of arrival curves. We say that $(\alpha^{u\prime}, \alpha^{l\prime})$ is tighter (meaning: more precise) than (α^u, α^l) (noted $(\alpha^{u\prime}, \alpha^{l\prime}) \leq_{AC} (\alpha^u, \alpha^l)$) iff(def) $\alpha^{l\prime} \geq \alpha^l$ and $\alpha^{u\prime} \leq \alpha^u$ We extend the $\leq_{AC}$ relation to any object of AC by: $\forall e \in AC \,.\, \bot_{AC} \leq_{AC} e$

We now define formally the "forbidden region removal" operator, $\mathbb{C}$.

Definition 14 We define operator $\mathbb{C}$ from AC to AC as:

$$\mathbb{C}\left(\bot_{AC}\right) \overset{\text{def}}{=} \bot_{AC} \quad \text{and} \quad \mathbb{C}\left(\alpha^u, \alpha^l\right) \overset{\text{def}}{=} \left(\begin{array}{l} \text{let } L = \alpha^l \oslash \alpha^u, U = \alpha^u \,\overline{\oslash}\, \alpha^l \\ \text{if } L \leq U \text{ then } (U, L) \\ \text{else } \bot_{AC} \end{array}\right.$$

When (α^u, α^l) is a pair of arrival curves then $L = \alpha^l \oslash \alpha^u$ and $U = \alpha^u \,\overline{\oslash}\, \alpha^l$ are functions in $\mathcal{F}$ (*i.e.* wide-sense increasing and equal to zero at zero, see Lemma 1). Nevertheless, they may cross each other, *i.e.*, it may happen that $L \not\leq U$. In such cases, the operator $\mathbb{C}$ returns the value $\bot_{AC}$. This means that the pair of arrival curves is not satisfiable (direct consequence Lemma 9 below).

We now show that operator $\mathbb{C}$ computes a tighter pair of arrival curves (Remark 1, trivial from the definition of $\mathbb{C}$), compared to the initial pair of curves, and which is equivalent to those (Lemma 9).

Remark 1 For any pair of arrival curves, (α^u, α^l), $\mathbb{C}\left(\alpha^u, \alpha^l\right)$ is tighter than (α^u, α^l), namely $\mathbb{C}\left(\alpha^u, \alpha^l\right) \leq_{AC} (\alpha^u, \alpha^l)$.

Lemma 9 (Equivalence of (α^u, α^l) and $\mathbb{C}\left(\alpha^u, \alpha^l\right)$) *Let (α^u, α^l) be a pair of arrival curves. Let R be a cumulative curve.*

$$R \models (\alpha^u, \alpha^l) \iff R \models (\alpha^l \oslash \alpha^u, \alpha^u \,\overline{\oslash}\, \alpha^l)$$

In particular, if $\mathbb{C}\left(\alpha^u, \alpha^l\right) = \bot_{AC}$, then (α^u, α^l) is not satisfiable; but the converse is false, in general.

Proof Let R be a cumulative curve. $R \models \mathbb{C}\left(\alpha^u, \alpha^l\right) \implies R \models (\alpha^u, \alpha^l)$ is a direct consequence of Lemma 1. Now assume that $R \models (\alpha^u, \alpha^l)$. Let $x \geq 0, t \geq 0, z \geq 0$. The following inequalities hold:

$$\begin{aligned} \alpha^l(z) \leq R(x+t+z) - R(x+t) &\leq \alpha^u(z) \\ \alpha^l(x+z) \leq \quad R(x+t+z) - R(t) \quad &\leq \alpha^u(x+z) \end{aligned}$$

As $R(x+t) - R(t) = R(x+t) - R(x+t+z) + R(x+t+z) - R(t)$, we can combine these two inequalities to get:

$$\alpha^l(x+z) - \alpha^u(z) \leq R(x+t) - R(t) \leq \alpha^u(x+z) - \alpha^l(z)$$

Since the inequalities hold for all $z \geq 0$,

$$\sup_{z \geq 0}\{\alpha^l(x+z) - \alpha^u(z)\} \leq R(x+t) - R(t) \leq \inf_{z \geq 0}\{\alpha^u(x+z) - \alpha^l(z)\}$$

And thus, for all $x \geq 0$ and $t \geq 0$,

$$(\alpha^l \oslash \alpha^u)(x) \leq R(x+t) - R(t) \leq (\alpha^u \,\overline{\oslash}\, \alpha^l)(x)$$

i.e. $R \models \mathbb{C}\left(\alpha^u, \alpha^l\right)$. Note that, as R is accepted by $\mathbb{C}\left(\alpha^u, \alpha^l\right)$, $\mathbb{C}\left(\alpha^u, \alpha^l\right)$ is not $\bot_{AC}$. □

5.2 $\mathbb{C}(\overline{\alpha^u}, \underline{\alpha^l})$: the Canonical Representative and its Properties

This section presents the main result of the paper. It basically states that $\mathbb{C}(\overline{\alpha^u}, \underline{\alpha^l})$ has many desirable properties: it is SA-SA, causal, and the best possible pair of curves equivalent to (α^u, α^l). We will start with some properties and their proof, and will proceed with the main theorems (easy to prove given the lemmas) in Section 5.2.2.

5.2.1 Properties of $\mathbb{C}(\alpha^u, \alpha^l)$ for SA-SA Curves

Lemma 10 (SA-SA preservation) *Let (α^u, α^l) be an SA-SA pair of arrival curves. If $\mathbb{C}(\alpha^u, \alpha^l) \neq \bot_{AC}$, then $\mathbb{C}(\alpha^u, \alpha^l)$ is SA-SA.*

Lemma 11 (Validity of $\mathbb{C}(\alpha^u, \alpha^l)$ with respect to themselves) *Let (α^u, α^l) be an SA-SA pair of arrival curves. If $(\alpha^{u*}, \alpha^{l^*}) \overset{def}{=} \mathbb{C}(\alpha^u, \alpha^l) \neq \bot_{AC}$, then*

$$\alpha^{l^*} \models (\alpha^{u*}, \alpha^{l^*}) \quad \textit{and} \quad \alpha^{u*} \models (\alpha^{u*}, \alpha^{l^*})$$

Lemma 12 ($\mathbb{C}$ does not add forbidden regions on SA-SA curves) *Let (α^u, α^l) be an SA-SA pair of arrival curves. $\mathbb{C}(\alpha^u, \alpha^l)$ is a fix-point of $\mathbb{C}$: $\mathbb{C}\left(\mathbb{C}(\alpha^u, \alpha^l)\right) = \mathbb{C}(\alpha^u, \alpha^l)$.*

Proof (Lemma 10) We note $(\alpha^{u*}, \alpha^{l^*}) = \mathbb{C}(\alpha^u, \alpha^l)$ with $\mathbb{C}(\alpha^u, \alpha^l) \neq \bot_{AC}$.

$$\alpha^{l^*}(t-s) + \alpha^{l^*}(s) = \sup_{\delta_t \geq 0} \left\{\alpha_l(t-s+\delta_t) - \alpha_u(\delta_t)\right\} + \sup_{\delta_s \geq 0} \left\{\alpha_l(s+\delta_s) - \alpha_u(\delta_s)\right\}$$

$\forall \varepsilon > 0, \exists \delta_t, \delta_s \geq 0,$

$$\begin{aligned}
\alpha^{l^*}(t-s) + \alpha^{l^*}(s) &\leq \alpha_l(t-s+\delta_t) - \alpha_u(\delta_t) + \alpha_l(s+\delta_s) - \alpha_u(\delta_s) + \varepsilon \\
&\leq (\alpha_l(t-s+\delta_t) + \alpha_l(s+\delta_s)) - (\alpha_u(\delta_t) + \alpha_u(\delta_s)) + \varepsilon \\
&\qquad\qquad\qquad\qquad\qquad\qquad \text{(reordering and grouping)} \\
&\leq \alpha_l(t-s+\delta_t+s+\delta_s) - \alpha_u(\delta_t+\delta_s) + \varepsilon \\
&\qquad\qquad \text{(by sub-additivity of } \alpha^u \text{ and super-additivity of } \alpha^l\text{)} \\
&\leq \alpha_l(t+\delta_t+\delta_s) - \alpha_u(\delta_t+\delta_s) + \varepsilon
\end{aligned}$$

Setting $\delta = \delta_s + \delta_t$, we get

$$\begin{aligned}
\forall \varepsilon > 0, \exists \delta \geq 0, \quad & \alpha^{l^*}(t-s) + \alpha^{l^*}(s) \leq \alpha_l(t+\delta) - \alpha_u(\delta) + \varepsilon \\
\forall \varepsilon > 0, \quad & \alpha^{l^*}(t-s) + \alpha^{l^*}(s) \leq \sup_{\delta \geq 0} \{\alpha_l(t+\delta) - \alpha_u(\delta)\} + \varepsilon \\
\forall \varepsilon > 0, \quad & \alpha^{l^*}(t-s) + \alpha^{l^*}(s) \leq \alpha^{l^*}(t) + \varepsilon \qquad \text{(definition of } \alpha^{l^*}\text{)} \\
& \alpha^{l^*}(t-s) + \alpha^{l^*}(s) \leq \alpha^{l^*}(t)
\end{aligned}$$

Hence α^{l^*} is super-additive. The proof for α^{u*} is similar. □

Proof (Lemma 11) We still use the notation $(\alpha^{u*}, \alpha^{l^*}) = \mathbb{C}(\alpha^u, \alpha^l)$ with $\mathbb{C}(\alpha^u, \alpha^l) \neq \perp_{AC}$. We'll prove the first equation (the other is similar). α^{l^*} being super-additive (Lemma 10), it is valid with respect to itself. We only have to prove that α^{l^*} is valid with respect to $(\alpha^{u*}, \alpha^{l^*})$. We first prove that it is valid with respect to α^u, *i.e.* that $\forall t \geq s \geq 0$, $\alpha^{l^*}(t) - \alpha^{l^*}(s) \leq \alpha^u(t-s)$. Let and $t \geq s \geq 0$. By definition, we have:

$$\alpha^{l^*}(s) = \sup_{y \geq 0} \left\{ \alpha^l(s+y) - \alpha^u(y) \right\}$$

