

Causality closure for a new class of curves in real-time calculus

Karine Altisen, Matthieu Moy

▶ **To cite this version:**

Karine Altisen, Matthieu Moy. Causality closure for a new class of curves in real-time calculus. Proceedings of the 1st International Workshop on Worst-Case Traversal Time, Nov 2011, Vienna, Austria. pp.3–10, 10.1145/2071589.2071590 . hal-00648628

HAL Id: hal-00648628
https://hal.science/hal-00648628

Submitted on 6 Dec 2011

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Causality Closure for a New Class of Curves in Real-Time Calculus

Karine Altisen
Karine.Altisen@imag.fr

Matthieu Moy
Matthieu.Moy@imag.fr

Verimag (UMR CNRS 5104), Grenoble INP
Grenoble, F-38041, France

ABSTRACT

Real-Time Calculus (RTC) [14] is a framework to analyze heterogeneous real-time systems that process event streams of data. The streams are characterized by *arrival curves* which 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.

The property of *causality* on arrival curves essentially characterizes the absence of deadlock in the corresponding generator. A mathematical operation called *causality closure* transforms arbitrary curves into causal ones.

In this paper, we extend the existing theory on causality to the class $\mathcal{U}pac$ of infinite curves represented by a finite set of points plus piecewise affine functions, where existing algorithms did not apply. We show how to apply the causality closure on this class of curves, prove that this causal representative is still in the class and give algorithms to compute it. This provides the tightest pair of curves among the curves which accept the same sets of streams.

Categories and Subject Descriptors

I.1.2 [**Computing Methodologies**]: Symbolic and Algebraic Manipulation—*Algorithms*; D.4.8 [**Computer Systems Organization**]: Performance of Systems—*Modeling Techniques*

General Terms

Performance

Keywords

Real-Time Calculus, Causality, Algorithms

1. INTRODUCTION

Modern embedded system design must deal with more and more constraints in several domains: an ever increasing size, deeply interdependent softwares and hardwares, timing constraints, low power constraints with power management mechanisms included. Furthermore, they have to be manufactured at low cost, and with an increasingly short life-cycle. In this context, it is no longer possible to wait for the physical prototypes to validate the decisions on the design of a system: hardware dimensioning (*e.g.* buffer size) and timing performance should be evaluated at an early stage of the development.

In this work, we focus on the performance evaluation of embedded systems using the *Real-Time Calculus framework (RTC)* [14]. It uses abstract models to analyze heterogeneous systems in a compositional manner. It allows the characterization of streams of events to be treated by a component with curves called *arrival curves*: they provide the worst and best cases on the number of events that may arrive in any window of time of a given length. RTC also uses the notion of service curves, which count available resources instead of events in a similar manner. A given component's interface is described with curves for its input stream and available resources, and some other curves for the outputs. The huge advantage of RTC is its ability, for already-modeled components, to give exact bounds (i.e. both conservative and as tight as possible) on the output stream of a component as a function of its input stream. Unfortunately, complex components can not be modeled in RTC. In particular, RTC models cannot handle state-based components such as power managed systems.

Some works proposed RTC extensions using Event Count Automata [13], Timed Automata [9, 10, 15, 1], and Synchronous Programs [2]; all of those approaches perform the analysis using some formal verification tool (model-checking, abstract interpretation) and they all face the *causality problem* [3] in a way or another. They express the input arrival curves as a non-deterministic generator, or an observer of events that encodes the constraints of the curves. Arrival curves are functions of relative time that constrains the number of events that can occur in an interval of time: for any sliding window of time, the functions *explicitly* express the maximum and minimum number of events. But, arrival curves may also contain *implicit constraints* deduced from explicit ones. It may happen that the corresponding generator of events deadlocks, due to conflicting implicit constraints. Curves exhibiting such contradictions are called non-causal. They can thus lead to deadlocks in generators, but also to spurious counter-examples in formal verification, or simply to suboptimal results. Conversely, they can also be produced by non-exact computations: e.g. when a curve is computed after

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
WCTT '11 November 29 2011, Vienna, UNK, Austria
Copyright 2011 ACM 978-1-4503-1008-6/11/11 ...$10.00.

abstracting the system [1], using abstract interpretation [2], or using a non-complete model-checking algorithm [13]. It is therefore useful in some cases, mandatory in others, to solve this issue by transforming non-causal curves into causal ones.

This *causality* problem has been studied theoretically in [3]: this work provides an algebraic characterization of a causal pair of curves and defines the causality closure which makes the implicit constraints explicit. This operation, as a free side effect, results in the best pair of curves among the equivalent ones.

The present paper provides *algorithms* that extend and apply the results in [3]; it computes the causality closure on infinite curves with a particular shape. We focus on so-called $\mathcal{U}pac$ curves defined by a finite set of points followed by a piecewise convex/concave affine part. This class of curves is interesting since it enables to have a finite and simple computational encoding for the curves whereas curves are infinite. They may include a precise description for short windows of time, description with affine pieces for larger windows and then the long term rate curve. The $\mathcal{U}pac$ class includes (but is strictly larger than) the class of curves used in [9], which cannot express non convex/concave curves. It is the one used in [2] which computes the performance (given as arrival curves) of a system modeled as a synchronous program and uses bounded model-checking and abstract interpretation. In [2], the causality closure is used to prevent the tool from computing spurious counter-examples in the proofs and to increase the precision of the result, but the actual algorithm and the underlying theory that handles causality for $\mathcal{U}pac$ curves were never published. This paper fills this gap with the following contributions:

- We identify a normal form on $\mathcal{U}pac$ curves, on which the causality closure can be computed easily; we prove that the causal representatives of such curves are still in the class;
- We provide an algorithm that transforms a pair of curves of the class into normal form and then computes the causality closure, i.e. the causal representative of the curves.

The algorithms are all in polynomial time (quadratic) and have been implemented in `ac2lus`. The results of the paper are all proved (due to space limitations, full proofs only appear in the extended version of this paper [12], but the intuition is given here). Furthermore, although all along the paper we talk about arrival curves, the reader should be convinced that the results and the algorithms also apply to service curves in Real-Time Calculus, and strict service curves in network calculus [4], which behave the same.

The outline of this paper is as follows: Section 2 summarizes the Real-Time Calculus background and recalls the notion of causality from [3]; Section 3 defines the above class of curves and provides the first contribution of this paper: a normal form and a normalization algorithm. Section 4 gives the second contribution: an algorithm for the causality closure on those curves.

