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

Karine Altisen and Matthieu Moy

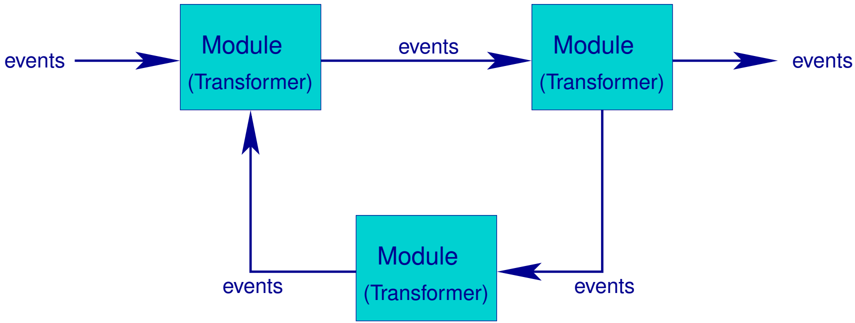
Grenoble INP (Verimag)
Grenoble
France

WCTT, 29 November 2011

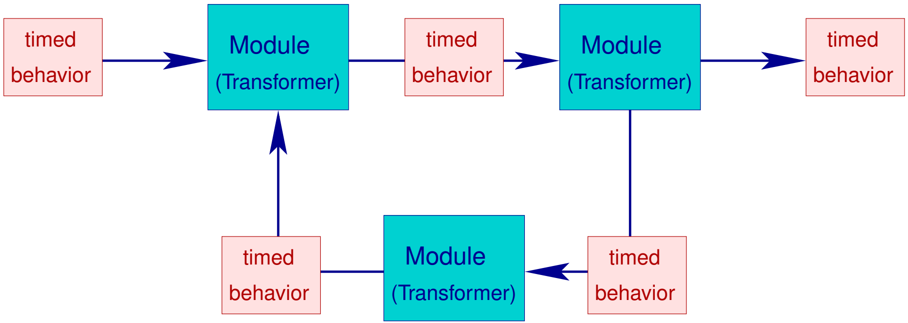
Summary

Summary

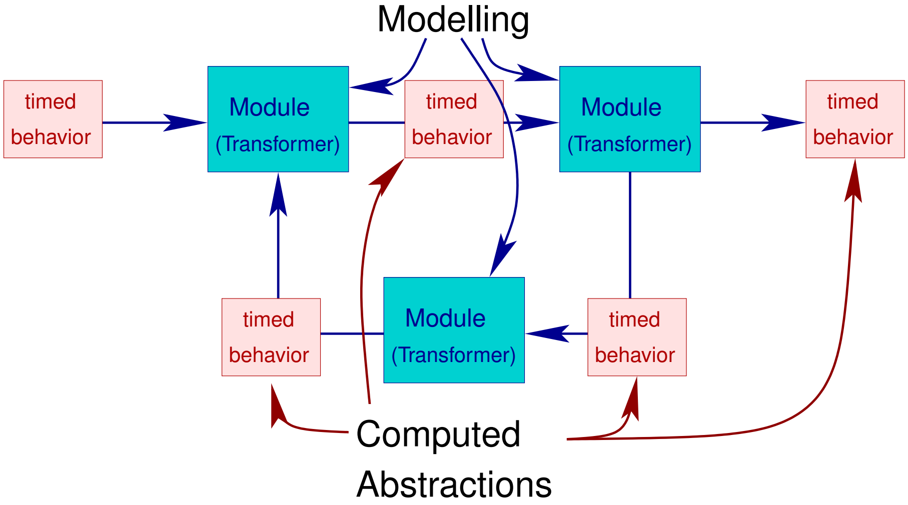
Modular Performance Analysis (MPA): The Big Picture



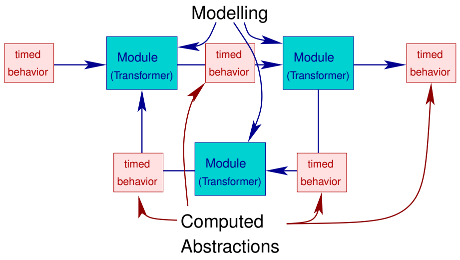
Modular Performance Analysis (MPA): The Big Picture



Modular Performance Analysis (MPA): The Big Picture

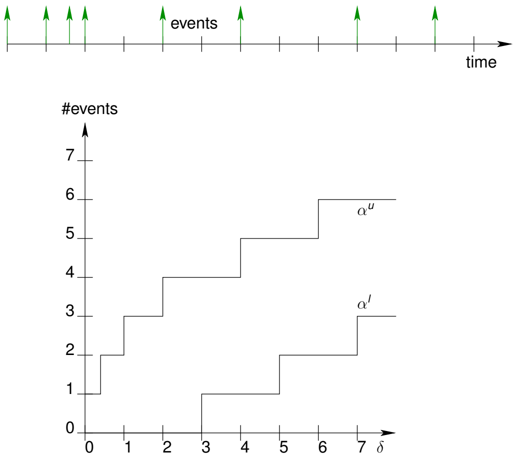


Modular Performance Analysis (MPA)



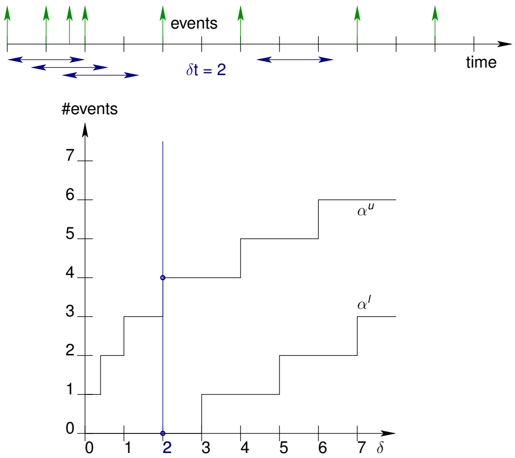
- With Real-Time Calculus (RTC):
  - "Timed Behavior" = "Arrival Curves"
  - "Modules" = arrival curves transformers (usually, generic components + service curves)

Arrival Curves in Real-Time Calculus (RTC)



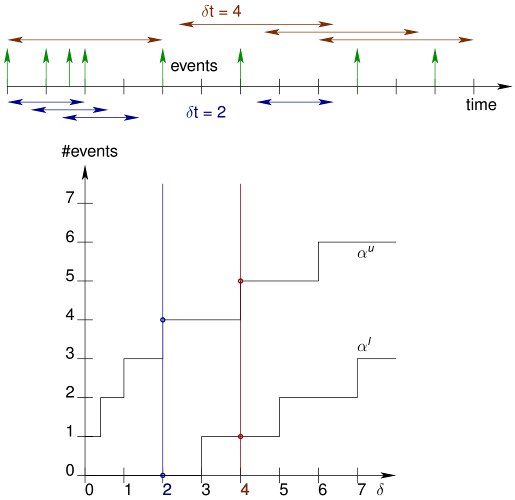
- $\alpha^u(\delta)$: max number of events in any window of size δ.
- $\alpha^l(\delta)$: min number of events in any window of size δ.

Arrival Curves in Real-Time Calculus (RTC)



- $\alpha^u(\delta)$: max number of events in any window of size δ.
- $\alpha^l(\delta)$: min number of events in any window of size δ.

Arrival Curves in Real-Time Calculus (RTC)



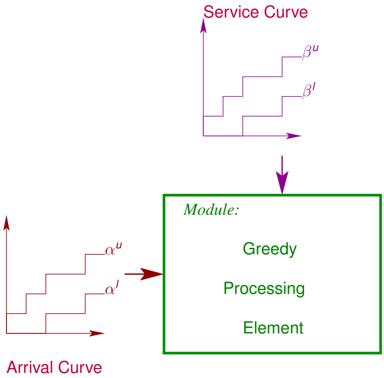
- $\alpha^u(\delta)$: max number of events in any window of size δ.
- $\alpha^l(\delta)$: min number of events in any window of size δ.

