Arrival Curves for Real-Time Calculus: the Causality Problem and its Solutions

Matthieu Moy and Karine Altisen

Verimag
Centre Équation - 2, avenue de Vignate 38610 Gières - FRANCE

Abstract. The Real-Time Calculus (RTC) [16] 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. System properties may then be computed using algebraic techniques in a compositional way. A well-known limitation of RTC is that it cannot model systems with states and recent works [7, 1, 13, 11] 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 [14]: it can be the case that, once having generated a finite prefix of an event stream, the generator deadlocks, since no extension of the prefix can satisfy the curves anymore. When trying to express the property of the curves with state-based models, one may face the same problem. This paper formally defines the problem on arrival curves, and gives algebraic ways to characterize causal pairs of curves, i.e. curves for which the problem cannot occur. Then, we provide algorithms to compute a causal pair of curves equivalent to a given curve, in several models. These algorithms provide a canonical representation for 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 that help taking or validating decisions on the conception of a system as early as possible in the design process.

There exists many methods among which we can distinguish two families. *Computational* approaches study fine-grain models of the system to represent its complete behavior. The validation of the system using such a model may involve simulation, testing and verification. As opposed, *analytical* techniques, such as Real Time Scheduling (founded with [9]) and Real Time Calculus [16], use purely analytical models, based on mathematical equations that can be solved efficiently. These models can represent in a simple way the amount of events to be processed and how fast they can be processed. Solving these equations can give, for example, the best and worst cases for performances.

Both families of approaches have their advantages and drawbacks. Simulating precisely an embedded system gives very precise results, but only for one simulation, and one instance of a system. Analytical approaches, on the other hand, give strict worst case execution times, and usually give results very fast, but do so only for cases that the theory can take into account. For example, Real-Time Calculus cannot handle the notion of state in the modeling of a system. Recent studies try to combine the approaches to take the best of both [7, 3, 10]. The work we present in this paper fully takes its root and motivation in one of those studies, while trying to combine Real-Time Calculus, state-based models and abstract interpretation, using synchronous languages [1].

The *Real-Time Calculus (RTC)* [16] 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 can be described with curves for its input stream and available resources and some other curves for the outputs. For already-modeled components, RTC gives exact bounds on the output stream of a component as a function of its input stream. This result can then be used as input for the next component. *Arrival curves* are function of relative time that constrains the number of events that can occur in an interval of time. For any sliding window of time of length Δ, the pair of arrival curves (α^u, α^l) gives *explicitly* the lower $\alpha^l(\Delta)$ and upper $\alpha^u(\Delta)$ bounds on the number of events (see examples in Figure 1). But, arrival curves may also contain *implicit constraints* indirectly deduced 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 [6], it is typical to produce a stream of events that satisfies some given arrival curves using a *generator of events*. Such generators are the computational representation of a pair of curves, they are built to generate any streams that satisfies the curves. There are multiple ways to write such generators [6, 10, 1, 3] but many faced the problem. For the explanation, let us consider a straightforward one, in discrete time: it computes at each point in time the lower and upper bounds on the number of events allowed to be emitted, based on the events already emitted, and it emits a random number of events within these bounds. Now, it may happen, due to implicit constraints, that some upper bound is strictly lower than the lower bound, leading the generator to deadlock.

Another case where implicit constraints are problematic is the case of formal verification of a system, with inputs and outputs characterized by arrival curves. One may want to prove a property like "If the input complies with the arrival curve pairs α_I, then the output satisfies the arrival curve pairs α_O". But verification tools based on reachability analysis (see, e.g. [5]) usually allow only the expression of "If the input complies with α_I *up to time* t, then the output complies with the α_O *up to time* t". Then, the tool may find a counter-example violating α_O without violating α_I *up to time* t, but it can be the case that this finite counter-example cannot be extended into an infinite execution that satis-