2. CAUSALITY IN REAL-TIME CALCULUS: MOTIVATION AND FORMALIZATION

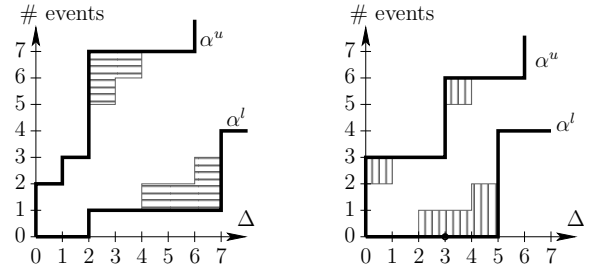
2.1 Real-Time Calculus Curves

The Real-Time Calculus focuses on components that process events; it uses curves to characterize the streams of events a component may compute. Event streams are abstracted with *cumulative curves* that represent the number of events that occurred since the origin of time $t = 0$. *Arrival curves* characterize the timing properties of a set of event streams using their cumulative curves. A pair of lower and upper arrival curves defines lower and upper bounds on the number of events allowed in a sliding window of Δ unit of time. Since the goal of the paper is to define algorithms, we need the curves to be *machine-representable*. We consider the discrete-time fluid-event model where the time is discrete, but the event counters are rational-numbers. Notice that the definitions and characterization of the causality are nevertheless model-independent.

Formally, the cumulative curves and the arrival curves are functions from $\mathcal{N}$ (time) to $\mathcal{Q}^+$ (number of events) where $\mathcal{N}$ is the set of naturals and $\mathcal{Q}^+$ is the set of non-negative rationals. We note $\overline{\mathbf{C}}$ the set of wide-sense increasing curves c from $\mathcal{N}$ to $\overline{\mathcal{Q}^+} = \mathcal{Q}^+ \cup \{\infty\}$ and such that $c(0) = 0$; $\mathbf{C}$ is the set of such curves from $\mathcal{N}$ to $\mathcal{Q}^+$ (no infinity value). The order on curves is point-wise.

Definition 1 (Cumulative Curves, Arrival Curves). *Functions $R \in \mathbf{C}$ are called* cumulative curves*: $R(t)$ 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 $\overline{\mathbf{C}} \times \mathbf{C}$, 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 $\forall x \in \mathcal{N}, \forall \Delta \in \mathcal{N}, \quad R(x + \Delta) - R(x) \in [\alpha^l(\Delta), \alpha^u(\Delta)]$. We say that a pair of arrival curves (α^u, α^l) is* satisfiable *iff there exists a cumulative curve R that satisfies (α^u, α^l).*



(a) Unreachable Regions (b) Forbidden regions

Figure 1: Implicit constraints on arrival curves

The fact that arrival curves are defined on relative time (windows of size Δ) involves many implicit constraints. Among those constraints, some may be tighter than the constraints explicitly given by the curves. In [3], two kinds of implicit constraints were informally distinguished (see Figure 1). The first ones (as in the above explanation) remove regions called *unreachable regions* (see Figure 1(a)): no (finite nor infinite) cumulative curve can cross through those regions. The removal is done using the sub and super additive closures of the curves which are briefly recalled here, but this result is well-known and detailed in *e.g.* [11]. For example, splitting a window of size Δ into two windows of size Δ_1 and Δ_2 (s.t. $\Delta_1 + \Delta_2 = \Delta$) may reveal a greater bound $\alpha^l(\Delta_1) + \alpha^l(\Delta_2)$ for $\alpha^l(\Delta)$. This removes some regions from the curve and leads to an equivalent but tighter pair of arrival curves.

Definition 2 (Sub and Super Additive Closures). *Let $\alpha \in \overline{\mathbf{C}}$. α is said to be* sub-additive *(resp. super-additive) iff $\forall s, t \in \mathcal{N} \ . \ \alpha(t + s) \leq \alpha(t) + \alpha(s)$ (resp*

$\alpha(t+s) \geq \alpha(t) + \alpha(s)$). *Among all the sub-additive (resp. super-additive) curves that are smaller (resp. greater) than* α *there exists an upper (resp. lower) bound called the* sub-additive *(resp.* super-additive*)* closure *of* α *and denoted by* $\overline{\alpha}$ *(resp.* $\underline{\alpha}$*).*

A pair of arrival curves (α^u, α^l) *is sub-additive and super-additive (denoted SA-SA for short) iff* α^u *is sub-additive and* α^l *is super-additive. We note* $(\overline{\alpha^u}, \underline{\alpha^l})$ *the SA-SA closure of* (α^u, α^l).

The SA-SA closure makes explicit the *unreachable regions* of the curves: they 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.

2.2 Causality and the Causality Closure

The second kind of constraints removes the *forbidden regions*: no execution can cross a forbidden region without being blocked some time latter by some contradiction between lower and upper constraints (for example, being forced to emit more than 2 events but less than 1 which is not possible). In other words, there cannot exist an infinite cumulative curve that crosses through such a region (see Figure 1(b)). It may occur due to the forbidden regions of some implicit constraints that an event stream (represented by a cumulative curve) satisfies a pair of arrival curves up to a certain time T, but then deadlocks since no time can elapse and no event can be emitted without violating the arrival curves. A pair of arrival curves for which this problem cannot happen is called *causal*.

Issues and Solutions with Non-Causal Curves. The causality problem has received surprisingly little attention in the Real-Time Calculus community, although many of the existing approaches which connect Real-Time Calculus to other formalisms (e.g. [9, 8, 13]) did produce, or have problems with non-causal pairs of curves.

Indeed, non-causal curves can be produced by non-exact algorithms, that is, whenever a conservative approximation is performed in a computation. This is the case when using techniques like abstract interpretation, model-checking algorithms with a timeout like in [2], or when abstractions are performed on the model before the computation [1]. It can also happen when an algorithm computes only part of the points of an arrival curve, like done in [13].

When an algorithm produces non-causal curves, the causality closure provides another pair of curve which is equivalent and tighter, somehow increasing the precision of the result. See [1] for an example where such a post-treatment of the curve leads to a better precision.

In addition to their sub-optimality, non-causal curves can be a real issue if used as input of some algorithms. Most model-checking algorithms may produce spurious counter-examples if fed with non-causal hypothesis. It is possible to solve the non-causality during the state-space exploration either in the tool itself (see for example the `-causal` option of Lesar [6]) or with an appropriate temporal logic formula as proposed in [10], but this leads to costly algorithms, and is not applicable with any tool (for example, Nbac, used in [2] cannot do it). As opposed to this, the causality closure can be applied *a priori* on the curves regardless of the tool being used for the analysis, and with much cheaper algorithms.