Hence,

$$\forall y \geq 0, \quad \alpha^{l^*}(s) \geq \alpha^l(s+y) - \alpha^u(y)$$

For any $x \geq 0$, we set $y = x + t - s$. $y \geq 0$, hence the above inequality yields:

$$\begin{aligned}
\forall x \geq 0, \quad & \alpha^{l^*}(s) \geq \alpha^l(s + x + t - s) - \alpha^u(x + (t-s)) & \\
\forall x \geq 0, \quad & \alpha^{l^*}(s) \geq \alpha^l(x+t) - \alpha^u(x) - \alpha^u(t-s) & \text{(sub-additivity of } \alpha^u) \\
& \alpha^{l^*}(s) \geq \sup_{x \geq 0} \left\{ \alpha^l(x+t) - \alpha^u(x) - \alpha^u(t-s) \right\} & \\
& \alpha^{l^*}(s) \geq \sup_{x \geq 0} \left\{ \alpha^l(x+t) - \alpha^u(x) \right\} - \alpha^u(t-s) & \\
& \alpha^{l^*}(s) \geq \alpha^{l^*}(t) - \alpha^u(t-s) & \text{(by definition of } \alpha^{l^*}(t))
\end{aligned}$$

Hence $\alpha^{l^*} \models (\alpha^u, \alpha^{l^*})$. This implies trivially $\alpha^{l^*} \models (\alpha^u, \alpha^l)$, and by Lemma 9 it implies in turn $\alpha^{l^*} \models (\alpha^{u*}, \alpha^{l^*})$. □

Proof (Lemma 12) The case of $\perp_{AC}$ is trivial. When $\mathbb{C}(\alpha^u, \alpha^l) \neq \perp_{AC}$, using the notation $(\alpha^{u*}, \alpha^{l^*}) = \mathbb{C}(\alpha^u, \alpha^l)$, we have to prove that

$$\alpha^{l^*} = \alpha^{l^*} \oslash \alpha^{u*} \quad \text{and} \quad \alpha^{u*} = \alpha^{u*} \,\overline{\oslash}\, \alpha^{l^*}$$

We prove that $\alpha^{l^*} = \alpha^{l^*} \oslash \alpha^{u*}$, the other equation could be proved similarly. This is equivalent to prove that $\forall t \geq 0, s \geq t, \alpha^{l^*}(s) - \alpha^{l^*}(t) \leq \alpha^{u*}(s-t)$ which is actually implied by $\alpha^{l^*} \models (\alpha^{u*}, \alpha^{l^*})$, itself guaranteed by Lemma 11. □

5.2.2 Key Theorems About Causality Closure

We now have the necessary prerequisites to introduce important results. In particular, Theorem 7 states that $\mathbb{C}(\overline{\alpha^u}, \underline{\alpha^l})$ is a causal pair of curves, equivalent to (α^u, α^l) when this pair of curves is well-defined. Intuitively, this means that removing forbidden regions on SA-SA curves, once, does not create new forbidden regions. On the example of Figure 6, page 22, this means that, since the curve (b) is SA-SA, (c) has no forbidden regions anymore. The theorem motivates Definition 15 where we introduce $\mathbb{C}(\overline{\alpha^u}, \underline{\alpha^l})$ as the *causality closure* of (α^u, α^l).

Theorem 7 *For any pair of arrival curves* (α^u, α^l),
- $\mathbb{C}(\overline{\alpha^u}, \underline{\alpha^l}) = \perp_{AC} \iff (\alpha^u, \alpha^l)$ *is not satisfiable;*
- $\mathbb{C}(\overline{\alpha^u}, \underline{\alpha^l})$ *is causal, SA-SA and equivalent to* (α^u, α^l)*, otherwise.*

Proof Firstly, Lemma 2 (stating that (α^u, α^l) and $(\overline{\alpha^u}, \underline{\alpha^l})$ are equivalent) and Lemma 9 (stating that $\mathbb{C}\left(\underline{\alpha^l}, \overline{\alpha^u}\right) = \bot_{AC} \implies (\overline{\alpha^u}, \underline{\alpha^l})$ is not satisfiable) give the first direction of the equivalence.

Conversely, let us assume that $(\overline{\alpha^u}^*, \underline{\alpha^l}^*) \overset{\text{def}}{=} \mathbb{C}(\overline{\alpha^u}, \underline{\alpha^l}) \neq \bot_{AC}$. By Lemma 11, $\underline{\alpha^l}^* \models \mathbb{C}(\overline{\alpha^u}, \underline{\alpha^l})$. By Lemma 9 ($(\overline{\alpha^u}, \underline{\alpha^l})$ and $\mathbb{C}(\overline{\alpha^u}, \underline{\alpha^l})$ are equivalent), we also have $\underline{\alpha^l}^* \models (\overline{\alpha^u}, \underline{\alpha^l})$. Since $(\overline{\alpha^u}, \underline{\alpha^l}) \leq_{AC} (\alpha^u, \alpha^l)$, this also implies $\underline{\alpha^l}^* \models (\alpha^u, \alpha^l)$ which is thus satisfiable.

Secondly, assume that (α^u, α^l) is satisfiable (*i.e.*, $\mathbb{C}(\overline{\alpha^u}, \underline{\alpha^l}) \neq \bot_{AC}$), then $(\overline{\alpha^u}, \underline{\alpha^l})$ is SA-SA; therefore, Lemma 12 applies. This gives the hypothesis for Lemma 4, which ensures causality. Lemma 10 ensures the SA-SA property and Lemma 9 the equivalence with (α^u, α^l). □

Definition 15 (Causality Closure) Let (α^u, α^l) be a pair of arrival curves. The causality closure of (α^u, α^l) is the pair of curves obtained by $\mathbb{C}(\overline{\alpha^u}, \underline{\alpha^l})$.

Theorem 8 (Optimality of Causality Closure) *For any pair of arrival curves (α^u, α^l), if (α^u, α^l) is satisfiable, then $\mathbb{C}(\overline{\alpha^u}, \underline{\alpha^l})$ is the tightest pair of curves equivalent to (α^u, α^l).*

By *tightest*, we mean the smallest w.r.t. $\leq_{AC}$; *i.e.*, $\mathbb{C}(\overline{\alpha^u}, \underline{\alpha^l})$ is made of the smallest (resp. the greatest) curve for the upper (resp. lower) part such that the properties are satisfied.

Proof We note $\mathbb{C}(\overline{\alpha^u}, \underline{\alpha^l}) = (\overline{\alpha^u}^*, \underline{\alpha^l}^*)$. Lemma 11 states that $\underline{\alpha^l}^* \models (\overline{\alpha^u}^*, \underline{\alpha^l}^*)$ and $\overline{\alpha^u}^* \models (\overline{\alpha^u}^*, \underline{\alpha^l}^*)$.

Any pair of curves equivalent to (α^u, α^l) would therefore have to accept $\underline{\alpha^l}^*$ and $\overline{\alpha^u}^*$. $(\overline{\alpha^u}^*, \underline{\alpha^l}^*)$ would, hence, be tighter than any such pair of curves. □

These last two theorems provide an interesting result: given any pair of curves, one can compute $\mathbb{C}(\overline{\alpha^u}, \underline{\alpha^l})$, and get either the information that the curves are not satisfiable, or the best possible pair of curves equivalent to the original one. In addition to this optimality, one also gets the desirable properties: causality and SA-SA. This result is implementable on top of any algorithmic toolbox implementing the basic operators: deconvolutions, sub-additive and super-additive closure [32, 6].

Theorem 8 also provides the existence and uniqueness of a tightest pair of curves equivalent to a given one. As a result, the following theorem also holds:

Theorem 9 *Let (α^u, α^l) be a pair of curves. If (α^u, α^l) is the tightest pair of curves representing a set of cumulative curves, then (α^u, α^l) is causal.*

A corollary follows:

Theorem 10 (Uniqueness) *Let (α^u, α^l) be a pair of arrival curves. The causality closure of (α^u, α^l) is the only pair of curves equivalent to (α^u, α^l), causal and SA-SA.*

Proof Any pair of curves causal and SA-SA is a fixed-point of the SA-SA closure and of $\mathbb{C}$, hence a fixed-point of the causality closure. By Theorem 8, it is the tightest pair of curves equivalent to (α^u, α^l) hence it is equal to the causality closure of (α^u, α^l). □

Any computation giving the best possible pair of curves also gives a causal pair of curves. Theorem 9 *explains why*, in practice, most pairs of arrival curves usually manipulated in Real-Time Calculus are causal. Indeed, curves obtained for example by measurements on a real system are causal by construction; furthermore most computations made in the RTC framework compute the optimal solution and thus preserve the causality property. This probably explains why this problem received so little attention up to recently. In practice, for example, simulators are *rarely* used in situation leading to deadlock (but they might be able to). But, with the causality closure presented here, we can now *guarantee* that it *never* happens after applying the causality closure.

On the other side, non-causal pairs of curves may arise whenever a computation is done in an *approximated manner*, even if the approximation is conservative. Next section provides deeper discussions about causality issues.

6 Issues and Solutions About Causality

The causality problem has received surprisingly little attention in the Real-Time Calculus community, although many existing approaches can face it. The goal of this section is first to exhibit from the literature approaches that encounter causality issues and how they treat the problem; second, it explains how to use the causality closure to avoid causality problems.

Usual approaches from RTC aims at computing output curves as functions of input arrival curves. Causality issues can appear at both sides. Some approaches may output non-causal curves (see Section 6.1); in such cases, computing causality closure on the resulting curves provides equivalent but more precise curves. Other approaches or techniques assume causality as input (see Section 6.2).

6.1 Non-Causal Output Curves

Non-causal curves can be produced by non-exact computations on some input arrival curves. Usually, those computations perform *conservative approximations*, to obtain guaranteed results, but even when the input is causal, output curves may be non-causal. This typically occurs when using other tools than RTC algebraic solutions. Abstract interpretation and model-checking algorithms with a timeout, as used in the tool `ac2lus` [1] compute some abstractions, and hence do not guarantee causality of the curves computed. Abstractions can also appear in the model before computation [4] yielding the same problem.