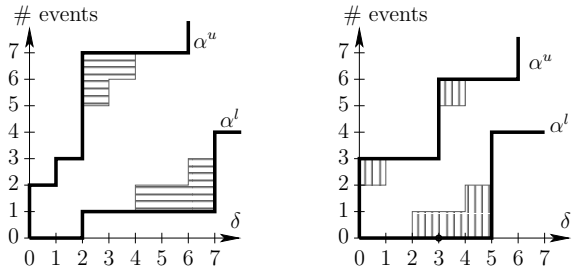
fies α_I. This would therefore be a *spurious counter-example*. Getting rid of these counter-examples sometimes requires heavyweight state exploration techniques (for example, the `-causal` option of `lesar` [15] does this for Boolean programs) but not all tools are able to do it (`nbac` [5] cannot, for example, and the problem is known to be undecidable for integer programs). The technique that translates the constraints of arrival curves into a model to be analyzed by a verifier tool was used for, e.g., timed automata [7, 3], event count automata [13] and synchronous programs [1]. For each tool, one can pose the questions: "what is the behavior of the tool when used on curves with forbidden regions?" and "do the tool output curves with forbidden regions?". Actually, except [1], the papers do not give answer to them. We will see that [7, 3] do not create curves with forbidden regions while [13] could at least in theory, and we explain why. Each of the tools would badly behave in the presence of forbidden regions, and this paper gives a way to get rid of them before using any 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 [8] and the second, which may produce deadlocks in generators and spurious counter-examples in verification is the goal of this paper. Let us discover those using a pair of arrival curves (α^u, α^l) (see Figure 1 for an example).

Firstly, by splitting some 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 specified the bounds $\alpha^l(6)$ and $\alpha^u(6)$ to be 1 and 7, the number of events in a window of size 6 can actually never be equal to 1 ($\alpha^l(6)$). 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 [8] 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*. 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 towards the future. Figure 1.(b) gives an example of such a case: since $\alpha^l(3) = 0$, the lower curve does not give a lower bound on the number of events that can occur in a window of time of size 3, but if an execution has no event during such a window, then the upper curve prevents it from emitting more than 3 events in the next 2 units of time, while the lower curve will force it to emit at least 4. It is therefore impossible to emit no events for 3 units of time. We call the regions that contain such points *forbidden regions*. No execution can cross a forbidden region unless it gets blocked some time latter, due to some contradiction between

lower and upper constraints. Borrowing the vocabulary used in [14], we call this kind of implicit constraints *causality constraints*. A pair of curves for which the beginning of an 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 an implication).



(a) Unreachable Regions (b) Forbidden regions

Fig. 1. Implicit and explicit constraints on arrival curves

Surprisingly, this question has received very little attention and to the best of our knowledge, no transformation has been published 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, a straightforward answer is that they cannot come from a real system, since curves derived from execution or simulation of real systems are always well-formed. The common practice is to use such 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, one uses other tools for deriving output arrival curves, given some input arrival curves. Those tools, among them model-checking of timed automata [4] on abstracted models, abstract interpretation of Lustre programs [5], may compute non-causal arrival curves, even when the input is causal.

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 constraint explicit gives tighter bounds on the number of events (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 set of flows of events [10] using different approximate 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 propose several solutions.

– We give a characterization of the notion of causal pairs of arrival curves.
– Combining this property with existing ones, we give a definition for a *canonical representation* of a pair of curves, which is causal and sub-additive/super-additive. We show that it is also the *tightest possible representation* of the original curve.
– We propose an algorithm that transforms a pair of arrival curves into its equivalent causal representation.

All results in the paper are proven (Due to place limitations, the proofs only appear in [12]) and may be applied to dense-time or discrete-time arrival curves on the one hand, to 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, but we believe it could be adapted to other contexts. Furthermore, although all along the paper we talk about arrival curves, the reader should be convinced that every results also apply to service curves.

The outline of this paper is as follows: Section 2 defines *arrival curves* and some few algebraic operators; Section 3 defines *causality* and gives a *characterization* of it; Section 4 shows how to compute the *tightest causal representation* of arrival curves; and Section 5 gives an *algorithm* for computing it for discrete finite curves.

2 Arrival Curves

We now define the notion of arrival curves that characterize timing properties on a set of event streams. A pair of lower and upper arrival curves defines lower and upper bounds on the number of events allowed in a sliding window of size δ. Event streams that satisfy the pair of arrival curves are abstracted with cumulative curves that represent the number of events that occurs since the beginning $t = 0$. In this paper, we do not focus on a particular model and every results (except Section 5) apply to all of them. Namely, time can be either continuous or discrete, and we consider both the fluid and discrete event-model. Formally, functions we consider are from $\mathcal{T}$, the time, to $\overline{\mathcal{E}} = \mathcal{E} \cup \{\infty\}$, the event count; and $\mathcal{T}$ (resp. $\mathcal{E}$) can be either $\mathcal{R}^+$, the set of non-negative reals, or $\mathcal{N}$, the set of naturals. We note $\mathcal{F}$ the set of wide-sense increasing functions f from $\mathcal{T}$ to $\overline{\mathcal{E}}$ and such that $f(0) = 0$; $\mathcal{F}_{finite}$ is the set of such functions from $\mathcal{T}$ to $\mathcal{E}$.