Non-causal curves would also greatly complexify the design of an event generator for simulation-based approaches [7]: the usual implementations generate event streams that satisfy the constraints up to the current point in time, but doing so may result in deadlocks in the future (contradiction between the upper-bound and the lower-bound). A generator for non-causal curves would therefore have to explore the reachable state-space in the future to make sure a deadlock will not occur. Again, it is much simpler, and cheaper algorithmically, to make the curves causal *a priori*.

In summary, 1) when computing output curves as functions of input curves, applying the causality closure on inputs allows to get rid of the causality problem and therefore to use any method, even if it is not robust to non-causal inputs. And 2) applying the causality closure on the output may increase the precision of the result, providing the best equivalent output curves.

Characterization of Causality. The following paragraph recalls the formal definition of causality and the main results from [3] which are used in this paper.

DEFINITION 3 (CAUSAL ARRIVAL CURVES). *Let* (α^u, α^l) *be a pair of arrival curves.* (α^u, α^l) *is said to be* causal *iff for any* $T \in \mathcal{N}$*, any cumulative curve* R *that satisfies* (α^u, α^l) up to T *can be extended forever into a cumulative curve* R' *that also satisfies* (α^u, α^l)*. In other words,* (α^u, α^l) *is causal iff* $\forall T \geq 0, \forall R$,

$$\Big(R \models_{\leq T} (\alpha^u, \alpha^l)\Big) \implies \Big(\exists R' \mid R' \models (\alpha^u, \alpha^l) \text{ and } \forall t \leq T, R(t) = R'(t)\Big)$$

where the fact that R *satisfies* (α^u, α^l) up to T *is defined by:* $R \models_{\leq T} (\alpha^u, \alpha^l)$ *iff* $\forall t \leq T, \forall \Delta \leq t, R(t) - R(t-\Delta) \in [\alpha^l(\Delta), \alpha^u(\Delta)]$.

We now give an algebraic characterization of causality. Intuitively, let us consider some pair of curves (α^u, α^l), and a cumulative curve R such that $R(T) = \alpha^l(T)$. We now look at an extension of R for x units of time after T. The number of events to emit between T and $T+x$ (i.e. $R(T+x) - R(T)$) has to be lower than $\alpha^u(x)$, but the total number of event since the beginning (i.e. $R(T+x)$) has to be greater than $\alpha^l(T+x)$. Since we have chosen $R(T) = \alpha^l(T)$, this means that the number of events emitted between T and $T+x$ has to be at least $\alpha^l(T+x) - \alpha^l(T)$. Combining the constraints on α^u and α^l, we deduce that for (α^u, α^l) to be causal, we must have $\forall x \geq 0, \alpha^u(x) \geq R(T+x) - R(T) \geq \alpha^l(T+x) - \alpha^l(T)$, hence $\alpha^l(T) \geq \sup_{x \geq 0} \left\{\alpha^l(T+x) - \alpha^u(x)\right\} = \left(\alpha^l \oslash \alpha^u\right)(T)$. This example is far from being a proof, but gives a part of the intuition, and exhibits a formula using the deconvolution operator $\oslash$. Following this intuition, it was shown in [3] a necessary and sufficient condition for the causality, based on this formula: a pair of curves is causal iff its SA-SA closure is stable when applying some deconvolutions on it. We use the classical $(min, +)$ (resp. $(max, +)$) deconvolution operators [11] $\oslash$ and $\overline{\oslash}$: for c and d in $\overline{\mathbf{C}}$, for $x \geq 0$, $(c \oslash d)(x) \stackrel{\text{def}}{=} \sup_{t \geq 0}\{c(x+t) - d(t)\}$ and $(c \,\overline{\oslash}\, d)(x) \stackrel{\text{def}}{=} \inf_{t \geq 0}\{c(x+t) - d(t)\}$.

THEOREM 1 (CHARACTERIZATION OF CAUSALITY). *Let* (α^u, α^l) *be a pair of arrival curves:*

$$(\alpha^u, \alpha^l) \text{ is causal} \iff \underline{\alpha^l} = \underline{\alpha^l} \oslash \overline{\alpha^u} \quad \text{and} \quad \overline{\alpha^u} = \overline{\alpha^u} \,\overline{\oslash}\, \underline{\alpha^l}$$

Notice that the causality, unlike SA-SA properties, is 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.

Given a pair of arrival curves, the removal of its forbidden regions is achieved using deconvolution operators, as a direct consequence of the characterization.

DEFINITION 4 (THE $\mathbb{C}$ OPERATOR). *A pair (α^u, α^l) of arrival curves has been defined as functions in $\overline{\mathbf{C}} \times \mathbf{C}$ such that $\alpha^u \geq \alpha^l$ (see Definition 1). We note $\perp_{AC}$ the set of pairs of functions in $\overline{\mathbf{C}} \times \mathbf{C}$ such that the former constraint is false; $\perp_{AC}$ will be considered as a single element, for simplicity, even if it represents an infinite set of objects. Let AC be the set of all pairs of arrival curves* plus $\perp_{AC}$. *The operator $\mathbb{C}$, is defined from AC to AC as:*

$$\mathbb{C}\left(\perp_{AC}\right) \stackrel{def}{=} \perp_{AC}$$

$$\mathbb{C}\left(\alpha^u, \alpha^l\right) \stackrel{def}{=} \begin{cases} \text{let } \mathbb{C}^u = \alpha^u \mathbin{\overline{\oslash}} \alpha^l, \mathbb{C}^l = \alpha^l \oslash \alpha^u \\ \text{if } \mathbb{C}^u \geq \mathbb{C}^l \text{ then } (\mathbb{C}^u, \mathbb{C}^l) \\ \text{else } \perp_{AC} \end{cases}$$

The operator $\mathbb{C}$ makes explicit some implicit constraints of the curves and removes some forbidden regions. In some cases, it may occur that the lower and the upper curve cross over: this means that the curves were not satisfiable (i.e. no cumulative curve satisfies it); this is denoted with the $\perp_{AC}$ value. As opposed to this, when the curves were satisfiable, the operator computes a causal pair of curves equivalent to the original ones.

THEOREM 2. *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 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). Conversely, if (α^u, α^l) is the tightest pair of curves representing a set of cumulative curves, then (α^u, α^l) is causal.*

By *tightest*, we mean the smallest (resp. the greatest) curve for the upper (resp. lower) part among the set of equivalent pairs of arrival curves. Given a pair of curves, one can compute the *causality closure* $\mathbb{C}(\overline{\alpha^u}, \underline{\alpha^l})$, and get either the information that the curves are not satisfiable, or the tightest causal representative of the original arrival curves.