This can also happen when the method computes only part of the points of the resulting arrival curves [31]. In this work, the authors implement an interface between RTC and *Event Count Automata*. The output curves are computed point by point using exact model-checking, but the long-term rate computation uses an approximation which could produce non-causal curves. We give now an example where the procedure described in [31] produces a non-causal pair of curves (see Figure 7 where black curves are inputs). The system we consider is the *identity component* for which the output is equal to its input (*i.e.* the component immediately produces an output event whenever an input event is received); therefore the theoretical output curves are the same as the input ones. The procedure computes first a finite set of points for the beginning of the curve; here, we chose to compute the curves up to $\Delta = 3$. Then, to estimate the long-term rate, the procedure computes one more point with some larger Δ, taken here with value $\Delta = 9$. The resulting curves are made of 8 points (see plots in Figure 7). As one can see, this curve is not causal. Indeed, looking at the super-additive closure of α^l (see dashed red line $\alpha^{l'}$ in Figure 7) the big step between $\alpha^{l'}(8)$ and $\alpha^{l'}(9)$ is greater than $\alpha^u(1)$. Note that computing the causality closure actually removes the forbidden region between α^l and $\alpha^{l'}$ and provides back the exact curve α^l.

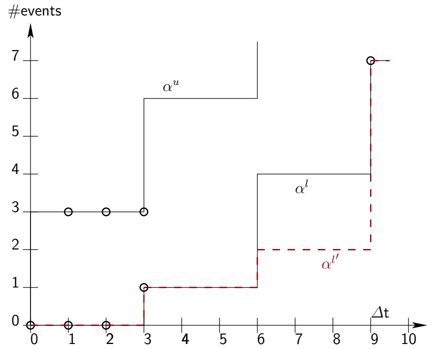


Fig. 7 Example of computation of a non-causal curve using Event Count Automata [31]

On the other hand, the CATS tool [18, 38] relies on exact model-checking, so the tool should always output causal curves. Still, even with exhaustive model-checking, the model-checker may reach a timeout or get out of memory, and CATS would produce a conservative approximation that may not be causal.

Precision Increased. Now, once any approach have computed some output arrival curves, and as far as it may produce non-causal curves, causality closure can be used on those outputs and provide curves which are equivalent and tighter, somehow increasing the precision of the result. As an example, in [4], output curves are computed, one point at a time on an abstract model: this produces non-causal output curves. As a post-treatment, curves are further refined using causality closure to obtain better precision.

6.2 Causal Input Curves

RTC approaches may also face causality issues from their inputs. Indeed, some approaches are not robust to non-causal input curves and require causality to give correct results whereas for some others, non-causal input curves may degrade the analysis of the system. We describe below two kinds of applications where causality issues occur: state-space exploration methods sometimes require causal input curves but often suffer from spurious counter-example when applied on non-causal input curves; whereas event generators may deadlock if the input curves are non-causal.

The work in [22, 36], which interfaces RTC with timed automata, gets rid of causality by constraining the class of curves it considers: input curves are causal by assumption. The extension of this work to arbitrary curves [23] deals with causality using model-checking.

Spurious Couter-Examples. Most model-checking algorithms may produce spurious counter-examples if fed with non-causal input curves. To illustrate the problem, let us assume again the identity component (the output is equal to its input). Again, the output arrival curve (α'^u, α'^l) is equal to the input (α^u, α^l). We set this input to the non-causal pair of curves (α^u, α^l) of Figure 1.(b). Using a verification tool, as done in *e.g.* `ac2lus` [1], we encode the input curve as an observer that tells whether the input stream conforms to the curves at each point in time, and encode the component as a program that takes the observed stream as input. Then, we ask the model checker whether it is safe to set $\alpha'^l(3) = 1$, *i.e.* whether the model-checker can prove that the system will never emit less than 1 event in a window of time of size 3. The model-checker answers: `false property`, and finds the counter-example which corresponds to: "emit nothing on the input for 3 time units". This counter-example is a finite trace which satisfies the input, but does violate the candidate output curve $\alpha'^l(3) = 1$. However, this counter-example cannot be prolonged into an infinite trace, hence is not a real counter-example. Indeed, it is safe to set $\alpha'^l(3) = 1$, but the model-checker failed to prove it because of non-causality. We call *spurious* such a counter-example.

It should be noted that the problem happens even though the model-checker considers infinite curves: the property to prove involves infinite arrival curves and event streams, but a counter-example for this property is a finite trace. The issue is that this finite trace may or may not correspond to the prefix of an infinite one.

It is possible to solve non-causality during the state-space exploration either in the tool itself (see for example the `-causal` option of Lesar [14, 34], which does this for Boolean programs) or with an appropriate temporal logic formula as proposed in [23], but this leads to costly algorithms and is not applicable in all tools. Actually, [23] proposes to identify non-causal curves with model-checking, but does not say what should be done in case the curves are not causal. As opposed to this, causality closure proposed here can be applied *a priori* on curves regardless of the tool being used for the analysis. Causality closure can be computed using cheaper algorithms than model-checking.

One indirect consequence of spurious counter-examples is that since they prevent the tool from returning the exact pair of curves, the output may not be causal anymore. In other words, the non-causality of curves can propagate through components analyzed using model-checking. We give a concrete example where `ac2lus` applied on non-causal input curves produces non-causal curves even using exact model-checking without timeout in Section 8.1. Even though we could not test it because we do not have access to the tool, we believe that the CATS tool [18, 38] has the same issue.

Event Generators. Event generators for RTC [20] are used to build event streams that complies with a given pair of arrival curves. They usually appear in simulation-based approaches [21, 5]. Non-causal curves can make their design likely complicated. Indeed, the usual implementations generate event streams that satisfy the constraints up to some point in time, but doing so may result in deadlocks in the future, namely contradiction between the upper bound and the lower bound. Back to the example of Figure 1.(b), a concrete event generator may chose to generate no event for 3 time units, but would deadlock at time 5. A generator for non-causal curves would therefore have to explore the reachable state-space proactively to make sure no deadlock can occur. This approach is usually costly, due to the well-known explosion state problem; we believe that it is less costly to make the curve causal, *a priori*, with polynomial cost algorithms.

We believe that causality closure is both simpler and more general than any state-space pruning algorithm, such as [5], to avoid the problem.

A generator that does not properly deal with non-causal curves eventually deadlocks after entering a forbidden region. When this happens, the whole execution has to be aborted since event traces generated up to deadlock are actually not valid. Also, since the generator produces only finite traces, there is no easy way to know whether a trace produced by a generator actually corresponds to a valid trace or not: it may not be a prefix of any infinite event trace. Hence, non-causality-aware generators may silently produce invalid event traces.

Extension of RTC. [12] proposes an extension of RTC, Finitary RTC, in order to solve performance issues encountered with traditional RTC. The framework assumes the curves to be causal and relies on the sufficient condition proven in Theorem 4.

6.3 How to Use Causality Closure

As a conclusion for previous section, applying causality closure on inputs, first, allows to get rid of causality problems and therefore to use any method, even if it is not robust to non-causal inputs. Second, computing causality closure on output results allow obtaining the most precise equivalent curves. We detail next how to apply this within a local computation step and then iteratively, in a compositionnal framework.

Local Computation. In general, when an algorithm A computes output curves as a function of input curves ($O = A(I)$), and if the algorithm does not work on non-causal curves I and/or may produce non-causal curves O, we can easily transform this algorithm A into an algorithm A' which accepts non-causal curves and produces only causal curves the following way:

Algorithm 1 (Robust algorithm A' based on A)

```
$I^{causal} \leftarrow \left\{\mathbb{C}(\overline{\alpha^u}, \underline{\alpha^l}), (\overline{\alpha^u}, \underline{\alpha^l}) \in I\right\}$ // Causality closure on inputs
$O^{non\text{-}causal} \leftarrow A(I^{causal})$ // Actual computation
$O^{causal} \leftarrow \left\{\mathbb{C}(\overline{\alpha^u}, \underline{\alpha^l}), (\overline{\alpha^u}, \underline{\alpha^l}) \in O^{non\text{-}causal}\right\}$ // Causality closure on outputs
return $O^{causal}$
```

The tool `ac2lus` [1] actually applies this algorithm: the causality closure is applied before computation, so does not have problems with non-causal input curves, and applies the causality closure operator to the output to possibly increase the precision.

In all the approaches combining RTC and other formalisms cited above, the line $O^{\text{non-causal}} \leftarrow A(I^{\text{causal}})$ is by far the most expensive (in time and memory). Hence, the overhead of the causality closure is small and makes the algorithm more general – since it accepts non-causal curves, and more precise.

Note that there are many ways to improve the precision of some analysis by computing additional constraints, but computing the causality closure of the result will in any case return, at low price, a pair of curves which is at least as precise.

It can be the case though that the causality closure of a pair of curves makes the analysis slower, *i.e.*, computing $A(I^{\text{causal}})$ is slower than computing $A(I)$. On the other hand, the causality closure of a pair of curves can also be simpler and therefore make the analysis faster. If O^{causal} is only used in a context where non-causal curves are acceptable, and if the causality closure yields an unacceptable slowdown of the next analysis, then returning O instead of O^{causal} may be sensible. For an example where a curve can be more complex than its causality closure, see Figure 8. The points marked with a circle are below the causality closure. In the original curve, a machine representation would have to store all of them, while the causality closure of the curve is straightforward ($y = x - 2$).

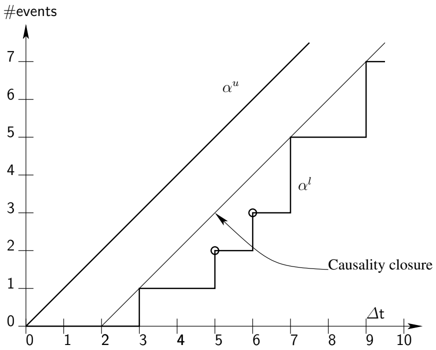


Fig. 8 Example of curves where the causality closure is simpler than the original curves

Composition of Computations. Now, we focus on composition of several RTC analysis. Consider the schema in Figure 9. Assume for example, that the analysis for component A does not preserve causality and that the analysis for component B requires causal input curves. In this case, the output of A, $({\alpha^u}_2, {\alpha^l}_2)$, may not be causal and the analysis for B, which is launched on it, may suffer from the abovementioned issues (see Section 6.2). Replacing the analysis for component A by the robust algorithm A' based on it (Algorithm 1) solves those issues.