Modules = Arrival/Service Curve Transformers

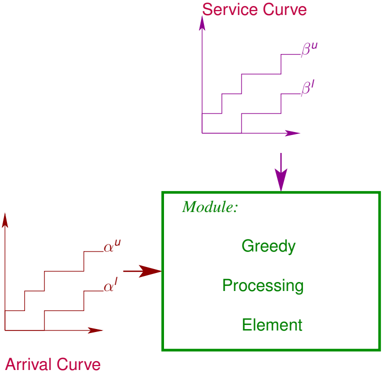
Module:

Greedy
Processing
Element

Modules = Arrival/Service Curve Transformers



Modules = Arrival/Service Curve Transformers



Analysis

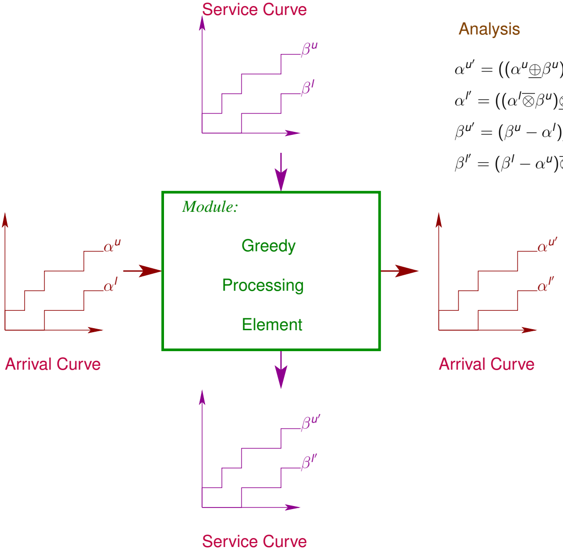
$$\alpha^{u'} = ((\alpha^u \underline{\oplus} \beta^u) \overline{\oplus} \beta^l) \wedge \beta^u$$

$$\alpha^{l'} = ((\alpha^l \overline{\otimes} \beta^u) \underline{\otimes} \beta^l) \wedge \beta^l$$

$$\beta^{u'} = (\beta^u - \alpha^l) \underline{\otimes} 0$$

$$\beta^{l'} = (\beta^l - \alpha^u) \overline{\otimes} 0$$

Modules = Arrival/Service Curve Transformers



Analysis

$$\alpha^{u'} = ((\alpha^u \underline{\oplus} \beta^u) \overline{\oplus} \beta^l) \wedge \beta^u$$

$$\alpha^{l'} = ((\alpha^l \overline{\otimes} \beta^u) \underline{\otimes} \beta^l) \wedge \beta^l$$

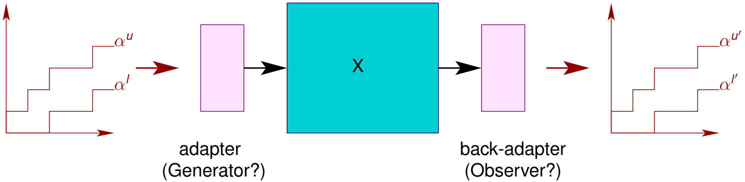
$$\beta^{u'} = (\beta^u - \alpha^l) \underline{\otimes} 0$$

$$\beta^{l'} = (\beta^l - \alpha^u) \overline{\otimes} 0$$

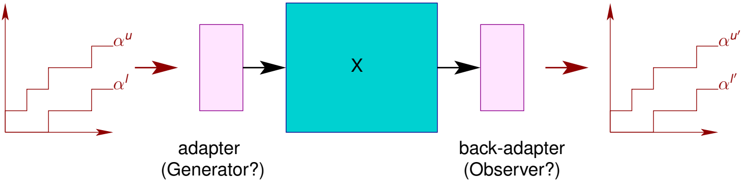
Real-Time Calculus (RTC): pros and cons

- Nice things with RTC
  - Can model: event streams, simple scheduling policies
  - Scales up nicely
  - Exact hard bounds
- Less nice thing with RTC
  - Limited expressiveness

Allowing more Complex Modules in MPA



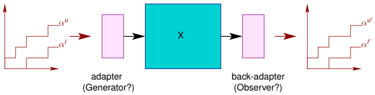
Allowing more Complex Modules in MPA



- X = Arbitrary program ⇒ testing (ETHZ)
- X = Timed Automata ⇒ model-checking (ETHZ, Uppsala, Verimag)
- X = Lustre ⇒ Abstract Interpretation, SMT Solving (tool ac2lus, Verimag)

Summary

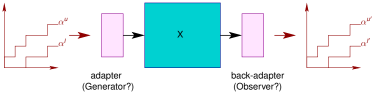
A Closer Look at the Generator



- The idea behind generators:

  "at each point in time, compute an interval $[l, u]$ on the number of events that can be emitted".

A Closer Look at the Generator

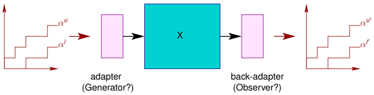


- The idea behind generators:

“at each point in time, compute an interval $[l, u]$ on the number of events that can be emitted”.

What if $l > u$? $\Rightarrow$ deadlock.

A Closer Look at the Generator



- The idea behind generators:

"at each point in time, compute an interval $[l, u]$ on the number of events that can be emitted".

What if $l > u$? $\Rightarrow$ deadlock.

This talk:

How can we prevent these deadlocks?

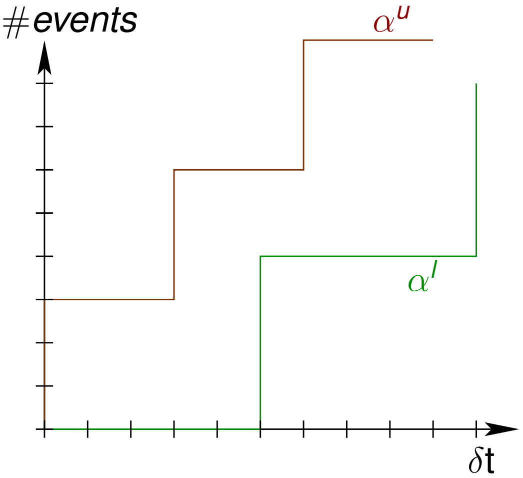
Causal and Non-Causal Curves

- A pair of arrival curve (α^u, α^l) is causal iff an event stream conformant with (α^u, α^l) *up to time t* can be extended into a stream conformant with (α^u, α^l) *forever*.

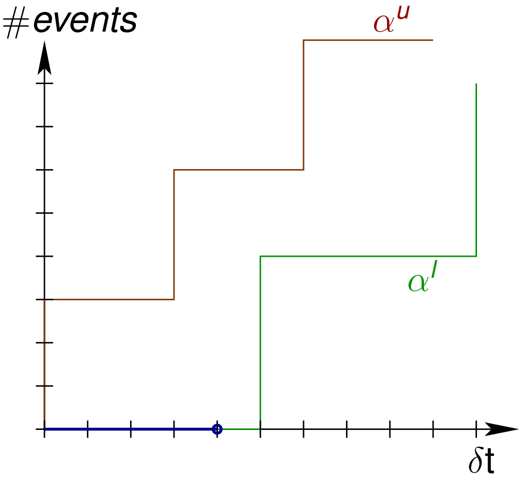
Causal and Non-Causal Curves

- A pair of arrival curve (α^u, α^l) is causal iff an event stream conformant with (α^u, α^l) *up to time t* can be extended into a stream conformant with (α^u, α^l) *forever*.
- i.e., (α^u, α^l) is causal iff the associated generator has no deadlock.