3. ULTIMATELY PIECEWISE AFFINE CURVES

We now study the set of $\mathcal{U}pac$ curves comprising both a finite prefix given by a set of points and a long-term rate given by a piecewise-affine, convex/concave pairs of curves. This class of curves is the one used in the tool `ac2lus` [2]. It was chosen to be both expressive and adapted to interfacing with other formalisms.

A wider class is the one of ultimately periodic curves, widely used in analytical models (like the MPA-RTC toolbox [16]). This class is expressive, but hard to use in the interfacing with state-based formalisms. Most interfacing approaches restrict to a much stricter class. For example, [9] use convex/concave piecewise affine curves. An extension to non-convex/concave curves is proposed in [10], but involves more complex synchronization of automata, hence a greater algorithmic complexity. [1] and [15] use discrete, finite curves, which are not able to express precisely the long-term rate of the streams. [13] 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 (a long term period), hence, the generality of the model is not exploited.

The $\mathcal{U}pac$ class is basically a combination of the finite, discrete curves from [15], with the convex/concave curves of [9]. It allows a precise and possibly non-convex/concave description of the initial portion of the 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.

We first define this class of curves. The goal is to obtain an algorithm that computes the causality closure on $\mathcal{U}pac$. In order to do that, we define a normal form and we propose an algorithm to compute for each curve in $\mathcal{U}pac$ its normal form. This is the first step for the computation of the causality closure: the normal form is SA-SA and the $\mathbb{C}$ operator will be easily (in a computational manner) applicable on the normal form of the curve.

3.1 $\mathcal{U}pac$: Formal Definitions

DEFINITION 5 ($\mathcal{U}pac$). *We define the class of $\mathcal{U}pac$ curves as the set of pairs of curves (α^u, α^l) such that there exists*

- $P(\alpha^u), P(\alpha^l) \in \mathcal{N}$*: size of the finite prefix (i.e. abscissa of the last point explicitly given in the representation);*
- $N(\alpha^u), N(\alpha^l) \in \mathcal{N}$*: number of pieces of the piecewise affine part of the curves;*
- *a set of values $p_i^u \in \mathcal{N}$, $i \in [0, P(\alpha^u)]$, and a set of rational values a_j^u, b_j^u, $j \in [1, N(\alpha^u)]$: representation of the curve α^u;*
- *a set of values $p_i^l \in \mathcal{N}$, $i \in [0, P(\alpha^l)]$, and a set of rational values a_j^l, b_j^l, $j \in [1, N(\alpha^l)]$: representation of the curve α^l;*

such that, $\forall \Delta \geq 0$:

$$F^u(\Delta) = \text{if } \Delta \in [0, P(\alpha^u)] \text{ then } p^u_\Delta \text{ else } +\infty;$$

$$F^l(\Delta) = \text{if } \Delta \in [0, P(\alpha^l)] \text{ then } p^l_\Delta \text{ else } p^l_{P(\alpha^l)};$$

$$I^u(\Delta) = \begin{cases} \text{if } N(\alpha^u) > 0 \\ \text{then } \min_{j \in [1, N(\alpha^u)]} \left\{ a_j^u \Delta + b_j^u \right\} \\ \text{else } +\infty; \end{cases}$$

$$I^l(\Delta) = \begin{cases} \text{if } N(\alpha^l) > 0 \\ \text{then } \max_{j \in [1, N(\alpha^l)]} \left\{ a_j^l \Delta + b_j^l \right\} \\ \text{else } 0; \end{cases}$$

with $\alpha^u(\Delta) = \min\{F^u(\Delta), I^u(\Delta)\}$
and $\alpha^l(\Delta) = \max\left\{F^l(\Delta), I^l(\Delta)\right\}$.

The tuple $\big(P(\alpha^u)$, $P(\alpha^l)$, $N(\alpha^u)$, $N(\alpha^l)$, $\{p_i^u\}_{i \in [0, P(\alpha^u)]}$, $\{a_j^u, b_j^u\}_{j \in [1, N(\alpha^u)]}$, $\{p_i^l\}_{i \in [0, P(\alpha^l)]}$, $\{a_j^l, b_j^l\}_{i \in [1, P(\alpha^l)]}\big)$ is called the representation *of the pair (α^u, α^l). It corresponds to the data-structure to be used in algorithms. We call the set of points p_i^u and p_i^l the* finite prefix *and each line $a_j \Delta + b_j$ the* affine pieces *of (α^u, α^l).*

Note that we require the individual points to be integers (this will be necessary to ensure the convergence of the algorithms later), but remain in the fluid-event model. Figure 2

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

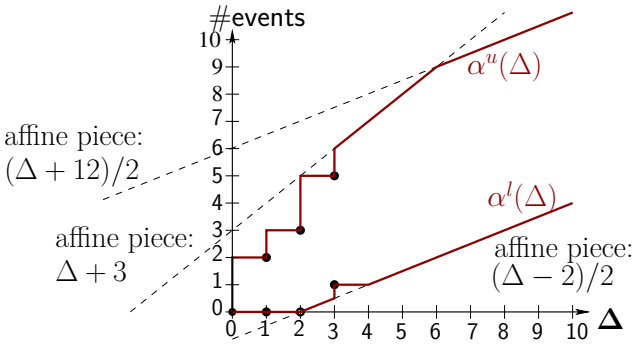


Figure 2: Example of a $\mathcal{U}pac$ curve

3.2 Normal form in $\mathcal{U}pac$

To compute the causality closure on $\mathcal{U}pac$ curves, the difficulty comes from the points of α^u and α^l, which can interact together, or with the affine pieces of the other curve. In particular, arbitrary curves with points and affine pieces are not necessarily SA-SA. We will first transform the curves to ones that obey a few well-formedness properties, which we call the normal form.

The causality closure is trivial if the curves have only affine pieces, as expressed later in Theorem 5. Curves comprising 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 difficulty can be eliminated thanks to the piecewise affine part of the curves: we can apply the SA-SA closure to the points of the curves, and only a finite number of points will remain under or above the affine pieces. If this is not the case, then it means the affine pieces add no information and can be removed. This transformation is called the normalization, and is presented in Algorithm 1; it leads to a SA-SA curve (Theorem 4) made of points and affine pieces. This allows to apply the $\mathbb{C}$ operator on it to get a causal pair of curves. The computation of $\mathbb{C}$ is made easier by Theorem 6, which reduces the computation of $\mathbb{C}$ to a version where all the operators are bounded.