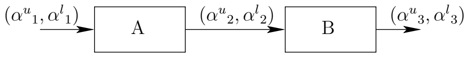

$({\alpha^u}_1, {\alpha^l}_1)$
A
$({\alpha^u}_2, {\alpha^l}_2)$
B
$({\alpha^u}_3, {\alpha^l}_3)$

Fig. 9 Composition of Two Components

7 Causality Closure for Special Classes of Arrival Curves

The previous sections gives a general, theoretical framework to define and compute causality closure. When common min/max-+ operators are applicable, the causality closure can be computed, using Definition 15, as $\mathbb{C}(\overline{\alpha^u}, \underline{\alpha^l})$, where $\mathbb{C}$ is the operator from Definition 14, defined in terms of deconvolutions.

In some classes of curves, however, computing $\mathbb{C}(\overline{\alpha^u}, \underline{\alpha^l})$ directly may not be possible as-is. For example, the causality closure of a *finite discrete* pair of curves (namely, made of a finite set of points) is in general not finite. This section summarizes the existing results on the algorithmic computation of the causality closure for particular classes. We give only an overall view and some intuitions on computations. For formal definitions and proofs, the reader may refer to [28, 29]. Table 1, below, gives an overview on how causality closure can be applied to classes of curves commonly used in RTC.

Class of Curves	**SA-SA Closure**	**Causality Closure**
Pseudo-periodic piecewise affine	Well-known, *e.g.* [39, 32]	Direct computation of $\mathbb{C}(\overline{\alpha^u}, \underline{\alpha^l})$
Convex/concave piecewise affine	By construction *(nothing to do)*	By construction
Finite discrete	Not applicable; Computation of a finite prefix instead. *E.g.* [6]	Not applicable; Computation of a finite prefix instead. Fixed-point iteration of $\mathbb{C}$, [2]
Ultimately piecewise affine ($\mathcal{U}pac$)	SA-SA normal form of curves, [3]	Computation of a dedicated operator on SA-SA normal form of curves, [3]

Table 1 Causality Closure for Common Classes of Curves Used in Real-Time Calculus

Results on Table 1 focus on pseudo-periodic [39] and convex-concave [22] piecewise-affine, finite discrete [38] and ultimately piecewise affine [1] curves. They are further commented in paragraphs 7.1, 7.3 and 7.4. The class of pseudo-periodic piecewise-affine curves is very expressive and commonly used in analytical models like the MPA-RTC toolbox [39]. It is however hard to use when interfacing with state-based formalisms. Indeed, most interfacing approaches restrict to a much stricter class. For example, [22] uses convex/concave piecewise affine curves. An extension to non-convex/concave curves is proposed in [23], but involves complex synchronization of automata, hence a greater algorithmic complexity. [4] and [38] use discrete, finite curves, which are not able to express precisely long-term rates of the streams. [31] can use any ultimately periodic curve to model the input of a component, but the output computed has a periodic part limited to a single entry (long term period), hence, the generality of the model is not exploited.

7.1 Pseudo-Periodic Piecewise Affine Curves

Pseudo-periodic piecewise affine curves are common in RTC. This is the class of curves used by the MPA-RTC framework [39] and the COINC network calculus toolbox [7]. In this class, SA-SA closure and deconvolution operators can be computed using existing algorithms as described in [6]. The causality closure of (α^u, α^l) can then be computed directly as $\mathbb{C}(\overline{\alpha^u}, \underline{\alpha^l})$ (i.e. $(\alpha^u \,\overline{\oslash}\,, \alpha^l \oslash \alpha^u)$ or $\bot_{AC}$, see Definition 14). A discussion on how to implement this in MPA-RTC is given in Section 8.3.

7.2 Convex/Concave Piecewise Affine Curves

Convex/concave piecewise affine curves (for which α^l is convex and α^u is concave) are an interesting class of curves, used for example in [22, 23, 24]. It is a subclass of pseudo-periodic piecewise affine curves, which includes simpler classes like affine curves (sometimes referred to as (σ, ρ)-curves). For a convex/concave piecewise affine curve, α^u (resp. α^l) can be expressed as the minimum (resp. maximum) of a set of affine functions. When reasoning about these curves, the minimum and maximum are naturally translated in conjunction of conditions. These curves are SA-SA by construction. They are also causal and, hence, do not need causality closure operator, as stated by Theorem 11 below.

Theorem 11 *Let $(\alpha^u, \alpha^l) \neq \bot_{AC}$ be a pair of piecewise affine, convex/concave curves. Then (α^u, α^l) is equal to its causality closure.*

Proof Let $a^u\Delta + b^u$ (resp. $a^l\Delta + b^l$) be the last affine piece of α^u (resp. α^l).

A direct consequence of the convex/concave property is that $\forall 0 \leq \Delta_1 \leq \Delta_2$, $\alpha^u(\Delta_2) - \alpha^u(\Delta_1) \geq a^u(\Delta_2 - \Delta_1)$ and $\alpha^l(\Delta_2) - \alpha^l(\Delta_1) \leq a^l(\Delta_2 - \Delta_1)$, since the slope of α^u (resp. α^l) keeps decreasing (resp. increasing) until it reaches a^u (resp. a^l). In particular, $\forall \Delta \geq 0, \alpha^u(\Delta) \geq a^u\Delta$ and $\alpha^l(\Delta) \leq a^l\Delta$.

Then, if we set $(\alpha^{u*}, \alpha^{l^*}) = \mathbb{C}(\alpha^u, \alpha^l)$,

$$\forall \Delta \geq 0, \forall t \geq 0, \quad \alpha^u(\Delta + t) - \alpha^l(t) \geq \alpha^u(\Delta) + a^u t - a^l t$$

$$\alpha^{u*}(\Delta) = \inf_{t \geq 0}\{\alpha^u(\Delta + t) - \alpha^l(t)\} \geq \inf_{t \geq 0}\{\alpha^u(\Delta) + t(\underbrace{a^u - a^l}_{\geq 0 \text{ if } (\alpha^u, \alpha^l) \neq \bot_{AC}})\} = \alpha^u(\Delta)$$

$$\alpha^{u*}(\Delta) \geq \alpha^u(\Delta)$$

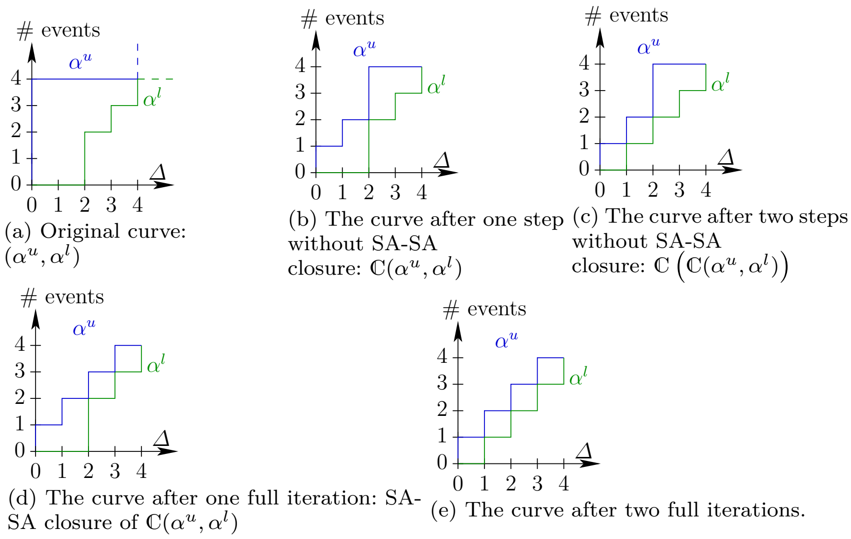
$$\alpha^{u*}(\Delta) = \alpha^u(\Delta) \qquad \text{(since } \alpha^{u*}(\Delta) \leq \alpha^u(\Delta) \text{ too)}$$

Using symmetric arguments, we can show that $\forall \Delta \geq 0$, $\alpha^{l^*}(\Delta) = \alpha^l(\Delta)$. □

7.3 Finite Discrete Curves

Finite discrete curves are defined with a finite set of points; up to those points, say up to T, $\alpha^u(t)$ is set to $+\infty$ and $\alpha^l(t)$ to its last value $\alpha^l(T)$. From an algorithmic point of view, each curve can be represented as an array of values. These curves are used by *e.g.* [1]. Unfortunately, this class of curves is not closed under SA-SA closure, *i.e.*, $(\overline{\alpha^u}, \underline{\alpha^l})$ is not, in general, a finite discrete curve. In this case, the causality closure $\mathbb{C}(\overline{\alpha^u}, \underline{\alpha^l})$ is not finite discrete either. But, as for SA-SA closure, we can compute a finite prefix with interesting properties. The solution consists in iterating the operator $\mathbb{C}$ until a fixed-point is reached; we further detail the algorithm next. More details can be found in [2].

Notice first, that the exact definition of causality closure, $\mathbb{C}(\overline{\alpha^u}, \underline{\alpha^l})$, is helpless from the algorithmic point of view for finite discrete curves since it relies on $(\overline{\alpha^u}, \underline{\alpha^l})$, which is a pair of infinite curves. Therefore, we manage to compute a finite prefix of $\mathbb{C}(\overline{\alpha^u}, \underline{\alpha^l})$ which is equivalent to (α^u, α^l), causal and SA-SA. To construct a causal and SA-SA pair of curves, we use Theorem 4 which states that any fix-point of $\mathbb{C}$ is causal and Lemma 9 which ensures that applying $\mathbb{C}$ does not change the set of



(a) Original curve: (α^u, α^l)

(b) The curve after one step without SA-SA closure: $\mathbb{C}(\alpha^u, \alpha^l)$

(c) The curve after two steps without SA-SA closure: $\mathbb{C}\left(\mathbb{C}(\alpha^u, \alpha^l)\right)$

(d) The curve after one full iteration: SA-SA closure of $\mathbb{C}(\alpha^u, \alpha^l)$

(e) The curve after two full iterations.