Definition 1 (Arrival Curves and Cumulative Curves). *$R \in \mathcal{F}_{finite}$ can model a* cumulative curve*: $R(t)$ represents the (finite) amount of events that occurred in the interval of time $[0, t]$.*

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

Let R be a cumulative curve and (α^u, α^l) be a pair of arrival curves. R is said to satisfy *(α^u, α^l) noted $R \models (\alpha^u, \alpha^l)$ iff(def)* $\quad \forall x \in \mathcal{T}, \forall \delta \in \mathcal{T}, \quad R(x + \delta) - R(x) \in [\alpha^l(\delta), \alpha^u(\delta)]$

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

We note $R \models_{\leq T} (\alpha^u, \alpha^l)$*, meaning that* R *satisfies* (α^u, α^l) up to T *iff(def)* $\forall t \leq T, \forall \delta \leq t, \quad R(t) - R(t-\delta) \in [\alpha^l(\delta), \alpha^u(\delta)]$

Next, comes the deconvolution operators that will be intensively used in the next sections. And then we briefly recall the notions of sub-additivity and super-additivity, that are used to erase the *unreachable regions* from the curves. Details on those notions can be find, e.g. in [8].

Definition 2 (Deconvolutions). *Let* f, g *be functions from* $\mathcal{T}$ *to* $\overline{\mathcal{E}}$ *and* $x \in \mathcal{T}$*,*

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

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

Definition 3 (Sub/Super-Additivity and Closures). *Let* $f \in \mathcal{F}$*,* f *is said to be* sub-additive *(resp. super-additive) iff* $\forall s, t \in \mathcal{T}$. $f(t+s) \leq f(t) + f(s)$ *(resp* $f(t+s) \geq f(t) + f(s)$*).*

Let $f \in \mathcal{F}$*. Among all the sub-additive (resp. super-additive) functions* $g \in \mathcal{F}$ *that are smaller (resp. greater) than* f *there exists an upper (resp. lower) bound called the* sub-additive *(resp.* super-additive*)* closure *of* f *and denoted by* $\overline{f}$ *(resp.* $\underline{f}$*). A pair of arrival curves* (α^u, α^l) *is Sub-Additive-Super-Additive (denoted SA-SA for short) iff(def)* α^u *is sub-additive and* α^l *is super-additive. We call* $(\overline{\alpha^u}, \underline{\alpha^l})$ *the SA-SA closure of* (α^u, α^l)*.*

SA-SA closure makes explicit some of the implicit constraints of an arrival curve. It makes explicit the *unreachable regions* (Figure 1.(a)), which are the regions between α^l and its super-additive closure $\underline{\alpha^l}$ in the one side, between α^u and its sub-additive closure $\overline{\alpha^u}$ on the other. Informally, they represents the points between α^u and α^l that are not reachable by any finite or infinite cumulative curves.

3 Causality: Definition and Characterization

We now define the notion of causality. The problem we are studying is the one of an event stream that is correct up to a certain time T, but "can not be continued" without violating the pair of curves. This can be seen as a deadlock of the flow, which could then neither let time elapse nor emit an additional event. A pair of arrival curves for which this problem can not happen is called *causal*. We first give a formal definition for causality, and then give a characterization with algebraic formulas.

Definition 4 (Causal Arrival Curves). *Let* (α^u, α^l) *be a pair of arrival curves.* (α^u, α^l) *is said to be* causal *iff any cumulative curve* R *that satisfies* (α^u, α^l) up to T *can be extended indefinitely into a cumulative curve* R' *that also satisfies* (α^u, α^l)*. In other words,* (α^u, α^l) *is causal iff(def)* $\quad \forall T \geq 0$*,* $\forall R, \big(R \models_{\leq T} (\alpha^u, \alpha^l)\big) \implies \big(\exists R' \mid R' \models (\alpha^u, \alpha^l)$ *and* $\forall t \leq T, R(t) = R'(t)\big)$

Unlike the sub-additivity and super-additivity properties, the 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 since the impossibility to extend a cumulative curve can come only from a contradiction between an upper bound and a lower bound.

3.1 Characterization of Causality

Causality reveals new implicit constraints. 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 try to define α^{l^*}, defined informally as "α^l without its forbidden regions". $\alpha^{l^*}(\delta)$ is the smallest value for which a cumulative curve R verifying $R(t+\delta)-R(t) \geq \alpha^{l^*}(\delta)$ is guaranteed to be extensible infinitely by emitting the maximum amount 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, then R cannot be extended without violating either α^u or α^l, which means that the region below α^{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 the N for which the curves would intersect. Formally, this can be written as $\alpha^{l^*} = \sup_{\delta \geq 0} \left\{\alpha^l(\delta_0 + \delta) - \alpha^u(\delta)\right\}$, which is the definition of the 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.