3.2.1 Properties of SA-SA Closure of Finite Prefix

We first focus on the finite prefix (*i.e.* the points) of $\mathcal{U}pac$ curves, in order to introduce two intermediate results that will be useful in the computation of the normal form and of the causality closure.

We first need to define properly the notion of "finite curves" and the associated operators. We define "finite curves" as restriction of infinite ones: the restriction of (α^u, α^l) *to* $[0, T]$ is defined as:

$$\forall t \le T, \quad {\alpha^u}\big|_{\mathrm{T}}(t) \stackrel{\text{def}}{=} \alpha^u(t) \text{ and } {\alpha^l}\big|_{\mathrm{T}}(t) \stackrel{\text{def}}{=} \alpha^l(t)$$

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

This way, ${\alpha^u}\big|_{\mathrm{T}}$ and ${\alpha^l}\big|_{\mathrm{T}}$ match the intuition of finite curves, can be easily represented with a finite set of points, but are still infinite objects on which the usual theorems apply. This notion matches the $\mathcal{U}pac$ subset of curves with no affine pieces ($N(\alpha^l) = N(\alpha^u) = 0$).

The SA-SA closure $\left(\overline{{\alpha^u}\big|_{\mathrm{T}}}, \underline{{\alpha^l}\big|_{\mathrm{T}}}\right)$ represent the same set of cumulative curves as $({\alpha^u}\big|_{\mathrm{T}}, {\alpha^l}\big|_{\mathrm{T}})$. The situation is illustrated on Figure 3, representing an upper curve defined explicitly up to $\Delta = 3$, and its sub-additive closure.

It should be noted that, in this case, $(\overline{\alpha^u}, \underline{\alpha^l})$ is no longer a $\mathcal{U}pac$ curve, since it has an infinite number of points (as far as $\exists t > 0. \alpha^u(t) < +\infty$). However, one can define the property *SA-SA up to P* and the associated closure (see [3] for details): the SA-SA closure up to P and $(\overline{\alpha^u}, \underline{\alpha^l})$ will be identical on $[0, P]$. Additionally, [5] (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 $[0, P]$ leading to a simple, quadratic algorithm. The SA-SA closure up to $P(\alpha^l)$ (resp. $P(\alpha^u)$) can be applied to the finite prefix of any pair of curves in $\mathcal{U}pac$. The finite prefix of α^u (resp. α^l) allows us to compute the *slope* $S^P(\alpha^u)$ (resp. $S^P(\alpha^l)$) of the curve, the *point of maximal influence* $\Delta^P(\alpha^u)$ (resp. $\Delta^P(\alpha^l)$), and the *maximal drift* $d_m^P(\alpha^u)$ (resp. $d_m^P(\alpha^l)$) defined as follows and illustrated in Figure 3.

Definition 6. *Let (α^u, α^l) be a pair of arrival curves, and $P > 0$. We define the following:*

$$S^P(\alpha^u) \stackrel{def}{=} \min_{\Delta \le P} \{\alpha^u(\Delta)/\Delta\}$$

$$\Delta^P(\alpha^u) \stackrel{def}{=} \min\{\Delta \in [0, P] \quad | \quad S^P(\alpha^u) \times \Delta = \alpha^u(\Delta)\}$$

$$d_m^P(\alpha^u) \stackrel{def}{=} \sup_{\Delta \le \Delta^P(\alpha^u)} \{\overline{\alpha^u}(\Delta) - S^P(\alpha^u) \times \Delta\}$$

$S^P(\alpha^l)$, $\Delta^P(\alpha^l)$ and $d_m^P(\alpha^l)$ are defined in a symmetrical way.

Since we work here in discrete time, the min and max are over finite sets and are well-defined.

Interesting properties of the definitions are given by the following lemma. The sub-additive closure of the curve remains above the line $S^P(\alpha^u) \times \Delta$ defined by the slope (Lemma 2). The distance of the curve to this line remains bounded, and the bound $d_m^P(\alpha^u)$ depends only on the finite prefix (Lemma 3). Also, $\overline{\alpha^u}$ will actually have contact with the line $S^P(\alpha^u) \times \Delta$ at least periodically, with a period of $\Delta^P(\alpha^u)$ which is smaller than the size of the finite prefix (Lemma 1).

Lemma 1. *Let (α^u, α^l) be a pair of arrival curves and $T > 0$. Then:*

$$\forall k \in \mathcal{N}, \quad \overline{{\alpha^u}\big|_T}(k \times \Delta^P(\alpha^u)) = k \times \Delta^P(\alpha^u) \times S^P(\alpha^u)$$
$$\underline{{\alpha^l}\big|_T}(k \times \Delta^P(\alpha^l)) = k \times \Delta^P(\alpha^l) \times S^P(\alpha^l)$$

Lemma 2. *Let (α^u, α^l) be a pair of arrival curves and $T > 0$. Then:*

$$\forall \Delta, \quad \alpha^u(\Delta) \ge \overline{{\alpha^u}\big|_T}(\Delta) \ge S^P(\alpha^u) \times \Delta$$
$$\alpha^l(\Delta) \le \underline{{\alpha^l}\big|_T}(\Delta) \le S^P(\alpha^l) \times \Delta$$

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

$$\forall \Delta \ge 0, \quad \overline{\alpha^u}(\Delta) - S^P(\alpha^u) \times \Delta \le d_m^P(\alpha^u)$$
$$S^P(\alpha^l) \times \Delta - \underline{\alpha^l}(\Delta) \le d_m^P(\alpha^l)$$

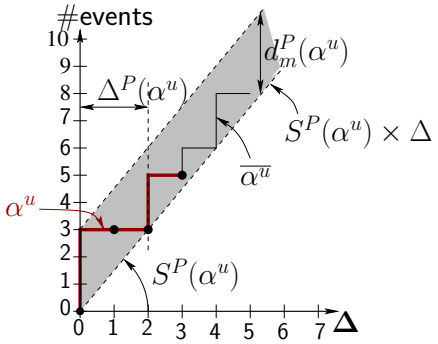


Figure 3: Point of maximal influence of α^u. The curve α^u remains in the greyed area and has contact with $S^P(\alpha^u) \times \Delta$ at least with period $\Delta^P(\alpha^u)$.

As a consequence, affine pieces $a\Delta + b$ with a slope steeper than $S^P(\alpha^u) \times \Delta$ do not add information to the curve (provided the explicit points of α^u are below the affine piece), and can be removed.