Fig. 10 Step by step causality closure for finite curves

accepted cumulative curves. So, given the fact that we know how to compute $\mathbb{C}$, the proposed algorithm applies repeatedly until a fix-point is reached, and the result then is causal. In addition, we apply SA-SA closure at each iteration to speed up convergence and ensure SA-SA property on the result.

The pair of curves α^*, computed by iterating $\mathbb{C}$ on the SA-SA closure of α, is a finite representative of the causality closure $\mathbb{C}(\alpha^u, \alpha^l)$ of α. α^* is causal and SA-SA, and its SA-SA closure is equal to $\mathbb{C}(\alpha^u, \alpha^l)$: the SA-SA closure of α^* is equivalent to α^* hence to $\mathbb{C}(\alpha^u, \alpha^l)$ (by Theorem 7). Since α^* is causal, so is its SA-SA closure (again, by Theorem 7). Theorem 10 implies that the SA-SA closure of α^* is the causality closure of α. In other words, α^* can be seen as a machine representation of $\mathbb{C}(\overline{\alpha^u}, \underline{\alpha^l})$: although it is finite, it contains all the information to recover any point of $\mathbb{C}(\overline{\alpha^u}, \underline{\alpha^l})$.

We illustrate the process with an example in Figure 10. The original pair of curves is (a), which is already SA-SA on $[0, 4]$ (but clearly not SA-SA because of the curve α^u with $+\infty$ values). One application of $\mathbb{C}$ is not sufficient: the curve (b) is not even SA-SA on interval $[0, 4]$, and still has forbidden regions. We iterate the $\mathbb{C}$ operator once more and get (c), which is causal, but not SA-SA.

As the result is causal, one could be satisfied but this is not the finite prefix of the causality closure as announced. To obtain it, a finite prefix of the SA-SA closure is computed, each time, before applying $\mathbb{C}$ again. In the example, this gives curves (d) and then (e) by applying $\mathbb{C}$ again. Then, neither SA-SA closure nor $\mathbb{C}$ would change the curve anymore since we reached a fix-point. In this case, the final curves are the finite prefix on $[0, 4]$ of the causality closure.

We still need a way to compute $\mathbb{C}$ efficiently. The definition of $\mathbb{C}$ contains the supremum of an infinite set, which, as it is, would not be computable. Fortunately, operator $\mathbb{C}$ applied to finite restrictions of curves is indeed much simpler. Since $\forall t > T$, $\alpha^u(t) = +\infty$ and $\alpha^l(t) = \alpha^l(T)$ (where T is the last date at which some value is defined), the values of (α^u, α^l) beyond T do not have to be taken into account when computing the deconvolutions. So, $\mathbb{C}$ can be easily computed quadratically on finite discrete curves.

The detailed proof appears in [28]. It is made simple by the fact that we work in discrete time and events. In particular, this makes the set of possible curves finite: since the computed curves become tighter and tighter, their sequence has to reach a fix-point in a finite number of steps.

The full algorithm for computing the causal and SA-SA pair of curves equivalent to the finite pair of arrival curves A_0 is given in Algorithm 2.

Algorithm 2 (Causal (SA-SA) Curves for Finite Discrete Curves)

```
$A \leftarrow A_0$
repeat
    $A \leftarrow$ SA-SA-closure($A$) // Not mandatory, but speeds up convergence,
                                  // and ensures SA-SA property of the result
    $A' \leftarrow A$
    $A \leftarrow \mathbb{C}(A)$
until $A = \bot_{AC}$ or $A' = A$
```

The loop terminates, since the sequence is decreasing and there is a finite number of possible curves tighter that the original one, but finding a bound on the number of iterations (other than brute-force counting possible tighter curves) is still an open question. In practice, however, the number of iterations is low (one or two in most of the examples we tried, and up to 5 in tricky corner-cases, see Section 8.2).

After the loop, A is either $\bot_{AC}$ or a causal pair of finite discrete curves; it is equivalent to A_0, the original pair of curves; and it is a finite prefix of some SA-SA curves if the SA-SA finite closure was applied (first line within the loop). In this case, it is a finite prefix of the causality closure, i.e., the best pair of curves equivalent to the original A_0.

7.4 Ultimately Piecewise Affine Curves

Ultimately piecewise affine curves ($\mathcal{U}pac$) are a combination of finite discrete curves from [38] and convex/concave curves of [22]. This class of curves is the one used in the tool `ac2lus` [1]. It was chosen to be both expressive and adapted to interfacing other formalisms. A $\mathcal{U}pac$ pair of curves (α^u, α^l) is described, for α^u (resp. α^l), with a prefix given by a finite set of points followed by a convex (resp. concave) piecewise affine curve given by a finite set of affine pieces. α^u (resp. α^l) is then defined as the minimum (resp. maximum) between all points and affine pieces. The $\mathcal{U}pac$ class allows a precise and possibly non-convex/concave description of the initial portion of curves, as well as a set of constraints on the long-term rate of the event stream; it may be easily machine-representable: the finite portion is basically an array and each affine piece is encoded with its slope and its Y-intercept.

Figure 11 shows an example: the upper part is made of 3 points and two affine pieces; the lower part, 3 points, one affine piece.

7.4.1 Normal Form in $\mathcal{U}pac$

To compute causality closure on $\mathcal{U}pac$ curves, the difficulty comes from the points of α^u and α^l, which can interact with each other, or with affine pieces of the other

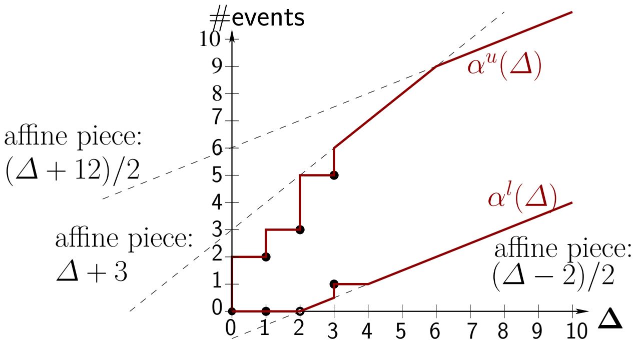


Fig. 11 Example of a *Upac* pair of curves

curve. In particular, arbitrary curves with points and affine pieces are not necessarily SA-SA. We first transform the curves into ones that obey a few well-formedness properties called *normal form*. Properties on normal form then allows, in general, to apply operator $\mathbb{C}$ to get the causality closure.

Causality closure is trivial when curves have only affine pieces, as expressed in Theorem 11. Curves with only points are not SA-SA, and could not be made so algorithmically, since their SA-SA closure has an infinite number of points (this is the case dealt with in Section 7.3). This difficulty can be eliminated thanks to the piecewise affine part of the curves, when it exists. When we apply SA-SA closure to the points of the curves, only a finite number of points remains under or above the affine pieces. If this is not the case, this means that affine pieces add no information w.r.t. the set of points and can be removed: we say in this case that affine pieces are *not relevant*. Removing non-relevant affine pieces or making the finite prefix SA-SA up to crossing some affine piece is called *normalization* of the curves (refer to [3] for details).

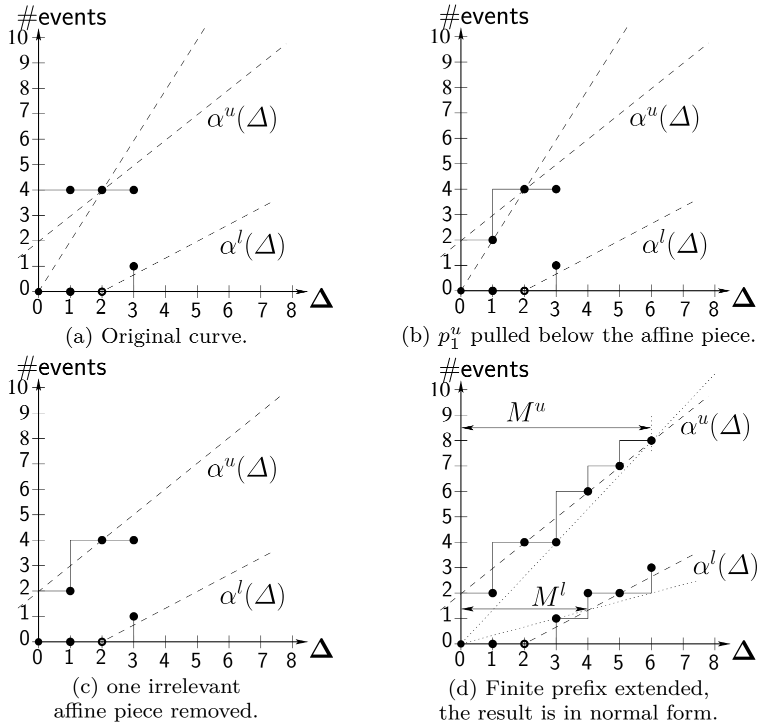
Precisely, a pair of curves in normal form has an SA-SA finite prefix with as many points on the upper and lower curves, followed by a set of relevant affine pieces. In the common case, normalization computes a date M such that all points after M are dominated by some affine piece, and extends the finite prefix by SA-SA closure up to M. This ensures that the result is SA-SA. The formal definition of the normal form [3] includes special cases when α^l and/or α^u has no relevant affine piece, or $(\alpha^u, \alpha^l) = \bot_{AC}$.

The transformation of a pair of curves into normal form is illustrated by Figure 12 (common case where both curves have relevant affine pieces). It essentially consists in adding explicit points from the SA-SA closures to the curve until one can be sure all points are above the affine pieces.

7.4.2 $\mathbb{C}$ for Upac Curves

If none of the curve have affine piece after applying normalization, then the normalized curves are basically finite discrete curves, and we simply apply Algorithm 2 from Section 7.3.