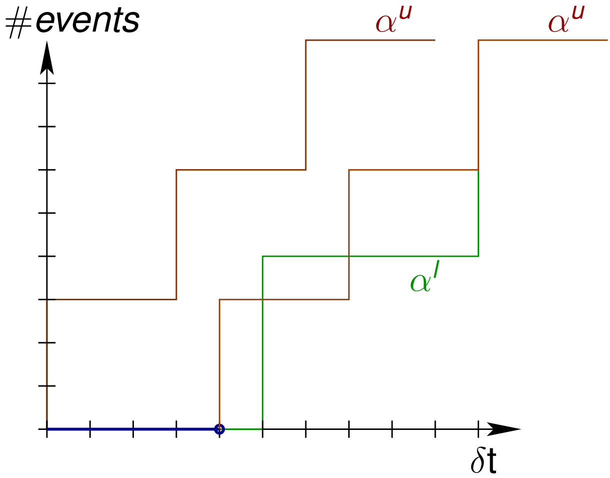
Causality problem: Forbidden Regions



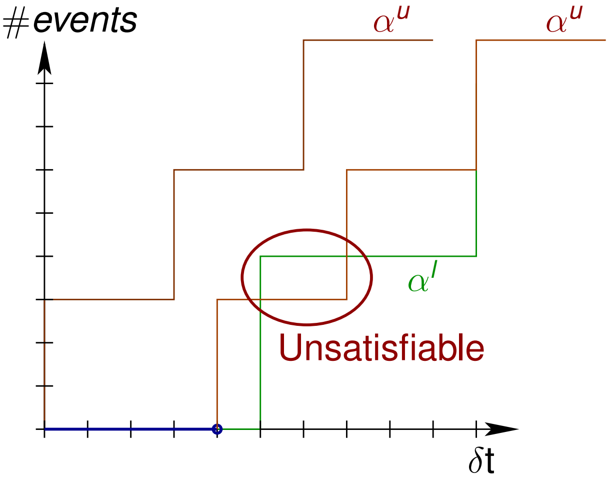
Causality problem: Forbidden Regions



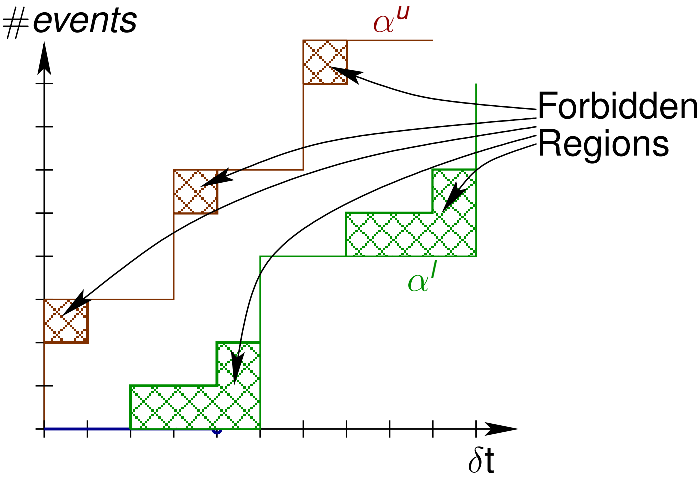
Causality problem: Forbidden Regions



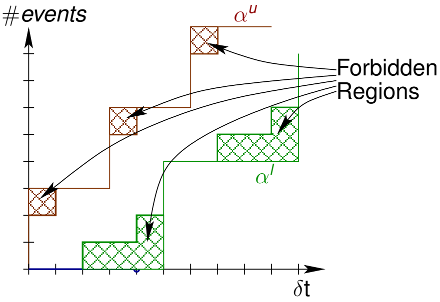
Causality problem: Forbidden Regions



Causality problem: Forbidden Regions



Causality problem: Forbidden Regions



If a stream gets in a forbidden region,
it will eventually reach a dead-end

Problems with Non-Causal Curves

- Simulating a Generator: the generator may stop with "you shouldn't have been there, I can't continue"
- Formal verification: spurious counter-examples (finite event stream that cannot be extended into an infinite one)

Problems with Non-Causal Curves

- Simulating a Generator: the generator may stop with "you shouldn't have been there, I can't continue"
- Formal verification: spurious counter-examples (finite event stream that cannot be extended into an infinite one)
- Practical issue: non-causal curves are hard to think with!

Problems with Non-Causal Curves

- Simulating a Generator: the generator may stop with "you shouldn't have been there, I can't continue"
- Formal verification: spurious counter-examples (finite event stream that cannot be extended into an infinite one)
- Practical issue: non-causal curves are hard to think with!

- Good news:

$$(\alpha^u, \alpha^l) \text{ is causal}$$
$$\Leftrightarrow$$
$$(\alpha^u, \alpha^l) \text{ is the tightest equivalent pair of curve}$$

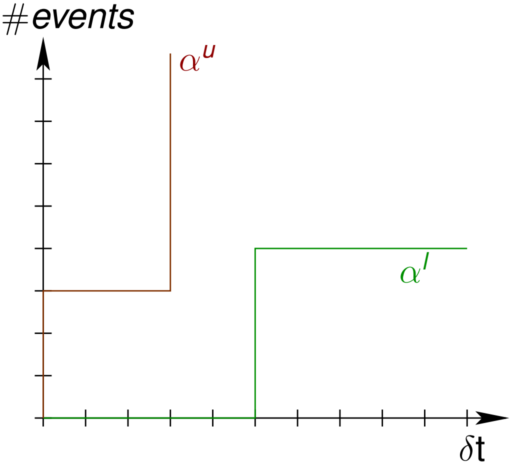
Summary

Causality Closure: Making Curves Causal

- The causality closure of (α^u, α^l) is a pair of curves that is:
  - Equivalent to (α^u, α^l) (i.e. same set of accepted event streams)
  - Causal (i.e. finite accepted event streams can be extended infinitely)
- How to compute it?
- Intuition: Causal curves are curves without forbidden regions (?)

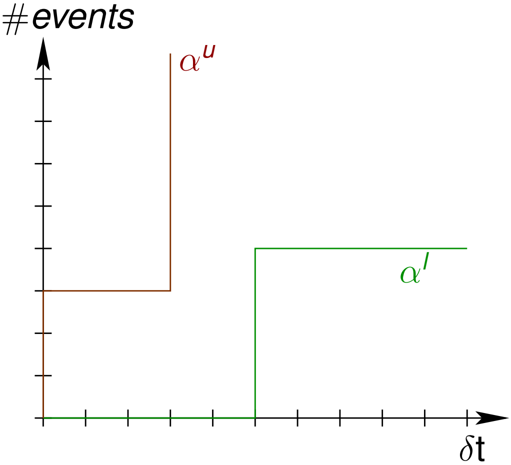
Towards an Algorithm for Causality Closure

- First idea: deconvolution ($\oslash$, $\overline{\oslash}$) to remove forbidden regions
- Insufficient:



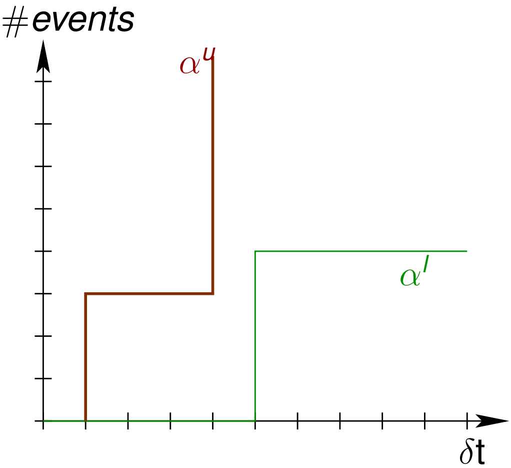
Towards an Algorithm for Causality Closure

- First idea: deconvolution ($\oslash$, $\overline{\oslash}$) to remove forbidden regions
- Insufficient:



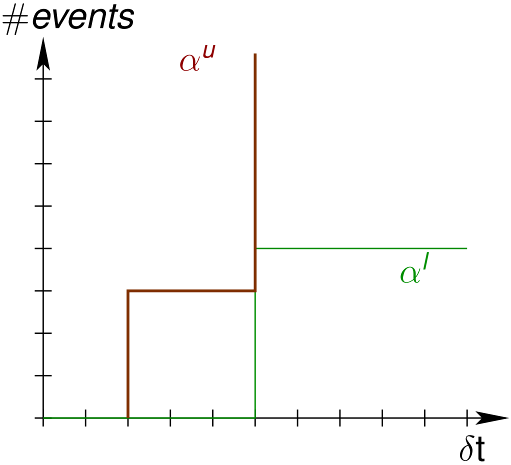
Towards an Algorithm for Causality Closure

- First idea: deconvolution ($\oslash$, $\overline{\oslash}$) to remove forbidden regions
- Insufficient:



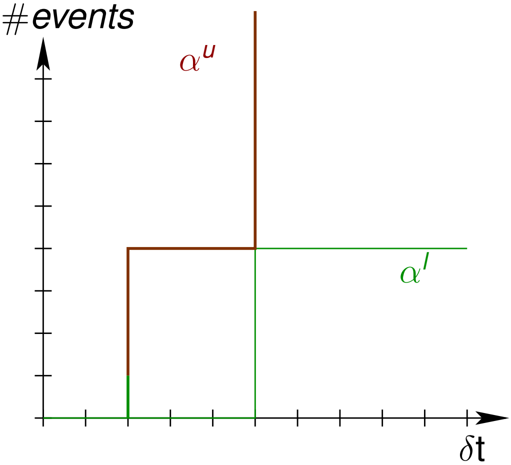
Towards an Algorithm for Causality Closure

- First idea: deconvolution ($\oslash$, $\overline{\oslash}$) to remove forbidden regions
- Insufficient:



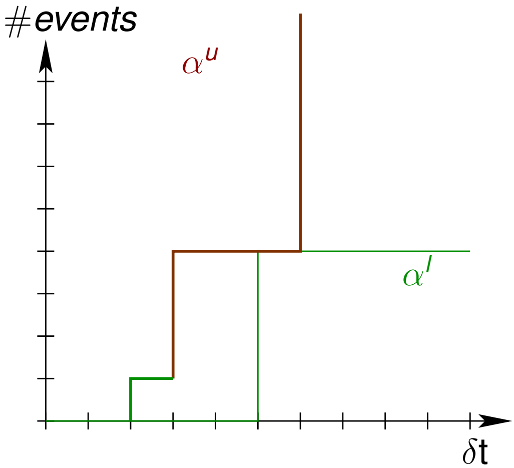
Towards an Algorithm for Causality Closure

- First idea: deconvolution ($\oslash$, $\overline{\oslash}$) to remove forbidden regions
- Insufficient:



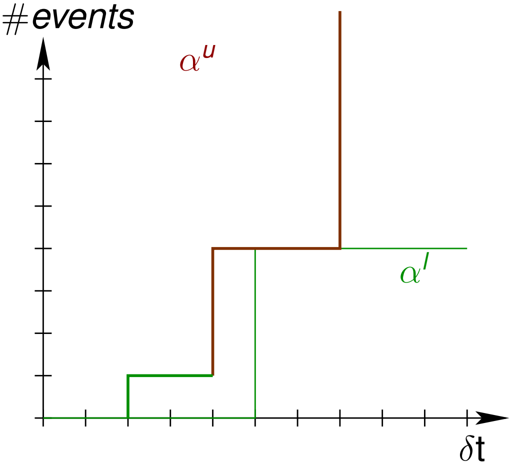
Towards an Algorithm for Causality Closure

- First idea: deconvolution ($\oslash$, $\overline{\oslash}$) to remove forbidden regions
- Insufficient:



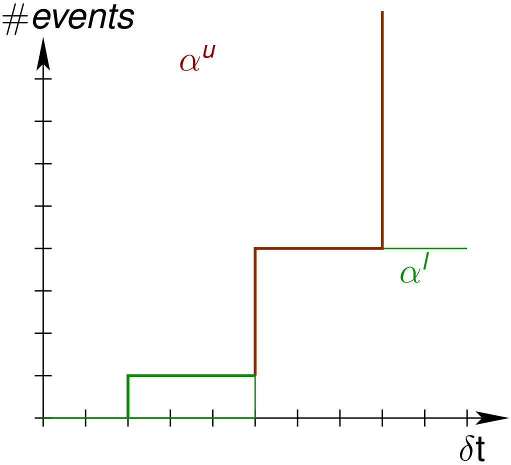
Towards an Algorithm for Causality Closure

- First idea: deconvolution ($\oslash$, $\overline{\oslash}$) to remove forbidden regions
- Insufficient:



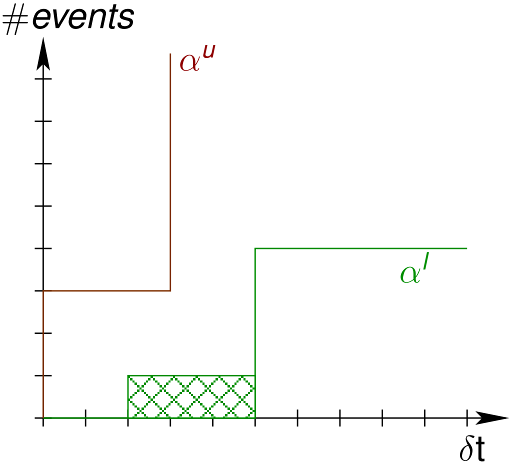
Towards an Algorithm for Causality Closure

- First idea: deconvolution ($\oslash$, $\overline{\oslash}$) to remove forbidden regions
- Insufficient:



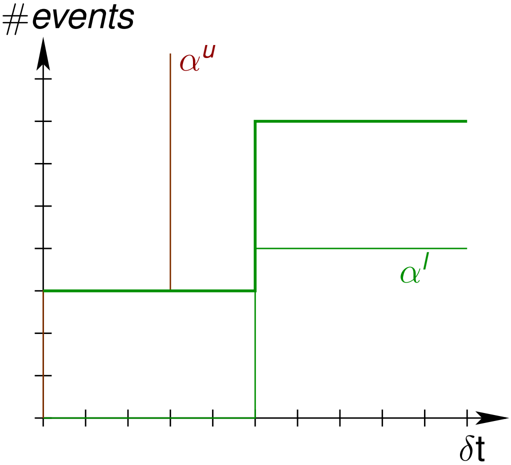
Towards an Algorithm for Causality Closure

- First idea: deconvolution ($\oslash$, $\overline{\oslash}$) to remove forbidden regions
- Insufficient:



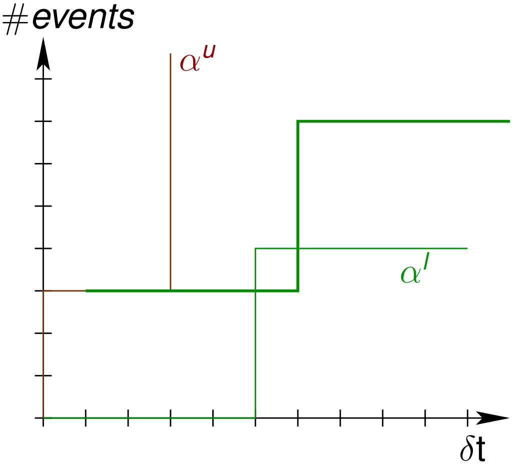
Towards an Algorithm for Causality Closure

- First idea: deconvolution ($\oslash$, $\overline{\oslash}$) to remove forbidden regions
- Insufficient:



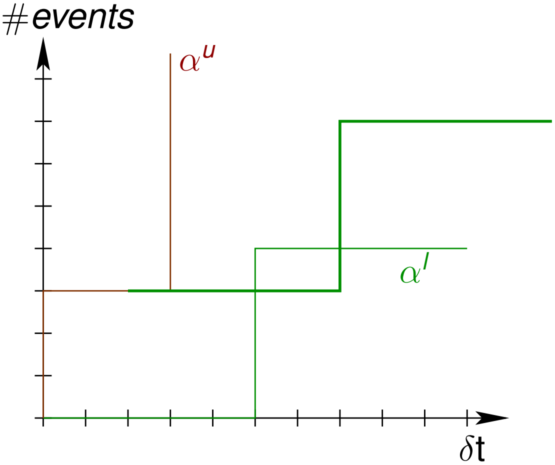
Towards an Algorithm for Causality Closure

- First idea: deconvolution ($\oslash$, $\overline{\oslash}$) to remove forbidden regions
- Insufficient:



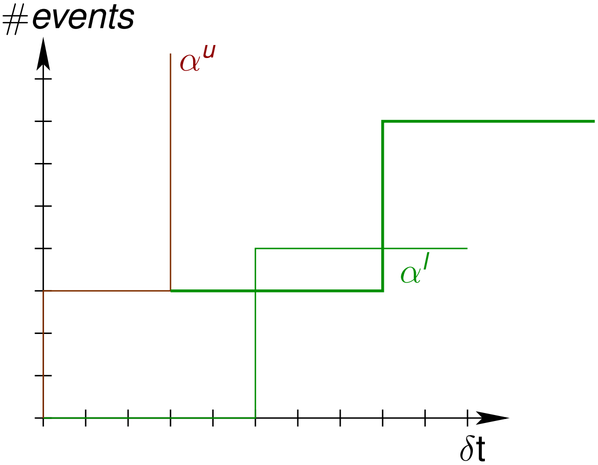
Towards an Algorithm for Causality Closure

- First idea: deconvolution ($\oslash$, $\overline{\oslash}$) to remove forbidden regions
- Insufficient:



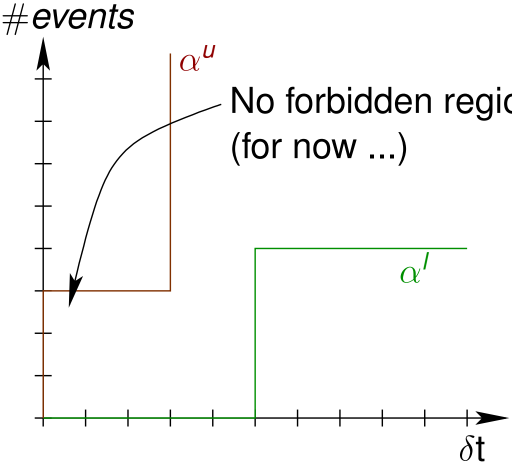
Towards an Algorithm for Causality Closure

- First idea: deconvolution ($\oslash$, $\overline{\oslash}$) to remove forbidden regions
- Insufficient:



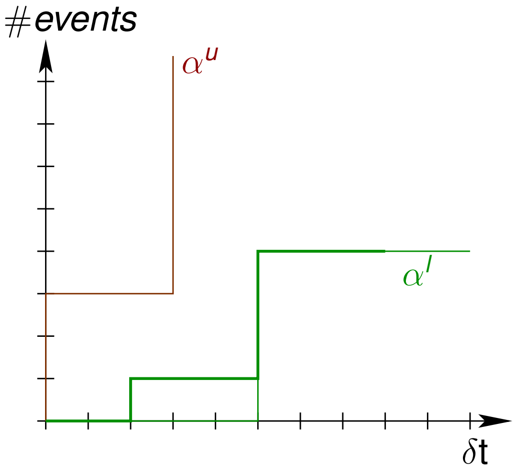
Towards an Algorithm for Causality Closure