THEOREM 3. *Let (α^u, α^l) be a pair of arrival curves in $\mathcal{U}pac$, different from $\perp_{AC}$, and $J \in [1, N(\alpha^u)]$ such that these two conditions are satisfied:*

$$\forall i \in [0, P(\alpha^u)], \forall j \in [1, N(\alpha^u)], \quad p_i^u \leq a_j^u \times i + b_j^u$$
$$a_J^u \geq S^P(\alpha^u)$$

Then, removing the affine piece $a_J^u + b_J^u$ from (α^u, α^l) yields an equivalent curve. In this case, we say that the affine piece $a_J^u + b_J^u$ is not relevant.

Similarly for α^l, if

$$\forall i \in [0, P(\alpha^l)], \forall j \in [1, N(\alpha^l)], \quad p_i^l \leq a_j^l \times i + b_j^l$$
$$a_J^l \geq S^P(\alpha^l)$$

Then, removing the affine piece $a_J^l + b_J^l$ from (α^u, α^l) yields an equivalent curve.

3.2.2 Normal Form: Definition and Algorithm

We now have the necessary background to introduce the normal form, on which the causality closure will be computed:

DEFINITION 7 (NORMAL FORM OF CURVES IN $\mathcal{U}pac$). *A pair of arrival curves (α^u, α^l) in $\mathcal{U}pac$ is said to be in normal form if $P(\alpha^u) = P(\alpha^l) = P$ and at least one of the following conditions is satisfied:*

1. *$(\alpha^u, \alpha^l) = \perp_{AC}$*
2. *$N(\alpha^u) = N(\alpha^l) = 0$ and (α^u, α^l) is SA-SA up to P*
3. *$N(\alpha^u) = 0$, α^u is sub-additive up to P and α^l is super-additive.*
4. *$N(\alpha^l) = 0$, α^l is super-additive up to P and α^u is sub-additive.*
5. *(α^u, α^l) is SA-SA*

Case 2 corresponds to the case of finite curves [3]. In this case, we say that (α^u, α^l) has no relevant affine pieces. Cases 3 and 4 correspond to asymmetric cases where only one of α^u and α^l has relevant affine pieces. In these cases, we consider the SA-SA curves in theory, but the representation is restricted to the SA-SA set of points on the prefix.

Converting an arbitrary pair of curves into a normal-form is straightforward using the properties stated in Section 3.2.1; the algorithm is given below. For the common case where both curves have relevant affine pieces *(case 5)*, the transformation of a pair of curves into normal form is illustrated by Figure 4. It essentially consists in adding explicit points to the curve until one can be sure all the points are above the affine pieces. In the general case, the transformation is as follows.

ALGORITHM 1 (NORMALIZATION OF CURVES IN $\mathcal{U}pac$). *For any curve in $\mathcal{U}pac$, we apply the steps:*

1. *Make sure all the explicit points p_i^u (resp. p_i^l) are under (resp. above) all affine pieces; if not, modify p_i^u (resp. p_i^l), keeping integer abscissa; add points on α^u or α^l until $P(\alpha^u) = P(\alpha^l) = P$. See Figure 4.(b).*
2. *Eliminate affine pieces of α^u (resp. α^l) which have a slope greater or equal (resp. lower or equal) to $S^P(\alpha^u)$ (resp. $S^P(\alpha^l)$). It can be proved that this does not change the curve. See Figure 4.(c).*

Then, multiple cases can occur:

If $N(\alpha^u) = N(\alpha^l) = 0$ **then**

3. *Apply the SA-SA closure up to P.*

If $N(\alpha^u) \neq 0$ **and** $N(\alpha^l) \neq 0$ **then**

3. *Compute the abscissa M_j^u of the intersection between $S^P(\alpha^u) \times \Delta$ and the affine piece j of α^u (and similarly M^l for α^l). Set $M^u = \min_j\{M_j^u\}$, $M^l = \min_j\{M_j^l\}$, $M = \max\{M^u, M^l\}$.*
4. *Add explicit points p^u and p^l to the curves, so that $P(\alpha^u) = P(\alpha^l) = M$.*
5. *Apply the SA-SA closure up to M to (α^u, α^l). See Figure 4.(d).*

If $N(\alpha^u) \neq 0$ **and** $N(\alpha^l) = 0$ **then**

3. *Compute the abscissa M_j^u of the intersection between $S^P(\alpha^u) \times \Delta$ and the affine piece j of α^u. Set $M = \min_j\{M_j^u\}$.*
4. *Add explicit points p^u and p^l to the curves, so that $P(\alpha^u) = P(\alpha^l) = M$.*
5. *Apply the SA-SA closure up to M to (α^u, α^l).*

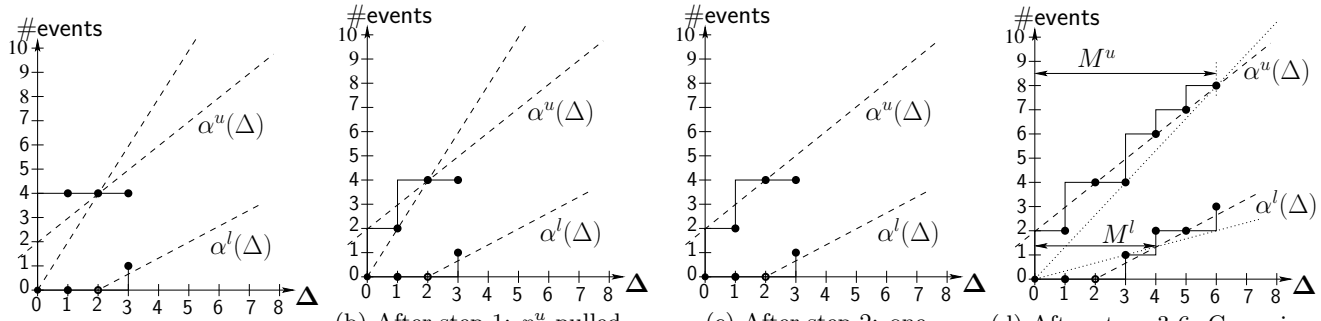
If $N(\alpha^u) = 0$ **and** $N(\alpha^l) \neq 0$ **then** *apply the same transformation as above, replacing α^l by α^u and vice-versa in the text.*

The normalization trivially implies SA-SA closure up to M. The SA-SA property is actually true for the whole curve, when it has some relevant affine pieces:

THEOREM 4. *Let (α^u, α^l) be a pair of curves obtained by applying the normalization (Algorithm 1) on a pair of curves in $\mathcal{U}pac$. Then (α^u, α^l) is in $\mathcal{U}pac$ and in normal form. In particular, if α^l (resp. α^u) has at least one relevant affine piece, then α^l is super-additive (resp. sub-additive).*

4. CAUSALITY CLOSURE IN $\mathcal{U}pac$

First, the algorithm for the causality closure applies the normalization on the curves. Then, the idea is to apply Theorem 2 with the operator $\mathbb{C}$ (see Definition 4) on the curves: the theorem states that $\mathbb{C}\left(\overline{\alpha^u}, \underline{\alpha^l}\right)$ is the causality closure of (α^u, α^l). This step is divided into 2 parts. The part where the curves have no relevant affine pieces at all was treated in [3] and is quickly recalled in Section 4.1; it requires a fix-point computation. The other part of the algorithm factorizes the three other cases where curves have at least one affine piece (no affine pieces on α^l, no affine pieces on α^u, both curves with affine pieces) and is provided in Section 4.2.