When curves in normal form have either α^l, α^u or both with relevant affine pieces, it can be shown that considering only a finite set of points in the infimum



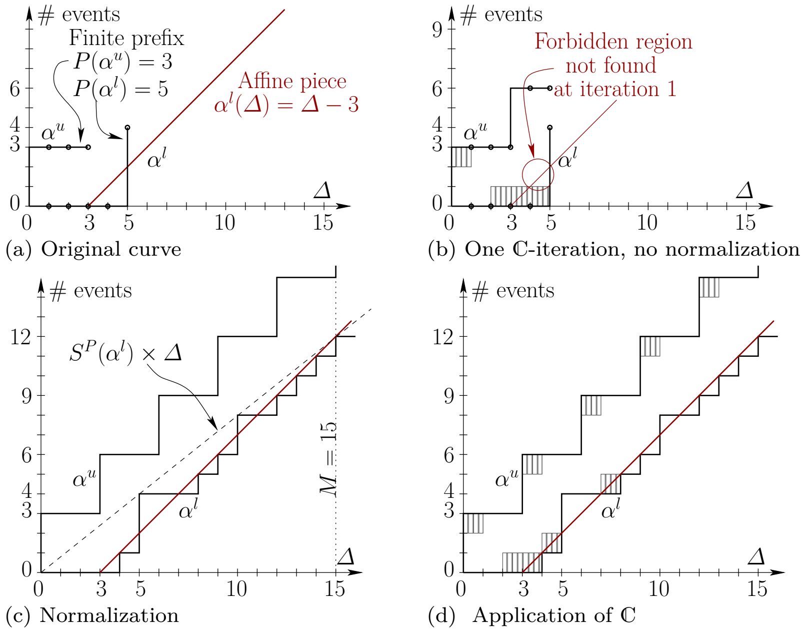
(a) Original curve.

(b) p_1^u pulled below the affine piece.

(c) one irrelevant affine piece removed.

(d) Finite prefix extended, the result is in normal form.

Fig. 12 Step by step transformation into normal form



(a) Original curve

(b) One $\mathbb{C}$-iteration, no normalization

(c) Normalization

(d) Application of $\mathbb{C}$

Fig. 13 Causality Closure on a $\mathcal{U}pac$ Curve With One Affine Piece

and supremum defining the operator $\mathbb{C}$ is sufficient (see [3] for results and proofs). This provides an algorithm which is given again in [3] and further illustrated in Figure 13 which details the whole causality closure algorithm on an example. The pair of curves is given in Figure 13.(a): α^u has no affine piece and α^l has one. Figure 13.(b) shows an attempt to use operator $\mathbb{C}$ on the curves without performing normalization. Since the curves are not SA-SA, $\mathbb{C}$ is able to remove *some* forbidden regions but misses one (the point $\alpha^l(4) = 2$). On the other hand, the normalization algorithm (13.(c)) adds some points to the prefix of the curves, and applying $\mathbb{C}$ on the result yields a causal pair of curves, without further iteration (13.(d)).

8 Experimental Evaluation

We now show experimental results obtained with causality. We show the impact of using causality closure when performing a model-checking based analysis, measure the performance of our implementation on discrete, finite curves, and study the applicability in the MPA-RTC framework. All experiments were performed on a Intel(R) Core(TM) i3-2120 CPU @ 3.30GHz machine with 8 Gb RAM.

8.1 Using Causality Closure in a Model-Checking Based Analysis

To demonstrate the benefits of causality closure, we consider an analysis using the tool `ac2lus` [1] which implements the operator and allows to analyze Lustre program using model-checking. We first illustrate causality closure on a didactic example, and then on a more realistic case study.

8.1.1 Illustration on a Didactic Example

We consider, here, a very simple system. The input curve is given by $\alpha^l(0) = \alpha^l(1) = \ldots = \alpha^l(4) = 0$, $\alpha^l(5) = 4$, and $\alpha^u(0) = 0$, $\alpha^u(1) = \alpha^u(2) = \alpha^u(3) = 3$ (used previously on Figure 1.(b)), and is non-causal. For simplification, we write $\alpha^l = 0, 0, 0, 0, 0, 4$ and $\alpha^u = 0, 3, 3, 3$ and use this notation in the sequel. The processing component is infinitely fast (it transmits input events on its output immediately). We aim at computing a valid output arrival curve (α'^u, α'^l). By definition, $(\alpha'^u, \alpha'^l) = (\alpha^u, \alpha^l)$ is a valid solution. Obviously, the causality closure of (α^u, α^l), given by $\alpha^{l*} = 0, 0, 1, 1, 2, 4, \ldots$ and $\alpha^{u*} = 0, 2, 3, 3, \ldots$, is also a valid solution, and it is the tightest one.

We run the analysis with `ac2lus` using different strategies to check how close we can get to this theoretical result, in practice. All experiments use the model-checker Kind [13], with a timeout of 10 seconds for each individual call to `kind`. `ac2lus` individually computes each points of the output curves; here, we compute 10 points. For each point, the tool proposes a guessed value using binary search; then the guess is validated using `kind` on a so-called observer (roughly, the encoding of the property that the point is correct). We experiment two strategies at the output side: `det_observer` uses a deterministic observer (it returns false whenever the property is violated) and `nondet_observer` uses a non-deterministic observer (if the property is violated, then there exists an execution of the observer which returns false).

Both strategies are equivalent in theory, but the model-checker may fail to prove or disprove the property from one strategy but not from the other.

The result of the experiment is given in Table 2. For clarity, values which are looser than in the optimal curve are underlined. First line shows the default behavior of `ac2lus` which runs causality closure on the input pair of curves, and then runs the analysis. As one can see, the time spent during causality closure is negligible (< 1 millisecond) in comparison with the total analysis time. Actually, applying causality closure even reduces the analysis time in this case: comparing first and second lines (resp. third and fourth) of Table 2, we see that not applying causality closure saves 0.2 (resp. 0.09) milliseconds in the one hand, but leads to an analysis which is 8.8 (resp. 1.7) seconds slower in the other hand. In first and second lines, the end result is the same and corresponds to the optimal output curves.

Not applying causality closure on inputs (second and fourth lines) lead to suboptimal values in the output curves like $\alpha^{l'}(2) = 0$. The suboptimality is not due to approximations or timeouts in the model-checker: in all cases of Table 2, `kind` was either able to prove the property or to disprove it with a concrete counter-example. Values like $\alpha^{l'}(2) = 0$ come from spurious counter-examples as explained in Section 6.2. Indeed, when `ac2lus` checks whether $\alpha^{l'}(1) = 1$ is a valid point, the sequence of input 0, 0 is a valid counter-example (since `ac2lus` checks the property "at all instants, if the input is correct then the output is correct", rather than "if at all instant the input is correct, then at all instant the output is correct"). It does not violate the input curve, but does violate an output curve with $\alpha^{l'}(1) = 1$, hence $\alpha^{l'}(2) = 0$ is the best value at this point. If we apply causality closure on the output pair of curves, the precision loss can be recovered, as in second line.

Nevertheless, fourth line shows a case where the loss of precision cannot be recovered after the fact. Using a non-deterministic observer, the model checker is not able to compute the optimal lower curve (regardless of causality closure) in practice. As a result, the curve $\alpha^{l'}$ does not allow recovering the precision lost in the computation of $\alpha^{u\prime}$, and even the causality closure of the output curves still has $\alpha^{u\prime}(1) = 3$ while the optimal curve has $\alpha^{u\prime}(1) = 2$.

As a conclusion for this small example, not applying causality closure on the input curves does not result in incorrect curves, but yields suboptimal curves even on simple cases like this one. Applying causality closure, we compute tighter results, and faster.

8.1.2 Analysis of a Scheduler Model

We now consider a two-inputs, two-outputs system scheduled with a fixed-priority scheduler modeled in Lustre, taken from [1], itself inspired from the case study in [22] from which we removed the first CPU. The first input (I_1) is processed with highest priority and each processed event produces an event on the first output (O_1). The second input I_2 is processed with lower priority events to O_2. The service curve for the whole system is defined by $\beta^u = 0, 3, 4$ and $\beta^l = 0, 1, 4$. I_1 is defined by the pair of curves $\alpha_1^u = 0, 3, 3, 3, 6, 6$ and $\alpha_1^l = 0, 0, 0, 0, 0, 4$, and I_2 by the pair $\alpha_2^u = 0, 3, 3, 3, 3, 3, 3$ and $\alpha_2^l = 0, 0, 0, 0, 0, 0, 1$. All curves are causal except (α_1^u, α_1^l).

The results are given in Table 3. The analysis computes 8 points for the output service curves β' and for the output arrival curves of O_1 and O_2, but for clarity, only the results for O_2 are shown in the table. Values are underlined when one of the

Causality closure on inputs	Strategy	Causality closure time	Analysis Time	Output (α'^u, α'^l)	Prefix of $\mathbb{C}(\overline{\alpha'^u}, \underline{\alpha'^l})$
Yes	`det_observer`	0.2 ms	35.0 s	$\alpha'^u = 0, 2, 3, 3, 5, 6, 6, 8, 9, 9, 11$ $\alpha'^l = 0, 0, 1, 1, 2, 4, 4, 5, 5, 6, 8$	same as (α'^u, α'^l)
No	`det_observer`	0	43.8 s	$\alpha'^u = 0, \underline{3}, 3, 3, \underline{6}, 6, 6, 8, 9, 9, 11$ $\alpha'^l = 0, 0, \underline{0}, \underline{0}, \underline{0}, 4, 4, 5, 5, 6, 8$	$\alpha^{u*} = 0, 2, 3, 3, 5, 6, 6, 8, 9, 9, 11$ $\alpha^{l*} = 0, 0, 1, 1, 2, 4, 4, 5, 5, 6, 8$
Yes	`nondet_observer`	0.09 ms	62.8 s	$\alpha'^u = 0, 2, 3, 3, 5, 6, 6, 8, 9, 9, 11$ $\alpha'^l = 0, 0, \underline{0}, 1, \underline{1}, \underline{2}, 4, 4, 5, 5, 6$	same as (α'^u, α'^l)
No	`nondet_observer`	0	64.5 s	$\alpha'^u = 0, \underline{3}, 3, 3, \underline{6}, 6, 6, 8, 9, 9, 11$ $\alpha'^l = 0, 0, \underline{0}, \underline{0}, \underline{0}, 2, 4, 4, 5, 5, 6$	$\alpha^{u*} = 0, \underline{3}, 3, 3, \underline{6}, 6, 6, 8, 9, 9, 11$ $\alpha^{l*} = 0, 0, \underline{0}, 1, \underline{1}, \underline{2}, 4, 4, 5, 5, 6$