- First idea: deconvolution ($\oslash$, $\overline{\oslash}$) to remove forbidden regions
- Insufficient:



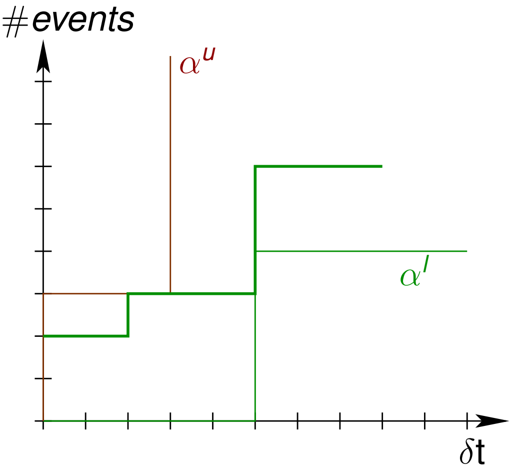
Towards an Algorithm for Causality Closure

- First idea: deconvolution ($\oslash$, $\overline{\oslash}$) to remove forbidden regions
- Insufficient:



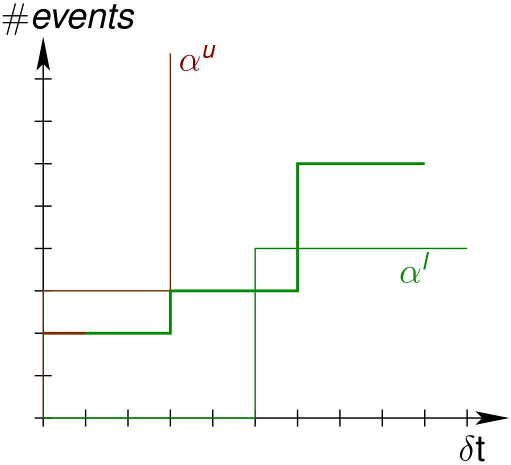
Towards an Algorithm for Causality Closure

- First idea: deconvolution ($\oslash$, $\overline{\oslash}$) to remove forbidden regions
- Insufficient:



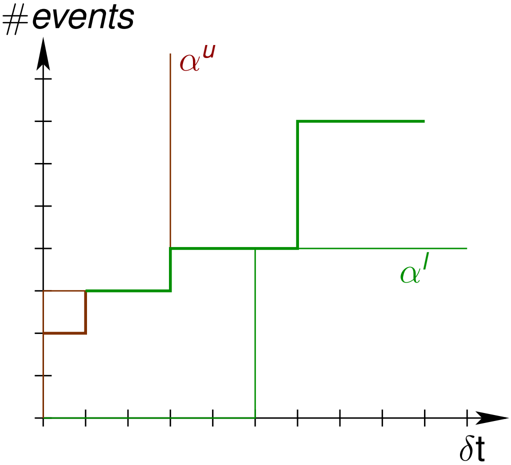
Towards an Algorithm for Causality Closure

- First idea: deconvolution ($\oslash$, $\overline{\oslash}$) to remove forbidden regions
- Insufficient:



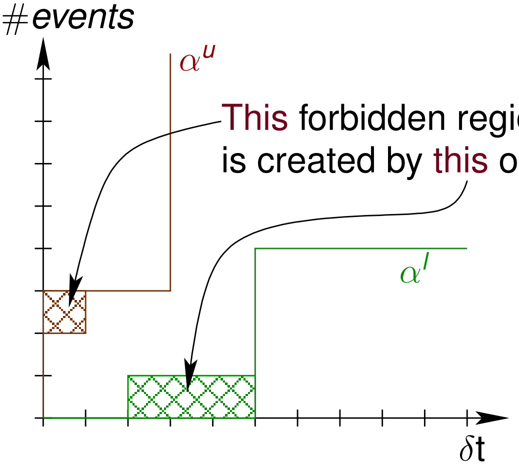
Towards an Algorithm for Causality Closure

- First idea: deconvolution ($\oslash$, $\overline{\oslash}$) to remove forbidden regions
- Insufficient:



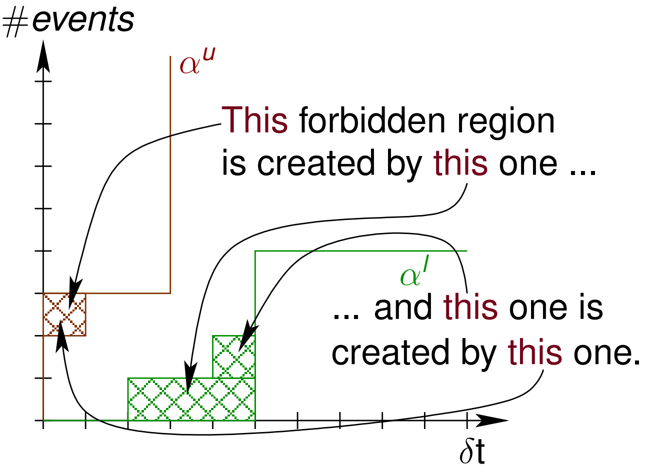
Towards an Algorithm for Causality Closure

- First idea: deconvolution ($\oslash$, $\overline{\oslash}$) to remove forbidden regions
- Insufficient:



Towards an Algorithm for Causality Closure

- First idea: deconvolution ($\oslash$, $\overline{\oslash}$) to remove forbidden regions
- Insufficient:



Towards an Algorithm for Causality Closure

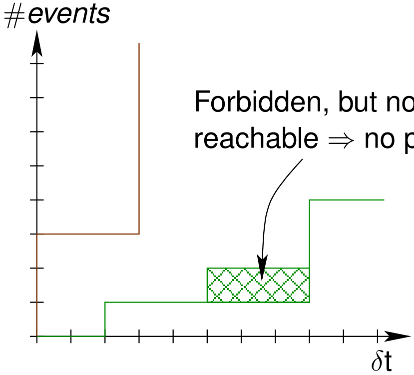
- Second idea: Curves without forbidden regions are (?) causal, let's iterate forbidden region removal until fix-point and we're done (?)

Towards an Algorithm for Causality Closure

- Second idea: Curves without forbidden regions are (?) causal, let's iterate forbidden region removal until fix-point and we're done (?)
- Issue 1: Will it terminate?

Towards an Algorithm for Causality Closure

- Second idea: Curves without forbidden regions are (?) causal, let's iterate forbidden region removal until fix-point and we're done (?)
- Issue 1: Will it terminate?
- Issue 2: some causal curves do have forbidden regions!



Sub/super Additivity and Unreachable Regions

Another (Well Known) Issue

- $\alpha^l(\delta_1 + \delta_2)$ = minimum number of events in any time window of duration $\delta_1 + \delta_2$.
- $\alpha^l(\delta_1) + \alpha^l(\delta_2)$ is another valid bound. It may be better.
- $\Rightarrow$ If so, we say that $\alpha^l(\delta_1 + \delta_2)$ is unreachable.

Sub/super Additivity and Unreachable Regions

Another (Well Known) Issue

- $\alpha^l(\delta_1 + \delta_2)$ = minimum number of events in any time window of duration $\delta_1 + \delta_2$.
- $\alpha^l(\delta_1) + \alpha^l(\delta_2)$ is another valid bound. It may be better.
- $\Rightarrow$ If so, we say that $\alpha^l(\delta_1 + \delta_2)$ is unreachable.
- Technically, (α^u, α^l) have no unreachable regions iff
  - α^l is super-additive,
  - α^u is sub-additive.
- $\overline{\alpha^u}$ = sub-additive closure of α^u
- $\underline{\alpha^l}$ = super-additive closure of α^l

Theorem 1: Causality and Forbidden Region

For sub-additive/super-additive pairs of curves,
Causality $\Leftrightarrow$ Absence of forbidden region

Theorem 1: Causality and Forbidden Region

For sub-additive/super-additive pairs of curves,
Causality $\Leftrightarrow$ Absence of forbidden region

$\Rightarrow$ Causal curve can be obtained by
the fix-point of forbidden region removal
for sub-additive/super-additive curves.

Theorem 2: no Need to Iterate!

- Removing forbidden regions from a sub-additive/super-additive pair of curves
  - Doesn't create new forbidden regions
  - Preserves sub-additive/super-additive property

Theorem 2: no Need to Iterate!

- Removing forbidden regions from a sub-additive/super-additive pair of curves
  - Doesn't create new forbidden regions
  - Preserves sub-additive/super-additive property

$\Rightarrow$ Applying forbidden region removal from $(\overline{\alpha^u}, \underline{\alpha^l})$ gives a causal pair of curves.

$\left(\overline{\alpha^l} \oslash \underline{\alpha^u}, \overline{\alpha^u} \overline{\oslash} \underline{\alpha^l}\right)$ is the causality closure.