(a) Original curve. (b) After step 1: p_1^u pulled below the affine piece. (c) After step 2: one affine piece removed. (d) After steps 3-6: Curve in normal form.

Figure 4: Step by step transformation into normal form

4.1 Curves With No Affine Pieces

For curves in normal form in $\mathcal{U}pac$ with no relevant affine pieces at all *(case 2 of the normal form)*, the algorithm has been proposed in [3] and is briefly summarized here. The major difficulty for the curves with only a finite prefix is that their SA-SA closure is not representable with a finite number of points; so we cannot directly use the result of Theorem 2. Instead, we use the fact that any fix-point of $\mathbb{C}$ is causal. Therefore, the algorithm iterates the computation of $\mathbb{C}$ on the finite prefix of the curves; the termination is ensured by the fact that the points of the prefix are natural numbers. The iteration either reaches the $\perp_{AC}$ value (the curves were not satisfiable) or a causal curve equivalent to the original one. Computing the finite SA-SA closure on the resulting curves provides the expected result.

4.2 $\mathbb{C}$ for Curves With at Least One Affine Piece

As a first remark, let us consider the particular case of convex/concave affine piecewise curves. This is an interesting class of curves in $\mathcal{U}pac$, used for example in [9, 10]. It corresponds to the particular case where the finite prefix uniquely contains $(0,0)$. An interesting property is that 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. Those curves are always causal:

THEOREM 5. *Let $(\alpha^u, \alpha^l) \neq \perp_{AC}$ be a pair of piecewise affine, concave/convex curves. Then (α^u, α^l) is causal.*

We now focus on the general case, *i.e.* curves in normal form in $\mathcal{U}pac$, *with either α^l, α^u or both having affine pieces*: $N(\alpha^u) > 0$ or $N(\alpha^l) > 0$. We show that we can directly apply the operator $\mathbb{C}$ on the curves and that its computation can be done in quadratic time.

THEOREM 6. *Let (α^u, α^l) be a pair of curves in $\mathcal{U}pac$, in normal form, such that $(\alpha^u, \alpha^l) \neq \perp_{AC}$, with either α^u or α^l having relevant affine pieces. Let $M = P(\alpha^l) = P(\alpha^u)$ be the index of the last point of (α^u, α^l) given explicitly (as it was computed in Algorithm 1). Let $\mathbb{C}\big|_M = \left(\mathbb{C}\big|_M^u, \mathbb{C}\big|_M^l\right)$ be the following operator: $\forall \Delta \geq 0$,*

$\mathbb{C}\big|_M^u\left(\alpha^u, \alpha^l\right)(\Delta) = \inf_{t\in[0,M]}\{\alpha^u(\Delta + t) - \alpha^l(t)\}$ *and* $\mathbb{C}\big|_M^l\left(\alpha^u, \alpha^l\right)(\Delta) = \sup_{t\in[0,M]}\{\alpha^l(\Delta + t) - \alpha^u(t)\}$

1. $\forall \Delta \geq 0, \mathbb{C}\left(\overline{\alpha^u}, \underline{\alpha^l}\right)(\Delta) = \mathbb{C}\big|_M\left(\overline{\alpha^u}, \underline{\alpha^l}\right)(\Delta)$
2. *If $N(\alpha^u) \neq 0$ then $\forall \Delta > M$, $\mathbb{C}^u\left(\overline{\alpha^u}, \underline{\alpha^l}\right)(\Delta) = \alpha^u(\Delta)$*
3. *If $N(\alpha^l) \neq 0$ then $\forall \Delta > M$, $\mathbb{C}^l\left(\overline{\alpha^u}, \underline{\alpha^l}\right)(\Delta) = \alpha^l(\Delta)$*

We give the intuition of the proof (see [12] for details) with an affine piece of α^l, and consider its interaction with the finite prefix of α^u. We consider the slope $S^P(\alpha^u)$ of the finite prefix and the slope a of the affine piece. If $S^P(\alpha^u) \geq a$, then according to the results of Section 3.2.1, the curve $\overline{\alpha^u}$ will remain above $S^P(\alpha^u) \times \Delta$ and it cannot create any forbidden region with the affine piece. If $S^P(\alpha^u)$ is lower than a, then we know that $\overline{\alpha^u}$ will "touch" periodically $S^P(\alpha^u) \times \Delta$ and will eventually end up below the affine piece. This implies that the curves are not satisfiable.

As a consequence, the $\mathbb{C}$ operator can easily be computed algorithmically: for each point to compute, the inf{} and the sup{} can be computed with a simple `for` loop iterating from 0 to M. The expression of $\mathbb{C}\big|_M\left(\overline{\alpha^u}, \underline{\alpha^l}\right)$ includes a SA-SA closure. When the curve has affine pieces, it is already SA-SA, hence no SA-SA closure needs to be applied. However, for curve with no affine pieces, since we only use the values of the SA-SA curves for $\Delta \leq 2M$, it is sufficient to compute the SA-SA closure up to $2M$. Furthermore, when the curve has at least one affine piece, this computation has to be done for the points of abscissa from 0 to M, the other points are given by the original curve itself.