We can therefore define more formally forbidden region 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 forbidden region. The contribution of this paper is the study of these forbidden regions, giving a formal characterization and algorithms to detect their presence and to eliminate them.

Theorem 1 (Characterization of Causality). *Let (α^u, α^l) be a pair of arrival curves. The following implications and equivalences hold:*

$$\begin{array}{ccc} \alpha^l = \alpha^l \oslash \alpha^u \\ \quad and \\ \alpha^u = \alpha^u \overline{\oslash} \alpha^l \end{array} \quad \overset{\textbf{(e)}}{\Longrightarrow} \quad (\alpha^u, \alpha^l) \textit{ is causal}$$

$$\Downarrow \textbf{(d)} \qquad \nearrow \textbf{(c)} \qquad \Updownarrow \textbf{(b)}$$

$$\begin{array}{ccc} \underline{\alpha^l} = \underline{\alpha^l} \oslash \overline{\alpha^u} \\ \quad and \\ \overline{\alpha^u} = \overline{\alpha^u} \overline{\oslash} \underline{\alpha^l} \end{array} \quad \overset{\textbf{(a)}}{\Longleftrightarrow} \quad (\overline{\alpha^u}, \underline{\alpha^l}) \textit{ is causal}$$

The main result is equivalence **(c)** which gives an algebraic characterization of causality for any pair of arrival curves. Intuitively, it states that a pair of

curves is causal if and only if its SA-SA closure has no forbidden region. A weaker version of this theorem is implication **(e)** which gives only a sufficient condition: a pair of arrival curves having no forbidden region is causal.

One could have expected for the converse to be true, i.e. that a pair of arrival curves is causal implies that it doesn't have forbidden regions. This result is indeed false in general: a pair of causal curves can have forbidden regions if they are included in their unreachable regions. This is shown in the counter-example of Figure 2. The vertically hatched region is a forbidden region, and we do not have $\alpha^l = \alpha^l \oslash \alpha^u$, but the curve is still causal. Actually, the forbidden region is below $\underline{\alpha^l}$, so it is not reachable. The causality implies the absence of forbidden region for SA-SA curves though, since all unreachable regions have been erased from them: this is equivalence **(a)**. The remainders **(b)** and **(d)** are intermediate results.

Indication for the proofs: all the proofs are detailed in [12], most of them being relatively long and technical, using several intermediate lemmas. The following gives the overall structure and the chronology of the proofs.

(a) The proof is completely skipped here due to space limitations.
(b) The proof is relatively straightforward and based on the fact that $(\overline{\alpha^u}, \underline{\alpha^l})$ and (α^u, α^l) accept the same set of cumulative curves.
(c) This characterization is obtained by transitivity of **(a)** and **(b)**.
(d) The proof is omitted due to space limitations.
(e) The sufficient condition is obtained by transitivity of **(c)** and **(d)**.

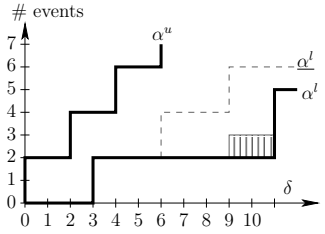


Fig. 2. Causal Curve with a Forbidden Region

4 Computing the 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 $\mathbb{C}$ operator, which removes the forbidden regions from a pair of curves.

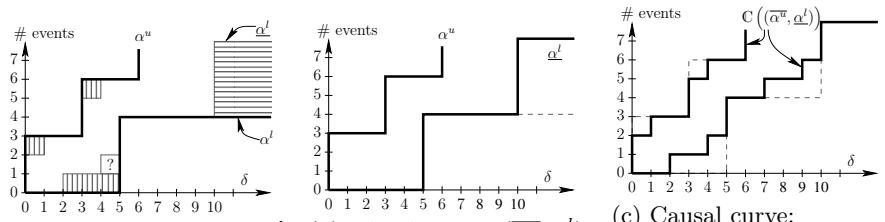
Notice that removing forbidden regions is done on the pair of curves, globally. As a result, while removing the forbidden regions on α^l, one may introduce new ones on α^u and vice-versa. One natural way to solve this issue is to iterate the

forbidden region removal until one reaches the fix-point (assuming it is reached in a finite number of steps, which is not always the case).