Table 2 Experimental Results on the Trivial Transformer

Causality closure on inputs	Strategy	Causality closure time	Analysis Time	Output $(\alpha_2'^u, \alpha_2'^l)$	Prefix of $\mathbb{C}(\alpha_2'^u, \alpha_2'^l)$
Yes	`det_observer`	0.17 ms	4 min 49 s	$\alpha_2'^u = 0, 2, 2, 3, 3, 4, 5, 5, 5$ $\alpha_2'^l = 0, 0, 0, 0, 1, 1, 1, 2, 2$	same as $(\alpha_2'^u, \alpha_2'^l)$
No	`det_observer`	0	4 min 31 s	$\alpha_2'^u = 0, \underline{3}, \underline{3}, 3, 3, 4, 5, \underline{6}, \underline{6}$ $\alpha_2'^l = 0, 0, 0, 0, \underline{0}, \underline{0}, \underline{0}, \underline{1}, \underline{1}$	same as $(\alpha_2'^u, \alpha_2'^l)$
Yes	`nondet_observer`	0.12 ms	14 min 41 s	$\alpha_2'^u = 0, \underline{3}, \underline{3}, 3, 3, 4, 5, \underline{6}, \underline{6}$ $\alpha_2'^l = 0, 0, 0, 0, \underline{0}, \underline{0}, \underline{0}, \underline{0}, \underline{1}$	same as $(\alpha_2'^u, \alpha_2'^l)$
No	`nondet_observer`	0	16 min 52 s	$\alpha_2'^u = 0, \underline{3}, \underline{3}, 3, 3, 4, 5, \underline{6}, \underline{6}$ $\alpha_2'^l = 0, 0, 0, 0, \underline{0}, \underline{0}, \underline{0}, \underline{0}, \underline{1}$	same as $(\alpha_2'^u, \alpha_2'^l)$

Table 3 Experimental Results on a Fixed-Priority Scheduler

other computations provides a tighter result. As in the former section, we ran four experiments, depending on the strategy used for the output observer and whether causality closure is applied on I_1. Again, the analysis using a deterministic observer and applying causality closure on inputs provides strictly better results than all other analyses.

In this example, the output curves for O_2 provided by all four analysis are causal (hence applying causality closure on the output cannot provide tighter curves). Nevertheless, $\alpha_2'^u(1) = 2$ (see first line of Table 3) is only obtained when applying causality closure on input I_1, prior to the analysis. It should be noted that non-causality of the first input results in imprecision on the second output, even though they are not directly connected. Indeed, internally the fixed-priority scheduler model computes the remaining service after the first computation and uses it on the second stream. This example shows that non-causality can propagate within a model, and again, the precision can not necessarily be recovered after the analysis.

In this example, the computation times for causal and non-causal curves are similar, and the time taken to apply causality closure is always negligible compared to the analysis.

8.2 Performance of Causality Closure when Scaling Up

In all the examples above, the time taken by causality closure is lower than 1 millisecond, hence negligible. The memory usage is never a problem since the computation is done in-place. This section analyses the performance of causality closure on large curves.

As explained in Section 7.3, we do not have a formal bound for the number of iterations needed in the case of discrete, finite curves. The worst number of iterations we ever observed was obtained for pairs of curves with the following shape: $\alpha^u = 0, a, \ldots (a \text{ times}) \ldots, a$ and $\alpha^l = 0, 0, \ldots (b \text{ times}) \ldots, 0, b, \ldots (a - b \text{ times}) \ldots, b, a$, with $a > b > 0$. Intuitively, those curves are composed of large "steps" that introduce smaller ones at each iteration. We tested all possible pairs (a, b) for $a, b \in \{1, ..., 1001\}$. The worst-case was reached for $a = 1001$ and $b = 569$. In this case, the algorithm needs 5 iterations, each iteration consisting in an SA-SA closure followed by an application of the forbidden region removal operator $\mathbb{C}$.

The application of causality closure on this curve is shown on Figure 14. The computation takes 1.6 seconds in total. This is not a theoretical worst-case, but a very pessimistic one with really large curves, while the performance is still very good.

8.3 Causality Closure in MPA-RTC

The MPA-RTC [32] toolbox provides $\oslash$ and $\overline{\oslash}$ operators (respectively called `rtcmindeconv` and `rtcmaxdeconv` in the toolbox), but unfortunately not sub-additivity and super-additivity closure operators. Note that this is not a theoretical limitation since, there exist algorithms to compute them [6]. Hence, applying causality closure on SA-SA pairs of curves in MPA-RTC is trivial, as far as the result is not empty:

```
function [l_causal, u_causal] = causality_closure(u, l)
l_causal = rtcmindeconv(l, u);
u_causal = rtcmaxdeconv(u, l);
```

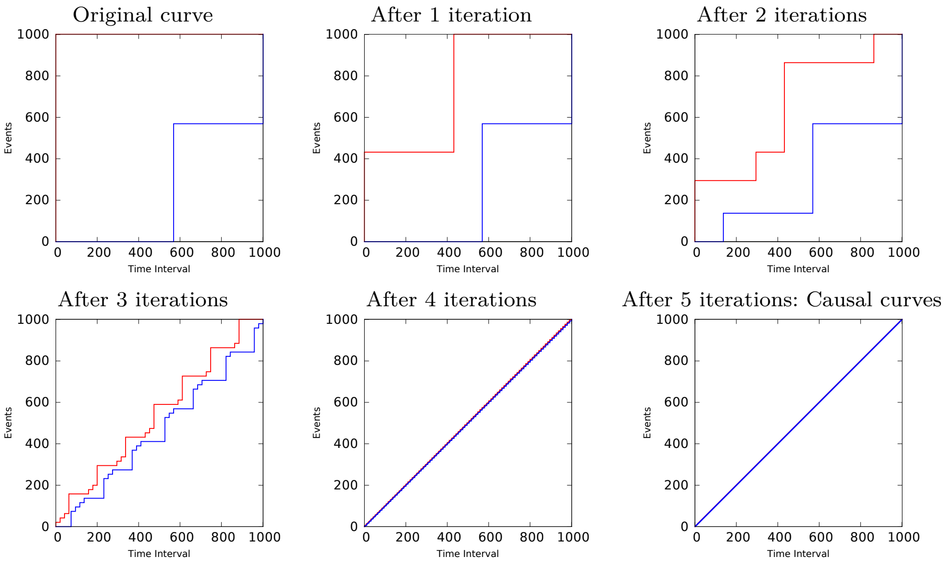


Fig. 14 Iterative Causality Closure on Large Arrival Curves

However, when dealing with non-SA-SA curves, this approach cannot be used in practice. Instead, MPA-RTC provides an operator called `rtctighten` which uses an iterative approach as we did in Section 7.3 for discrete, finite curves. This operator applies $(\alpha'^u, \alpha'^l) = (\alpha^u \otimes \alpha^u) \wedge (\alpha^u \overline{\oslash} \alpha^l), (\alpha^l \overline{\otimes} \alpha^l) \wedge (\alpha^l \oslash \alpha^u)$ N times. On some curves, the iteration can reach a fixed-point. As a fixed-point of `rtctighten` is also a fixed-point of $\mathbb{C}$, it is causal by Theorem 4. Furthermore, its upper curve is a fixed-point of $\otimes$ and its lower curve of $\overline{\otimes}$, hence it is SA-SA. As a consequence, a fixed-point of `rtctighten` is equal to the causality closure $\mathbb{C}(\overline{\alpha^u}, \underline{\alpha^l})$ of (α^u, α^l).

However, iterating `rtctighten` does not always reach a fixed-point; it may happen instead that each iteration doubles the length of the non-periodic part of the curve, resulting in an exponential (in N) computation time as well as an output curve whose length is also exponential (in N).

We illustrate this on the following curves:

```
l_base = rtccurve([[0 0 0]; [5 4 0]]);
u_base = rtccurve([[0 3 0]; [3 inf 0]]);
u_base_alt = rtccurve([[0 3 0]; [3 6 1]]);

l = rtccurve([[0 0 0]; [5 4 0]], 0, 5, 4);
u = rtccurve([0 3 0], 3);
```

where `u_base` represents the finite upper curve such that $\alpha^u(3) = 3$ and $\alpha^u(t) = +\infty$ for $t > 3$; and `l_base` represents the finite lower curve such that $\alpha^l(5) = 4$ and $\alpha^l(t) = 4$ for $t > 5$. As the MPA-RTC operator `rtcmaxdeconv` does not handle infinite values, we replace `u_base` with `u_base_alt` which is equivalent (`u_base` is extended with $\alpha^u(t) = t + 3$ for $t > 3$).

First, we compute `causality_closure(u_base_alt, l_base)` which lasts 6 milliseconds, but since (`u_base`, `l_base`) is not SA-SA, the result is neither optimal nor causal.

Second, we compute `rtctighten([u_base_alt, l_base], N)`. But, we could not reach a fixed-point with any N-value we tried. Instead, we could observe an exponential behavior: for instance, for $N = 10, 11, 12, 13$, the computation lasts respectively 2.4 seconds, 10.1 seconds, 42.8 seconds and 179.5 seconds, and the 4 calls return different pairs of curves. The output curves have non-periodic parts of length respectively 5120, 10240, 20480 and 40960, *i.e.* exactly 5×2^N (5 is the length of the non-periodic prefix of `[u_base_alt, l_base]`).

Now, we focus on the pair of curves (`u`, `l`) (periodic extension of (`u_base`, `l_base`): it is the SA-SA closure of (`u_base`, `l_base`), computed by hand. As expected, `causality_closure(u, l)` returns the causality closure of (`u`, `l`), which is also a fixed-point of `rtctighten`. The call to `causality_closure` takes 9 milliseconds. Similarly, a call to `rtctighten([u, l], 1)` lasts 12 milliseconds and returns a causal curve.

9 Conclusion

In this paper, we formally define the notion of causality for RTC curves, and set up a formal framework to study it. As already mentioned, and although all along the paper we talk about arrival curves, the results are applicable to arrival curves *as well as* to service curves.

To the best of our knowledge, the phenomenon has received little attention and no work has been carried out on the subject yet except [2, 3] and to some extent [5, 10]. This is mainly due to the usual way arrival curves are used within RTC frameworks (which do not produce non-causal curves) on the one hand and to the restrictions of the studies to some causal-by-definition class of arrival curves on the other hand.