Based on these remarks, the causality closure algorithm for $\mathcal{U}pac$ curves with at least one relevant affine piece follows:

ALGORITHM 2. *Given a pair of curves (α^u, α^l) in $\mathcal{U}pac$ in normal form represented by p_i^u, a_j^u, b_j^u, p_i^l, a_k^l, b_k^l ($i \in [0, M]$, $j \in [1, N(\alpha^u)]$, $k \in [1, N(\alpha^l)]$), we denote by p_i^{u*}, a_j^{u*}, b_j^{u*}, p_i^{l*}, a_k^{l*}, b_k^{l*} the representation of the causality closure $\mathbb{C}\left(\overline{\alpha^u}, \underline{\alpha^l}\right)$. This representation is computed as follows:*

- *In all cases, the affine pieces do not change (this is ensured by cases 2 and 3 of theorem 6):*

$$a_j^{u*} = a_j^u, \quad b_j^{u*} = b_j^u, \quad a_k^{l*} = a_k^l, \quad b_k^{l*} = b_k^l$$

- *To compute the points p_i^* of the finite prefix, define $(\alpha^u{}_{2M}, \alpha^l{}_{2M})$, a pair of curves: if $N(\alpha^u) \neq 0$ then $\alpha^u_{2M} = \alpha^u$ else the finite prefix of α^u_{2M} is the subadditive closure of α^u up to $2M$ and it has no affine pieces (likewise for α^l_{2M}). Then:*

$$p_i^{u*} = \mathbb{C}\big|_M^u(\alpha^u_{2M}, \alpha^u_{2M})(i), \quad p_i^{l*} = \mathbb{C}\big|_M^l(\alpha^u_{2M}, \alpha^u_{2M})(i)$$

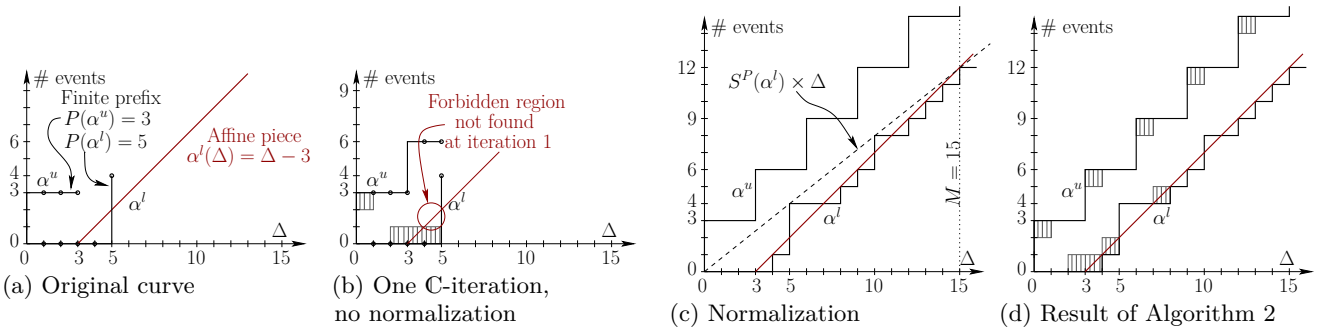


Figure 5: Causality Closure on a $\mathcal{U}pac$ Curve With One Affine Piece

Figure 5 illustrates the whole causality closure algorithm on an example. The pair of curves is given in Figure 5.(a): α^u has no affine piece, and α^l has one. Figure 5.(b) shows an attempt to use the $\mathbb{C}$ operator on the curves without performing a 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 (5.(c)) adds some points to the prefix of the curves, and applying $\mathbb{C}|_M$ on the result yields a causal pair of curves, without further iteration (5.(d)).

5. CONCLUSION

This paper provides an algorithm to compute the causality closure on an interesting class of curves, already used in several tools [9, 2]: curves with a finite prefix made of points, followed by convex/concave affine pieces. This class enables the precise modeling of the beginning of the curves together with the long term rate information. For this class of curves, the operators which compute the causality closure cannot be straightforwardly deduced from the initial work on causality closure [3]. This new algorithm can handle all the cases of curves in the class; it is efficient, quadratic in complexity (with reasonably-sized curves, this means the computation is almost instantaneous).

Furthermore, while the problem appeared to be relatively simple, the algorithm relies on several theorems, whose proofs were indeed non-trivial [12].

This work completes the theoretical and computational foundations for the connection of Real-Time Calculus to synchronous languages, implemented in the tool ac2lus [2]. Further works include to apply it to larger and more realistic case studies. This may involve using other verification tools and may require changes in the abstractions used to represent the set of streams.

6. REFERENCES

[1] K. Altisen, Y. Liu, and M. Moy. Performance evaluation of components using a granularity-based interface between real-time calculus and timed automata. In *QAPL*, 2010.

[2] K. Altisen and M. Moy. ac2lus: Bringing SMT-solving and abstract interpretation techniques to real-time calculus through the synchronous language Lustre. In *ECRTS*, Brussels, Belgium, Jully 2010.

[3] K. Altisen and M. Moy. Arrival curves for real-time calculus: the causality problem and its solutions. In *TACAS*, March 2010.

[4] A. Bouillard, L. Jouhet, and E. Thierry. Service curves in Network Calculus: dos and don'ts. Technical report.

[5] A. Bouillard and É. Thierry. An algorithmic toolbox for network calculus. *Discrete Event Dynamic Systems*, 18(1):3–49, 2008.

[6] N. Halbwachs, F. Lagnier, and C. Ratel. Programming and verifying critical systems by means of the synchronous data-flow programming language LUSTRE. *Transactions on Software Engineering*, 1992.

[7] S. Künzli, F. Poletti, L. Benini, and L. Thiele. Combining simulation and formal methods for system-level performance analysis. In *DATE*, pages 236–241, 3001 Leuven, Belgium, Belgium, 2006.

[8] S. Künzli and L. Thiele. Generating event traces based on arrival curves. In *MMB*, 2006.

[9] K. Lampka, S. Perathoner, and L. Thiele. Analytic real-time analysis and timed automata: A hybrid method for analyzing embedded real-time systems. In *EMSOFT*, 2009.

[10] K. Lampka, S. Perathoner, and L. Thiele. Analytic real-time analysis and timed automata: a hybrid methodology for the performance analysis of embedded real-time systems. *Design Automation for Embedded Systems*, pages 1–35, June 2010.

[11] J.-Y. Le Boudec and P. Thiran. *Network Calculus*. Springer Verlag, 2001.

[12] M. Moy and K. Altisen. Causality closure for a new class of curves in real-time calculus full version. Technical Report TR-2011-13, Verimag Research Report, 2011.

[13] L. T. Phan, S. Chakraborty, P. Thiagarajan, and L. Thiele. Composing functional and state-based performance models for analyzing heterogeneous real-time systems. In *RTSS*, 2007.

[14] L. Thiele, S. Chakraborty, and M. Naedele. Real-time calculus for scheduling hard real-time systems. In *ISCAS*, 2000.

[15] Uppsala University. Cats tool, 2007. `http://www.timestool.com/cats`.

[16] E. Wandeler. *Modular Performance Analysis and Interface-Based Design for Embedded Real-Time Systems*. PhD thesis, PhD Thesis ETH Zurich, 2006.