To illustrate this, an example is given in Figure 3. The original curve (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 curve (a): if we consider the curves (α^u, α^l) before any transformation, it does not seem to be a forbidden region. An execution emitting only 1 event in 4 units of time seems to be able to continue by emitting 3 events right after. Actually, this is impossible, and there are at least two ways to show it. the first way to remove this "?"-region is to apply the forbidden regions removal 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. After the second iteration of the forbidden region removal, we reached the fix-point, and implication **(e)** guarantees the causality. This iterative approach will be detailed in Section 5.

However, an interesting property of the $\mathbb{C}$ operator is that it does not create new forbidden regions when applied on SA-SA curves: its application on $(\overline{\alpha^u}, \underline{\alpha^l})$ provides the causal canonical representative of (α^u, α^l) (this approach is further discussed in this section). Back to the example in Figure 3, a second way to show that the "?"-region should be removed from α^l 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 units of time. This illustrates the approach followed in this section: we eliminate the forbidden regions with $\mathbb{C}$ (3.(c)) only after performing an SA-SA closure (3.(b)). The iterative approach will be kept for cases where the SA-SA closure cannot be applied due to algorithmic and coding limitations.



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

Fig. 3. Step-by-step causality closure

4.1 Removing Forbidden Regions: the $\mathbb{C}$ Operator

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 $\perp_{AC}$ the set of pairs of functions in $\mathcal{F}$ such that the former constraint is false. To simplify notations, $\perp_{AC}$ 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* $\perp_{AC}$.

Definition 5. *We define the* $\mathbb{C}$ *operator from AC to AC as:*

$$\mathbb{C}\left(\perp_{AC}\right) = \perp_{AC} \quad \textit{and} \quad \mathbb{C}\left(\alpha^l, \alpha^u\right) = \begin{cases} \textit{let } L = \alpha^l \oslash \alpha^u, U = \alpha^u \mathbin{\overline{\oslash}} \alpha^l \\ \textit{if } L \leq U \textit{ then } (L, U) \\ \textit{else } \perp_{AC} \end{cases}$$

When (α^u, α^l) is a pair of arrival curves then $L = \alpha^l \oslash \alpha^u$ and $U = \alpha^u \mathbin{\overline{\oslash}} \alpha^l$ are functions in $\mathcal{F}$ (i.e. wide-sense increasing and equal to zero at zero). But they may cross each other (it may happen that $L \not\leq U$): in these cases, the $\mathbb{C}$ operator computes the value $\perp_{AC}$. This means that the pair of arrival curves was not satisfiable (i.e. no cumulative curve satisfies it).

4.2 $\mathbb{C}(\overline{\alpha^u}, \underline{\alpha^l})$: 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 all the desirable properties: SA-SA, causality, and it is indeed the best possible pair of curves equivalent to (α^u, α^l).

Theorem 2. *For any pair of arrival curves* (α^u, α^l)*,*
- $\mathbb{C}(\overline{\alpha^u}, \underline{\alpha^l}) = \perp_{AC}$ *iff* (α^u, α^l) *is non-satisfiable;*
- $\mathbb{C}(\overline{\alpha^u}, \underline{\alpha^l})$ *is causal, SA-SA and equivalent to* (α^u, α^l)*, otherwise.*
- *when* (α^u, α^l) *is satisfiable,* $\mathbb{C}(\overline{\alpha^u}, \underline{\alpha^l})$ *is the* tightest *pair of curves equivalent to* (α^u, α^l)*.*

By *tightest*, we mean that $\mathbb{C}(\overline{\alpha^u}, \underline{\alpha^l})$ is made of the smallest (resp. the greatest) curve for the upper (resp. lower) part such that the properties are satisfied. The proofs are given in [12]. This gives 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: convolution, deconvolution, sub-additive and super-additive closure.

Theorem 2 also provides the existence and uniqueness of a tightest pair of curves equivalent to a given one. As a result, the following theorem states that it is causal.

Theorem 3. *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.*

Any computation giving the best possible pair of curves also gives a causal pair of curves. Theorem 3 *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 computations made in the RTC framework compute the optimal solution and thus preserve the causality property. It also probably explains why this problem received so little attention up to now.

On the other side, non-causal pairs of curves may arise whenever a computation is done in an *inexact manner*. This typically occurs using other tools than RTC algebraic solutions. Indeed, the recent works that interfaces RTC with state-based models face the problem. In [7], the authors get rid of it by constraining the class of curves they compute which are causal by definition (the extension to arbitrary curves which is part of their future works will have to deal with it though). But, in [10], the output curves are computed, one point at a time on an abstract model: this does result into non causal curves, which are refined after being computed. The CATS tool [3] relies on exact model-checking, so applied on a causal pair of curves, the tool would output causal curves. [13] also uses exact model-checking, but the long-term rate computation uses an approximation, which could generate non-causal curves (see [12] for an example).