One starting point is the intuitive notion of forbidden region, from which we derive a formal definition of causality based on the possibility to extend a curve. We state a sufficient condition for causality on SA-SA pairs of curves (this condition is also necessary for a wide class of curves).

We detail under which conditions causality may appear and be problematic. When using simulators or formal verification tools, causal pairs of curves are very often mandatory (unless involving, if at all possible, heavyweight computations).

We believe that causality should be considered as a basic well-formedness property like the sub-additive/super-additive closure, as done in [12]. Most pairs of curves in practice are already causal, and the ones that are not can easily be made so with the causality closure algorithm in most, if not all, interesting classes of curves.

The definition of causality and causality closure are needed for several approaches. For example, Finitary Real-Time Calculus [12] explicitly requires causal curves. The tool ac2lus [1] would yield spurious counter-examples hence return sub-optimal results if not applied on causal inputs. The connection to timed automata presented in [4] applies causality closure on its output to improve precision. The one presented in [23] has to deal with causality when applied on non convex/concave curves. Random generation of event streams [5, 20] needs to solve the causality problem too. Still, the proofs for the main theorems and algorithms about causality were never published. The main goal of this paper is to consolidate the theoretical foundations by providing detailed formal proofs for all these results.

To avoid non-causal curves, we proposed an algorithm that turns a non-causal pair of curves into an equivalent and causal one. After application of this algorithm, event generators based on arrival curves can no longer deadlock, and formal verifiers no longer produce spurious counter-examples linked to causality. As an additional benefit, the transformation gives the tightest pair of curves equivalent to the original one. It is also a canonical representative of all pairs of arrival curves, defining the same set of event streams.

The theory was developed for discrete and fluid event model, discrete and continuous time for infinite curves. Although one of the results is still a conjecture in the most general case (fluid event model, continuous time), other theorems still hold. In particular, every results about causality closure, and the algorithms derived from them are valid regardless of the conjecture.

Given any class of curves and framework implementing the basic operators ($\oslash$, $\overline{\oslash}$ and SA-SA closure), one can implement causality closure as a simple combination of those operators. Algorithms have also been adapted to discrete time and event model for the case of finite arrival curves, where the sub-additive and super-additive closure operators do not make sense. We also briefly present the case of concave/convex piecewise affine curves, which does not have the problem at all, and a combination of finite discrete curves with this model called $\mathcal{U}pac$, which also requires some adaptation of the general algorithm. These specific algorithms are implemented in the `ac2lus` [1] toolbox.

References

1. Altisen K, Moy M (2010) ac2lus: Bringing SMT-solving and abstract interpretation techniques to real-time calculus through the synchronous language Lustre. In: ECRTS, Brussels, Belgium, URL `http://www-verimag.imag.fr/~moy/publications/ac2lus-conf.pdf`
2. Altisen K, Moy M (2010) Arrival curves for real-time calculus: the causality problem and its solutions. In: TACAS
3. Altisen K, Moy M (2011) Causality closure for a new class of curves in real-time calculus. In: Proceedings of the 1st International Workshop on Worst-Case Traversal Time, ACM, Vienna, Autriche, pp 3–10, DOI 10.1145/2071589.2071590, URL `http://www-verimag.imag.fr/~moy/publications/wctt2011.pdf`
4. Altisen K, Liu Y, Moy M (2010) Performance evaluation of components using a granularity-based interface between real-time calculus and timed automata. In: QAPL, URL `http://www-verimag.imag.fr/~moy/publications/gran-paper.pdf`
5. Banerjee K, Dasgupta P (2014) Acceptance and random generation of event sequences under real time calculus constraints. In: Proceedings of the conference on Design, Automation & Test in Europe, European Design and Automation Association, p 254
6. Bouillard A, Thierry É (2008) An algorithmic toolbox for network calculus. Discrete Event Dynamic Systems 18(1):3–49

7. Bouillard A, Cottenceau B, Gaujal B, Hardouin L, Lagrange S, Lhommeau M, Thierry E (2009) COINC Library: A toolbox for Network Calculus. In: Fourth International Conference on Performance Evaluation Methodologies and Tools, Valuetools, Pisa, Italy, URL `https://hal.inria.fr/hal-00788929`
8. Bouillard A, Jouhet L, Thierry E (2009) Service curves in Network Calculus: dos and don'ts. Research Report RR-7094, INRIA, URL `http://hal.inria.fr/inria-00431674/en/`
9. Chakraborty S, Künzli S, Thiele L (2003) A general framework for analysing system properties in platform-based embedded system designs. In: DATE, Citeseer, vol 3, p 10190
10. Ghosh S, Dasgupta P (2015) Formal methods for pattern based reliability analysis in embedded systems. In: VLSI Design (VLSID), 2015 28th International Conference on, IEEE, pp 192–197
11. Giannopoulou G, Lampka K, Stoimenov N, Thiele L (2012) Timed model checking with abstractions: towards worst-case response time analysis in resource-sharing manycore systems. In: Proceedings of the tenth ACM international conference on Embedded software, ACM, pp 63–72
12. Guan N, Yi W (2013) Finitary real-time calculus: Efficient performance analysis of distributed embedded systems. In: Real-Time Systems Symposium (RTSS), 2013 IEEE 34th, IEEE, pp 330–339
13. Hagen G, Tinelli C (2008) Scaling up the formal verification of Lustre programs with SMT-based techniques. In: FMCAD
14. Halbwachs N, Lagnier F, Ratel C (1992) Programming and verifying critical systems by means of the synchronous data-flow programming language LUSTRE. Transactions on Software Engineering
15. Henzinger TA, Nicollin X, Sifakis J, Yovine S (1992) Symbolic model checking for real-time systems. Information and Computation 111:394–406
16. Jeannet B (2003) Dynamic partitioning in linear relation analysis. application to the verification of reactive systems. Formal Methods in System Design 23(1):5–37
17. Jonsson B, Perathoner S, Thiele L, Yi W (2008) Cyclic dependencies in modular performance analysis. In: EMSOFT, DOI http://doi.acm.org/10.1145/1450058.1450083
18. Krcál P, Mokrushin L, Yi W (2007) A tool for compositional analysis of timed systems by abstraction. In: Proceedings of 19th Nordic workshop on programming theory (NWPT07)
19. Kumar P, Goswami D, Chakraborty S, Annaswamy A, Lampka K, Thiele L (2012) A hybrid approach to cyber-physical systems verification. In: Proceedings of the 49th Annual Design Automation Conference, ACM, pp 688–696
20. Künzli S, Thiele L (2006) Generating event traces based on arrival curves. In: MMB
21. Künzli S, Poletti F, Benini L, Thiele L (2006) Combining simulation and formal methods for system-level performance analysis. In: DATE, 3001 Leuven, Belgium, Belgium, pp 236–241
22. Lampka K, Perathoner S, Thiele L (2009) Analytic real-time analysis and timed automata: A hybrid method for analyzing embedded real-time systems. In: EMSOFT
23. Lampka K, Perathoner S, Thiele L (2010) Analytic real-time analysis and timed automata: a hybrid methodology for the performance analysis of embedded real-time systems. Design Automation for Embedded Systems pp 1–35

24. Lampka K, Perathoner S, Thiele L (2013) Component-based system design: analytic real-time interfaces for state-based component implementations. International Journal on Software Tools for Technology Transfer 15(3):155–170
25. Le Boudec JY, Thiran P (2001) Network Calculus. Springer Verlag, URL `http://infoscience.epfl.ch/getfile.py?recid=282&mode=best`
26. Lin CW, Di Natale M, Zeng H, Phan LTX, Sangiovanni-Vincentelli A (2013) Timing analysis of process graphs with finite communication buffers. In: Real-Time and Embedded Technology and Applications Symposium (RTAS), 2013 IEEE 19th, pp 227–236
27. Liu CL, Layland JW (1973) Scheduling algorithms for multiprogramming in a hard-real-time environment. J ACM 20(1):46–61
28. Moy M, Altisen K (2009) Arrival curves for real-time calculus: the causality problem and its solutions. Tech. Rep. TR-2009-15, Verimag
29. Moy M, Altisen K (2011) Causality closure for a new class of curves in real-time calculus full version. Tech. Rep. TR-2011-13, Verimag Research Report
30. Perathoner S, Lampka K, Thiele L (2011) Composing heterogeneous components for system-wide performance analysis. In: DATE, IEEE, pp 842–847
31. Phan LT, Chakraborty S, Thiagarajan P, Thiele L (2007) Composing functional and state-based performance models for analyzing heterogeneous real-time systems. In: Real-Time Systems Symposium (RTSS), IEEE Computer Society, Los Alamitos, CA, USA, pp 343–352, DOI http://doi.ieeecomputersociety.org/10.1109/RTSS.2007.46
32. QComputer Engineering and Networks Laboratory (TIK), ETH Zurich, Switzerland (2008) Modular performance analysis with real-time calculus. Software Toolbox, URL `http://www.mpa.ethz.ch/`
33. Raymond P (1991) Compilation efficace d'un langage declaratif synchrone: Le generateur de code Lustre-v3. PhD thesis, Institut National Polytechnique de Grenoble - INPG, section 13.7, "Causalité" (pages 119–123)
34. Raymond P (2000) Lustre v4 Manual. Verimag
35. Schranzhofer A, Pellizzoni R, Chen JJ, Thiele L, Caccamo M (2011) Timing analysis for resource access interference on adaptive resource arbiters. In: Real-Time and Embedded Technology and Applications Symposium (RTAS), 2011 17th IEEE, IEEE, pp 213–222
36. Simalatsar A, Ramadian Y, Lampka K, Perathoner S, Passerone R, Thiele L (2011) Enabling parametric feasibility analysis in real-time calculus driven performance evaluation. In: Proceedings of the 2011 International Conference on Compilers, Architecture, and Synthesis for Embedded Systems, CASES 2011, ACM, Taipei, Taiwan, pp 155–164
37. Thiele L, Chakraborty S, Naedele M (2000) Real-time calculus for scheduling hard real-time systems. In: ISCAS
38. Uppsala University (2007) Cats tool. URL `http://www.timestool.com/cats`
39. Wandeler E (2006) Modular performance analysis and interface-based design for embedded real-time systems. PhD thesis, PhD Thesis ETH Zurich