Theorem 2: no Need to Iterate!

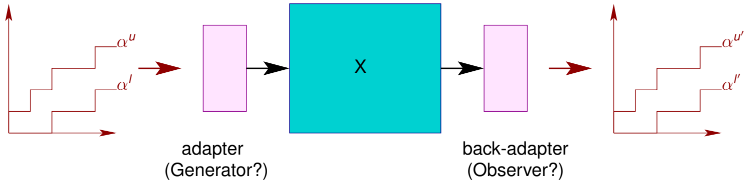
- Removing forbidden regions from a sub-additive/super-additive pair of curves
  - Doesn't create new forbidden regions
  - Preserves sub-additive/super-additive property

$\Rightarrow$ Applying forbidden region removal from $(\overline{\alpha^u}, \underline{\alpha^l})$ gives a causal pair of curves.

$\left(\overline{\alpha^l} \oslash \underline{\alpha^u}, \overline{\alpha^u} \,\overline{\oslash}\, \underline{\alpha^l}\right)$ is the causality closure.

$\rightsquigarrow$ Implementable in any framework implementing $\overline{\alpha}$, $\underline{\alpha}$, $\oslash$ and $\overline{\oslash}$.

Connection from RTC to X, Revisited



- Compute the causality closure of (α^u, α^l) beforehand $\Rightarrow$ avoids deadlocks in the generator (and spurious counter-examples in proofs)
- Compute the causality closure of $(\alpha^{u'}, \alpha^{l'})$ afterwards $\Rightarrow$ may increase precision
- (Same applies for service curves)

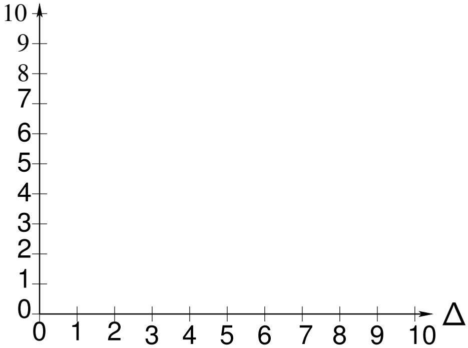
Summary

The $\mathcal{U}pac$ Class of Curves

- Use: ac2lus (bridge between RTC and the Lustre language)
- Goals:
  - Machine-representable
  - Efficient encoding into Lustre
  - Reasonably expressive

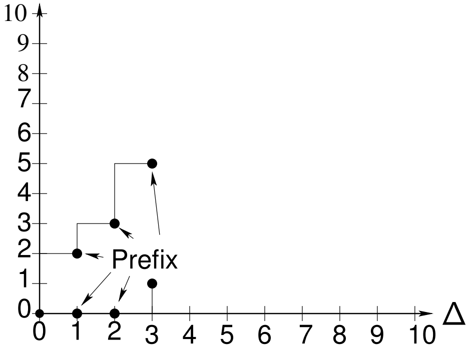
The $\mathcal{U}pac$ Class of Curves

- Use: ac2lus (bridge between RTC and the Lustre language)
- Definition: Ultimately piecewise-affine curves
  - Finite prefix = set of integer points
  - Long-term rate = affine pieces



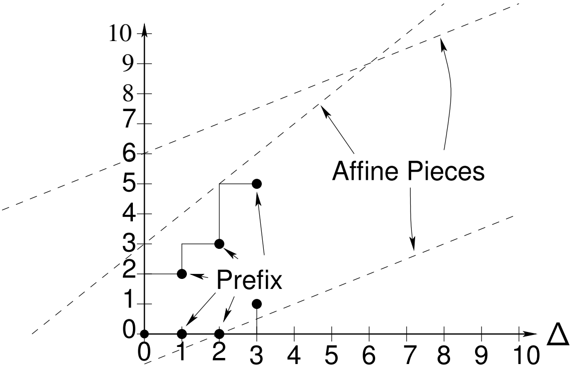
The $\mathcal{U}pac$ Class of Curves

- Use: ac2lus (bridge between RTC and the Lustre language)
- Definition: Ultimately piecewise-affine curves
  - Finite prefix = set of integer points
  - Long-term rate = affine pieces



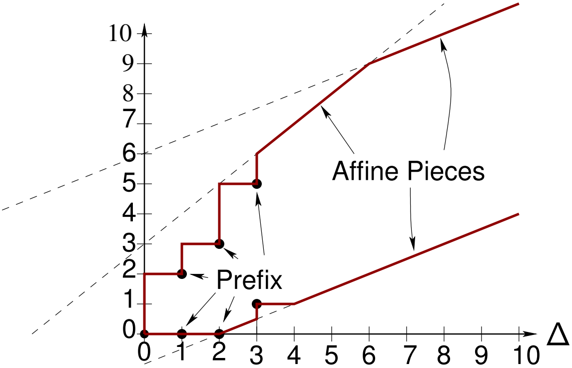
The $\mathcal{U}pac$ Class of Curves

- Use: ac2lus (bridge between RTC and the Lustre language)
- Definition: Ultimately piecewise-affine curves
  - Finite prefix = set of integer points
  - Long-term rate = affine pieces



The $\mathcal{U}pac$ Class of Curves

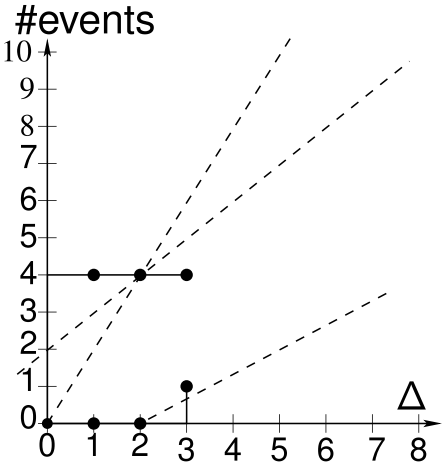
- Use: ac2lus (bridge between RTC and the Lustre language)
- Definition: Ultimately piecewise-affine curves
  - Finite prefix = set of integer points
  - Long-term rate = affine pieces



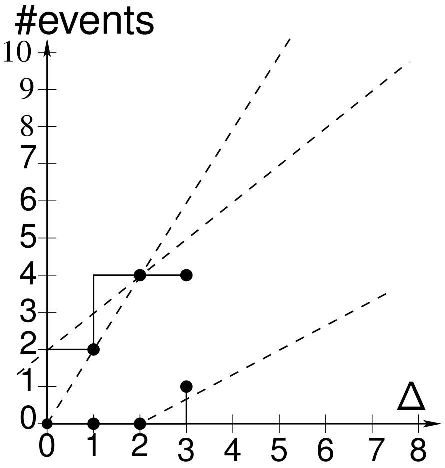
Causality Closure in $\mathcal{U}pac$

- Reminder: Causality closure implementable with $\overline{\alpha}$, $\underline{\alpha}$, $\oslash$ and $\overline{\oslash}$
- Problem: How to implement these operators efficiently?
- 2 steps algorithm:
  1. Normalization: compute $(\overline{\alpha^u}, \underline{\alpha^l})$ and prepare it for step 2
  2. Bounded version of $\oslash$ and $\overline{\oslash}$

Normalization of curves in $\mathcal{U}pac$

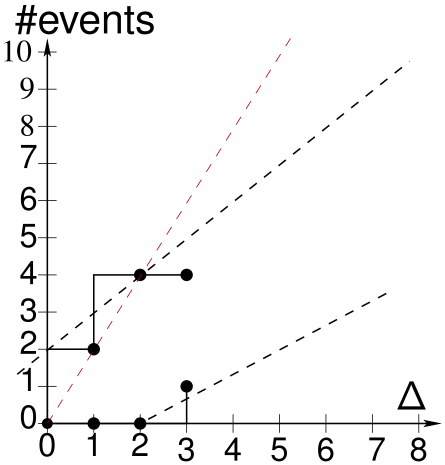


Normalization of curves in $\mathcal{U}pac$



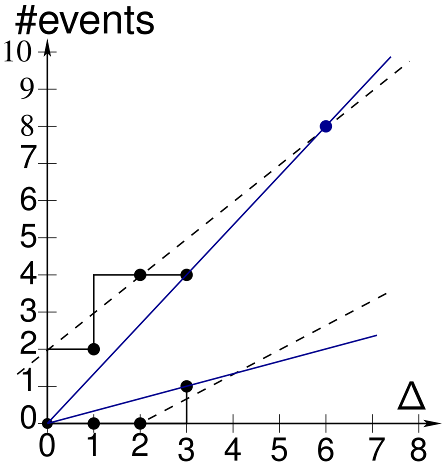
1. Make sure all points are below the affine pieces