Finally, in ac2lus [1] we use the abstract interpreter `nbac` [5], which also does some abstractions, and hence doesn't guarantee the causality of the curves computed.

5 Algorithms for Discrete Finite Curves

5.1 Definitions of Finite Arrival Curves

Up to this point, we dealt with infinite pairs of curves, but, as mentionned in the introduction, the original work that brought us to studying causality was to connect RTC curves to synchronous programming languages [1]. Our model uses a simple computer representation of arrival curves: we work in *discrete-time*, *discrete-event* model, and consider only *finite curves*, which makes them easy to represent and manipulate algorithmically speaking. We consider the infinite extension of the curves to remain in the theoretical framework presented in the previous sections and to be able to apply the same theorems. Therefore, instead of formalizing the notion of *finite* curves, we consider the *restriction* of infinite curves on a *finite* interval.

Working with discrete-time (resp. discrete-event) models doesn't change the above results, since we considered time (resp. event count) as the set $\mathcal{T}$ (resp. $\mathcal{E}$), being either $\mathcal{R}^+$ or $\mathcal{N}$. We now (in this chapter) set $\mathcal{T} = \mathcal{E} = \mathcal{N}$. On the other hand, working with finite curves will change the results a bit: the notion of SA-SA-closure doesn't fit well in the finite model, since the SA-SA-closure of a finite curve could be infinite.

We first give some definitions for finite arrival curves and then an algorithm to efficiently compute the causality closure using the $\mathbb{C}$ operator.

Definition 6 (Finite restriction of an arrival curve). *We denote by* $(\alpha^u\big|_T, \alpha^l\big|_T)$ *the restriction of* (α^u, α^l) to $[0, T]$ *defined as:*

$$\forall t \leq T, \quad \alpha^u\big|_T(t) \stackrel{def}{=} \alpha^u(t) \text{ and } \alpha^l\big|_T(t) \stackrel{def}{=} \alpha^l(t)$$
$$\forall t > T, \quad \alpha^u\big|_T(t) \stackrel{def}{=} +\infty \text{ and } \alpha^l\big|_T(t) \stackrel{def}{=} \alpha^l(T)$$

$(\alpha^u|_{\mathrm{T}}, \alpha^l|_{\mathrm{T}})$ still applies to infinite event streams, but only gives constraints for finite windows of time. Intuitively, it could be a pair of curves defined over $[0, T]$. But defining them as functions over $\mathcal{N}$ has the advantage of remaining within the definition of arrival curves given above: $\alpha^l|_{\mathrm{T}}$ and $\alpha^u|_{\mathrm{T}}$ are still functions in $\mathcal{F}$, but they can be represented easily as finite arrays of naturals.

It should be noted that finite restrictions of arrival curves can not have the SA-SA property (as far as $\exists t > 0.\alpha^u(t) < +\infty$). However, one can define the property *SA-SA over* $[0, T]$ and the associated closure (see [12] for details). Additionally, [2], page 7, provides an efficient way to compute the sub-additive closure in discrete events. It can easily be adapted to compute the SA-SA closure over $[0, T]$ leading to a simple, quadratic algorithm.

5.2 Causality closure for Finite Discrete Curves

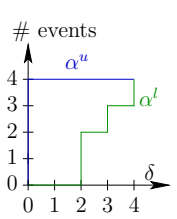
Unfortunately, the valid result for infinite curves, stating that $\mathbb{C}(\overline{\alpha^u}, \underline{\alpha^l})$ was a causal curve equivalent to (α^u, α^l) is helpless from the algorithmic point of view with finite curves: computing it would require computing $(\overline{\alpha^u}, \underline{\alpha^l})$, which is an infinite curve. But theorem 1**(e)** still holds (i.e. the fix-points of $\mathbb{C}$ are causal), and it can easily be shown that applying the $\mathbb{C}$ operator doesn't change the set of accepted cumulative curves. So, we can compute the fix-point by iterating $\mathbb{C}$.

We illustrate the process with an example in Figure 4. The original pair of curves is (a), and one can see that although the curves are 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.

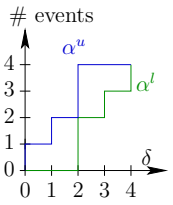
Another option which may speed up the algorithm, is to apply a finite SA-SA closure before applying $\mathbb{C}$ again: this gives curves (d) and then (e) by applying $\mathbb{C}$ again. Then, neither the SA-SA closure nor $\mathbb{C}$ would change the curve anymore: we reached the fix-point. In this case, the final curve has both the causality and the SA-SA properties on interval $[0, 4]$.

Theorem 4. *For any $T > 0$ and any pair of arrival curves (α^u, α^l) with $\forall t \in [0, T], \alpha^u(t) \neq +\infty$, the sequence $\mathbb{C}^n(\alpha^u|_T, \alpha^l|_T)$ admits a fix-point (denoted $\mathbb{C}^\infty(\alpha^u|_T, \alpha^l|_T)$, which is either $\perp_{AC}$ or a causal pair of arrival curves equivalent to $(\alpha^u|_T, \alpha^l|_T)$.*

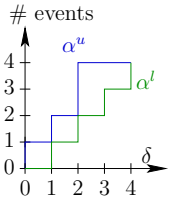
The above theorem states that, given an finite discrete pair of arrival curves, one may iteratively compute, by application of the $\mathbb{C}$ operator, a causal finite discrete pair of arrival curves which is equivalent to the original, if it is satisfiable; otherwise, the computation leads to $\perp_{AC}$. The convergence of the iterations can be accelerated by using, in addition to $\mathbb{C}$, other tightening operators that preserves the set of accepted cumulative curves like the SA-SA closure. This is expressed in the following theorem and applied in the example in Figure 4.(d) and 4.(e).



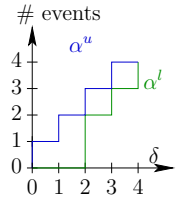
(a) Original curve: (α^u, α^l)



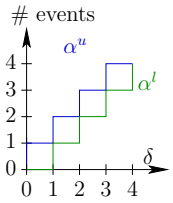
(b) The curve after one step without SA-SA closure: $\mathbb{C}(\alpha^u, \alpha^l)$



(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. 4. Step-by-step causality closure for finite curves

Theorem 5. *For any $T > 0$ and any pair of arrival curves (α^u, α^l) with $\forall t \in [0, T], \alpha^u(t) \neq +\infty$, the sequence defined by $({\alpha^u}_0, {\alpha^l}_0) = (\alpha^u|_T, \alpha^l|_T)$ and $\forall n \geq 0$, $({\alpha^u}_{n+1}, {\alpha^l}_{n+1}) = \mathbb{C}(\overline{{\alpha^u}_n}|_T, \underline{{\alpha^l}_n}|_T)$ admits a fix-point, which is either $\perp_{AC}$ or a causal and SA-SA pair of arrival curves equivalent to $(\alpha^u|_T, \alpha^l|_T)$.*

The detailed proof appears in [12]. It is made simple by the fact that we work in discrete time and events: this makes the set of possible curves finite. Since the sequence $({\alpha^u}_n, {\alpha^l}_n)$ becomes tighter and tighter, it has to reach a fix-point in a finite number of steps.

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, the operator $\mathbb{C}$ applied to finite restrictions of curves is indeed much simpler. Since $\forall t > T$, $\alpha^u(t) = +\infty$ and $\alpha^l(t) = \alpha^l(T)$, the values of (α^u, α^l) beyond T do not have to be taken into account in the computation of the deconvolutions, so $\mathbb{C}$ can be easily computed quadratically.

5.3 Algorithm

The full algorithm for computing the causal and SA-SA pair of curves equivalent to the finite pair of arrival curves A_0 defined on $[0, T]$ is given in Figure 5.

The loop terminates but finding a bound on the number of iterations other than the brute-force (just knowing that the sequence is decreasing and that there is a finite number of possible curves tighter that the original one) is still an open question. In practice, however, the number of iterations required is low (one or two in the examples we tried).

$A \leftarrow A_0$
repeat
  $A \leftarrow$ SA-SA-closure(A) /* Not mandatory, but speeds up convergence, and ensures SA-SA */
  $A' \leftarrow A$
  $A \leftarrow \mathbb{C}(A)$
until $A \neq \perp_{AC}$ or $A' = A$

Fig. 5. Computation of causality closure for finite, discrete curves

After the loop, A is either $\perp_{AC}$ or a causal pair of finite discrete curves; it is equivalent to A_0, the original pair of curves; and it is SA-SA on the interval $[0, T]$ if the SA-SA closure was applied (first line within the loop). In this case, it is the best pair of curves equivalent to the original A_0.