Normalization of curves in $\mathcal{U}pac$



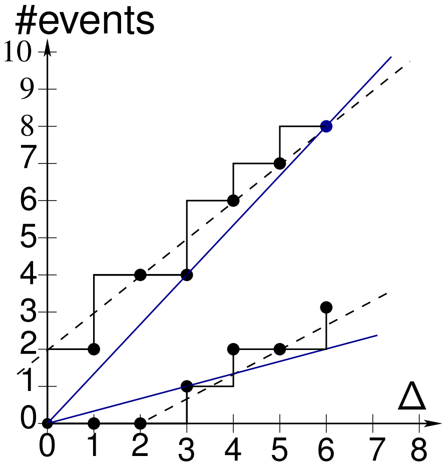
1. Make sure all points are below the affine pieces
2. Remove useless affine pieces

Normalization of curves in $\mathcal{U}pac$



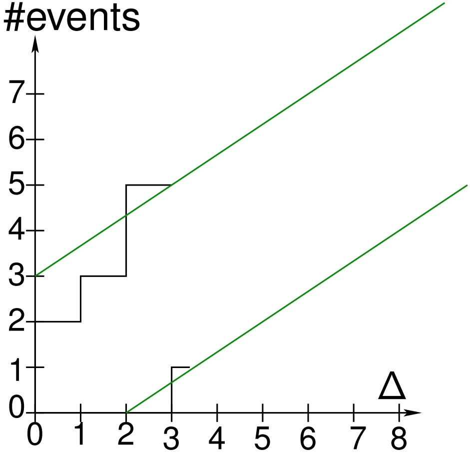
1. Make sure all points are below the affine pieces
2. Remove useless affine pieces
3. Find the point from which the prefix is dominated by affine pieces

Normalization of curves in $\mathcal{U}pac$

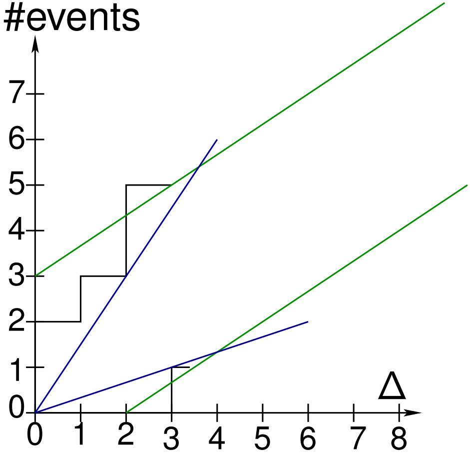


1. Make sure all points are below the affine pieces
2. Remove useless affine pieces
3. Find the point from which the prefix is dominated by affine pieces
4. Extend the prefix until this point

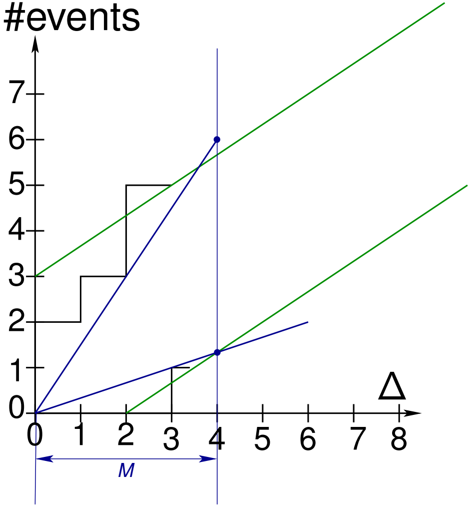
Bounded version of $\oslash$ and $\overline{\oslash}$



Bounded version of $\oslash$ and $\overline{\oslash}$

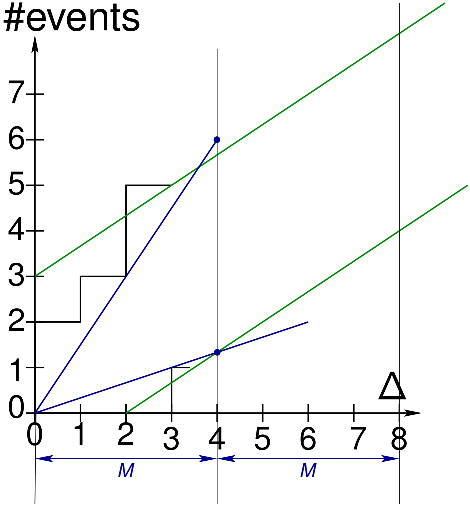


Bounded version of $\oslash$ and $\overline{\oslash}$



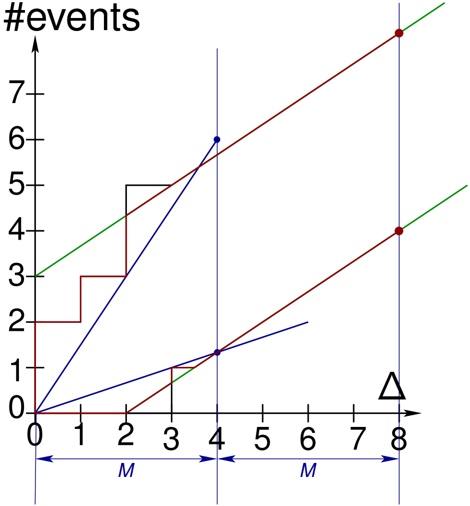
1. M = abscissa of intersection between *slope* of the prefix and affine pieces

Bounded version of $\oslash$ and $\overline{\oslash}$



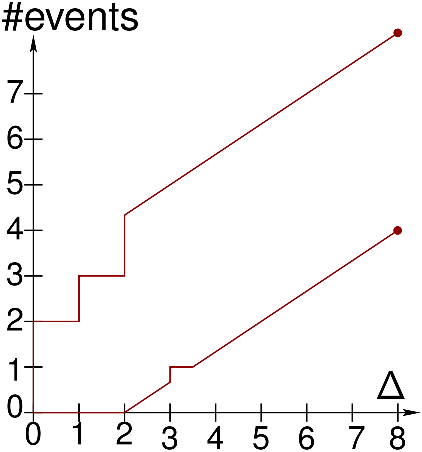
1. M = abscissa of intersection between *slope* of the prefix and affine pieces
2. $2M$ = last point to look at computing $\oslash$ and $\overline{\oslash}$

Bounded version of $\oslash$ and $\overline{\oslash}$



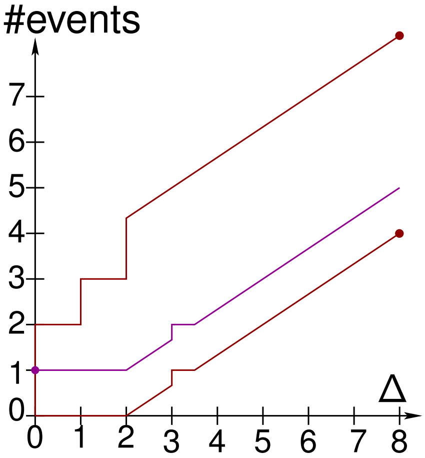
1. M = abscissa of intersection between *slope* of the prefix and affine pieces
2. $2M$ = last point to look at computing $\oslash$ and $\overline{\oslash}$

Bounded version of $\oslash$ and $\overline{\oslash}$



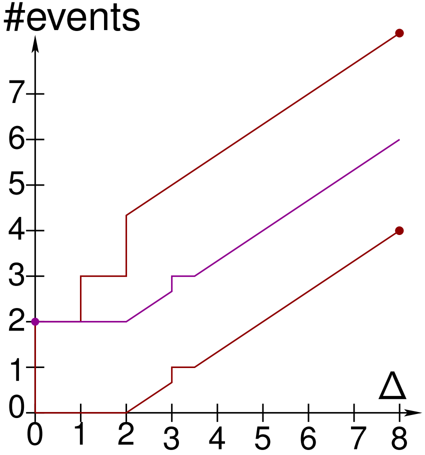
1. M = abscissa of intersection between *slope* of the prefix and affine pieces
2. $2M$ = last point to look at computing $\oslash$ and $\overline{\oslash}$
3. Compute $\overline{\alpha^l} \oslash|_{2M} \underline{\alpha^u}$

Bounded version of $\oslash$ and $\overline{\oslash}$



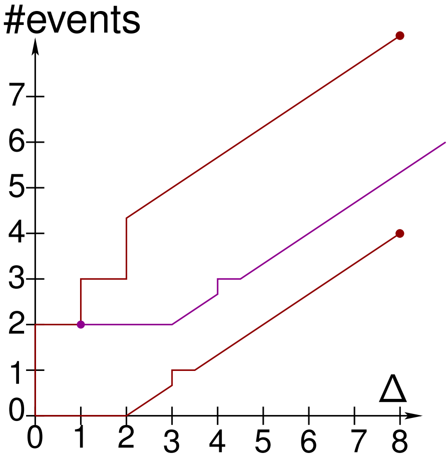
1. M = abscissa of intersection between *slope* of the prefix and affine pieces
2. $2M$ = last point to look at computing $\oslash$ and $\overline{\oslash}$
3. Compute $\overline{\alpha^l} \oslash|_{2M} \underline{\alpha^u}$

Bounded version of $\oslash$ and $\overline{\oslash}$



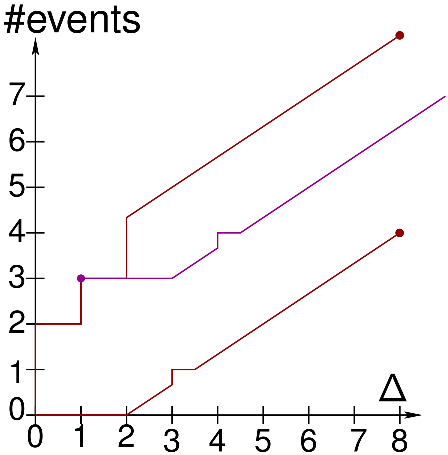
1. M = abscissa of intersection between *slope* of the prefix and affine pieces
2. $2M$ = last point to look at computing $\oslash$ and $\overline{\oslash}$
3. Compute $\overline{\alpha^l} \oslash|_{2M} \underline{\alpha^u}$

Bounded version of $\oslash$ and $\overline{\oslash}$



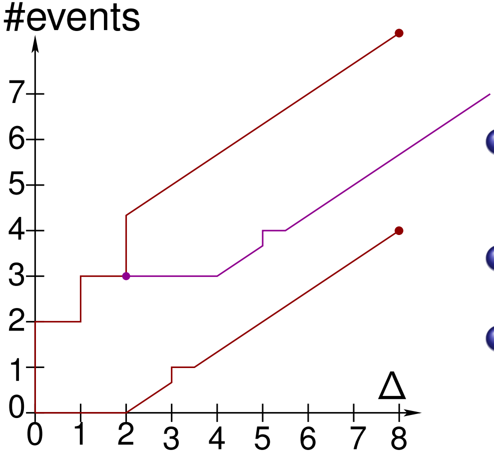
1. M = abscissa of intersection between *slope* of the prefix and affine pieces
2. $2M$ = last point to look at computing $\oslash$ and $\overline{\oslash}$
3. Compute $\overline{\alpha^l} \oslash|_{2M} \underline{\alpha^u}$

Bounded version of $\oslash$ and $\overline{\oslash}$



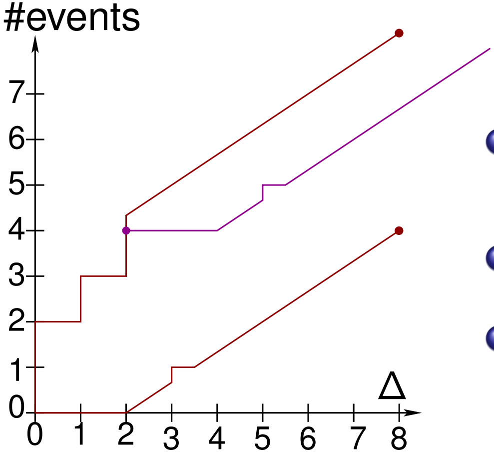
1. M = abscissa of intersection between *slope* of the prefix and affine pieces
2. $2M$ = last point to look at computing $\oslash$ and $\overline{\oslash}$
3. Compute $\overline{\alpha^l} \oslash|_{2M} \underline{\alpha^u}$

Bounded version of $\oslash$ and $\overline{\oslash}$



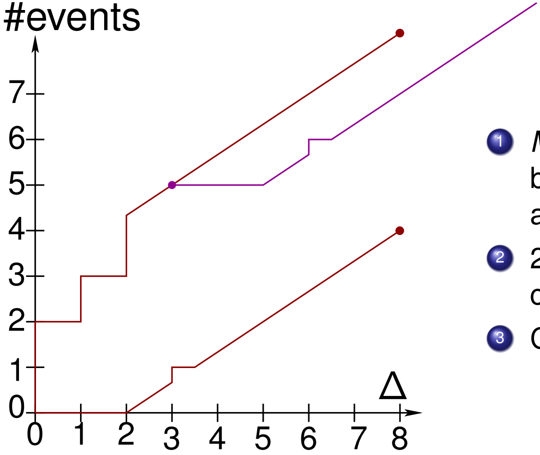
1. M = abscissa of intersection between *slope* of the prefix and affine pieces
2. $2M$ = last point to look at computing $\oslash$ and $\overline{\oslash}$
3. Compute $\overline{\alpha^l} \oslash|_{2M} \underline{\alpha^u}$

Bounded version of $\oslash$ and $\overline{\oslash}$



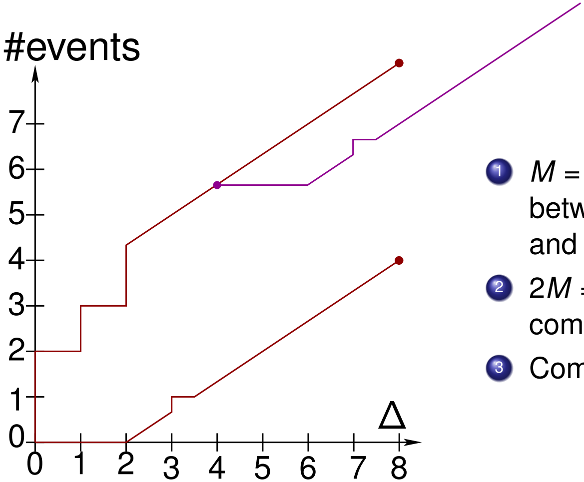
1. M = abscissa of intersection between *slope* of the prefix and affine pieces
2. $2M$ = last point to look at computing $\oslash$ and $\overline{\oslash}$
3. Compute $\overline{\alpha^l} \oslash|_{2M} \underline{\alpha^u}$

Bounded version of $\oslash$ and $\overline{\oslash}$



1. M = abscissa of intersection between *slope* of the prefix and affine pieces
2. $2M$ = last point to look at computing $\oslash$ and $\overline{\oslash}$
3. Compute $\overline{\alpha^l} \oslash|_{2M} \underline{\alpha^u}$

Bounded version of $\oslash$ and $\overline{\oslash}$



1. M = abscissa of intersection between *slope* of the prefix and affine pieces
2. $2M$ = last point to look at computing $\oslash$ and $\overline{\oslash}$
3. Compute $\overline{\alpha^l} \oslash|_{2M} \underline{\alpha^u}$

Summary

Summary of Contributions

- Reminders about causality closure (see [TACAS2010] for details)
- (Algorithmic) Application to $\mathcal{U}pac$
- Implemented in ac2lus
  - Causality closure on input, before running proofs
    $\Rightarrow$ avoids spurious counter-examples
  - Causality closure on outputs
    $\Rightarrow$ can increase precision

Details I've spared you ...

- Corner-cases (no relevant affine piece on one or two curves)
- Correctness proofs *(A bit more tricky than expected ;-))*

Questions?

Thank You!

Backup Slide : Main Theorems About Causality

$$
\begin{array}{ccc}
\begin{array}{c} \alpha^l = \alpha^l \oslash \alpha^u \\ \text{and} \\ \alpha^u = \alpha^u \,\overline{\oslash}\, \alpha^l \end{array} & \Longrightarrow & (\alpha^u, \alpha^l) \text{ is causal} \\
\Downarrow & \text{⤢} & \Updownarrow \\
\begin{array}{c} \underline{\alpha^l} = \underline{\alpha^l} \oslash \overline{\alpha^u} \\ \text{and} \\ \overline{\alpha^u} = \overline{\alpha^u} \,\overline{\oslash}\, \underline{\alpha^l} \end{array} & \Longleftrightarrow & (\overline{\alpha^u}, \underline{\alpha^l}) \text{ is causal}
\end{array}
$$

Backup Slide: Bounded version of $\oslash$ and $\overline{\oslash}$

$$\overline{\alpha^l} \oslash\big|_{2M} \underline{\alpha^u} = \inf_{t \in [0,M]} \{\alpha^u(\Delta + t) - \alpha^l(t)\}$$

$$\left(= \mathbb{C}\big|_M^u \left(\alpha^u, \alpha^l\right)(\Delta) \text{ in the paper}\right)$$

$$\overline{\alpha^u}\, \overline{\oslash}\big|_{2M} \underline{\alpha^l} = \sup_{t \in [0,M]} \{\alpha^l(\Delta + t) - \alpha^u(t)\}$$

$$\left(= \mathbb{C}\big|_M^l \left(\alpha^u, \alpha^l\right)(\Delta) \text{ in the paper}\right)$$