6 Conclusion

We formally defined 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. We started from the intuitive notion of forbidden region, and the definition of causality based on the possibility to extend a curve, and stated the equivalence (valid for SA-SA pairs of curves) between absence of forbidden regions and the definition.

To the best we know, the phenomenon has received little attention and no work has been yet on the subject. This is mainly due to the usual way arrival curves were used within the RTC framework on the one hand and to the restrictions of the studies to some already causal class of arrival curves in the other hand. We detailed in which conditions causality can appear and be problematic. Dealing with general causal pairs of curves in a simulator or a formal verification tool is very often mandatory (unless using, if at all possible, heavyweight roundabout computations). To avoid non-causal curves, we propose an algorithm that turns a non-causal pair of curves into a causal one. After application of this algorithm, event generators based on arrival curves cannot deadlock, and formal verifiers do no more produce spurious counter-examples linked to causality.

The additional benefit of the transformation is that it gives the tightest pair of curves equivalent to the original one, which is also a canonical representative of all arrival curve pairs defining the same set of event streams. Indeed, compared to the "mathematical refinement algorithm" proposed in [10], our algorithm is more general and potentially more precise. It would be an improvement to replace this refinement algorithm by the causality closure.

The theorems and algorithms work for discrete and fluid event model, discrete and continuous time for infinite curves. Given any subset of these models, one just has to implement the basic operators to be able to use them. They 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. The later was implemented in the `ac2lus` [1] toolbox.

References

1. Karine Altisen and Matthieu Moy. Connecting real-time calculus to the synchronous programming language lustre. Technical Report TR-2009-14, Verimag Research Report, 2009.
2. A. Bouillard and É. Thierry. An algorithmic toolbox for network calculus. *Discrete Event Dynamic Systems*, 18(1):3–49, 2008.
3. Uppsala University DARTS, IT Dept. Cats tool, 2007. `http://www.timestool.com/cats`.
4. Thomas A. Henzinger, Xavier Nicollin, Joseph Sifakis, and Sergio Yovine. Symbolic model checking for real-time systems. *Information and Computation*, 111:394–406, 1992.
5. B. Jeannet. Dynamic partitioning in linear relation analysis. application to the verification of reactive systems. *Formal Methods in System Design*, 23(1):5–37, July 2003.
6. Simon Künzli, Francesco Poletti, Luca Benini, and Lothar Thiele. Combining simulation and formal methods for system-level performance analysis. In *DATE '06: Proceedings of the conference on Design, automation and test in Europe*, pages 236–241, 3001 Leuven, Belgium, Belgium, 2006. European Design and Automation Association.
7. Kai Lampka, Simon Perathoner, and Lothar Thiele. Analytic real-time analysis and timed automata: A hybrid method for analyzing embedded real-time systems. In *8th ACM & IEEE International conference on Embedded software, EMSOFT 2009*. ACM, October 2009.
8. Jean-Yves Le Boudec and Patrick Thiran. *Network Calculus*. Lecture Notes in Computer Science (LNCS). Springer Verlag, 2001.
9. C. L. Liu and James W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *J. ACM*, 20(1):46–61, 1973.
10. Yanhong Liu, Karine Altisen, and Matthieu Moy. Granularity-based interfacing between RTC and timed automata performance models. Technical Report TR-2009-10, Verimag, Centre Équation, 38610 Gières, August 2009.
11. Leonid Mokrushin. Compositional analysis of timed systems by abstraction. PowerPoint Slides, 2007.
12. Matthieu Moy and Karine Altisen. Arrival curves for real-time calculus: the causality problem and its solutions. Technical Report TR-2009-15, Verimag Research Report, 2009.
13. Linh T. X. Phan, Samarjit Chakraborty, P. S. Thiagarajan, and Lothar Thiele. Composing functional and state-based performance models for analyzing heterogeneous real-time systems. *Real-Time Systems Symposium, IEEE International*, 0:343–352, 2007.
14. Pascal Raymond. *Compilation efficace d'un langage declaratif synchrone: Le generateur de code Lustre-v3*. PhD thesis, Institut National Polytechnique de Grenoble - INPG, November 1991. Section 13.7, "Causalité" (pages 119–123).
15. Pascal Raymond. *Lustre v4 Manual*. Verimag, February 2000.
16. Lothar Thiele, Samarjit Chakraborty, and Martin Naedele. Real-time calculus for scheduling hard real-time systems. In *International Symposium on Circuits and Systems ISCAS 2000*, volume 4, pages 101–104, Geneva, Switzerland, March 2000.