Sequencing in the Run-Time Event Calculus

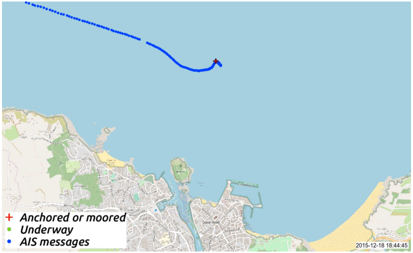
Periklis Mantenoglou[1]   Alexander Artikis[2,3]

[1]Örebro University, Sweden
[2]NCSR Demokritos, Greece
[3]University of Piraeus, Greece

Run-Time Event Calculus (RTEC)

- A composite event recognition framework that is formal, expressive and efficient.

$$
\begin{aligned}
&\mathbf{holdsFor}(\mathit{anchoredOrMoored}(\mathit{Vl}) = \text{true}, I) \leftarrow\\
&\quad \mathbf{holdsFor}(\mathit{stopped}(\mathit{Vl}) = \mathit{farFromPorts}, I_{sf}),\\
&\quad \mathbf{holdsFor}(\mathit{withinArea}(\mathit{Vl}, \mathit{anchorage}) = \text{true}, I_{a}),\\
&\quad \mathbf{intersect\_all}([I_{sf}, I_{a}], I_{sfa}),\\
&\quad \mathbf{holdsFor}(\mathit{stopped}(\mathit{Vl}) = \mathit{nearPorts}, I_{sn}),\\
&\quad \mathbf{union\_all}([I_{sfa}, I_{sn}], I).
\end{aligned}
$$

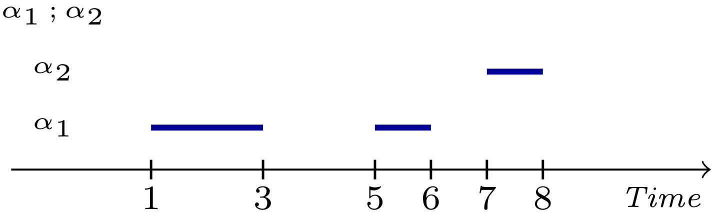


Artikis et al., An Event Calculus for Event Recognition. TKDE, 2015.
https://github.com/aartikis/rtec

Sequencing Operator for RTEC

Key Ingredients:

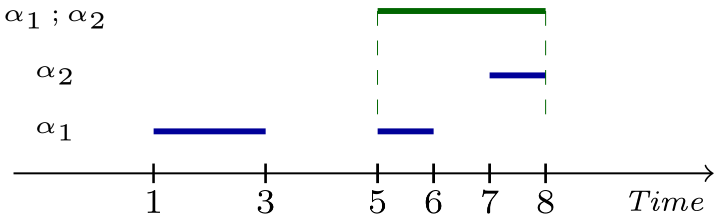
- Adjacency function: requires successive intervals.
- Composition function $\otimes$: constructs the output intervals.



Sequencing Operator for RTEC

Key Ingredients:

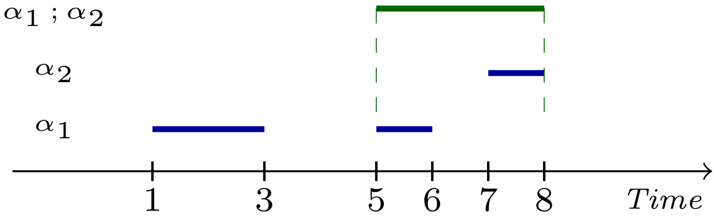
- Adjacency function: requires successive intervals.
- Composition function $\otimes$: constructs the output intervals.



Sequencing Operator for RTEC

Key Ingredients:

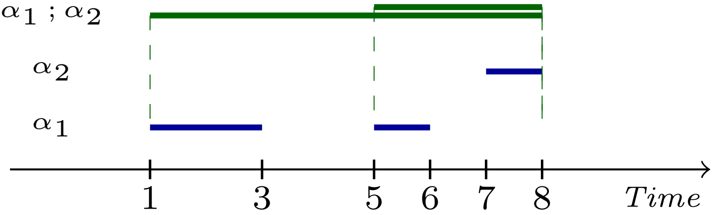
- Adjacency function: requires successive intervals.
- Composition function $\otimes$: constructs the output intervals.



- Our sequencing operator produces maximal disjoint intervals.

Maximal Disjoint Interval Requirement

- Sequencing operators of complex event recognition frameworks do not meet the maximal disjoint interval requirement.



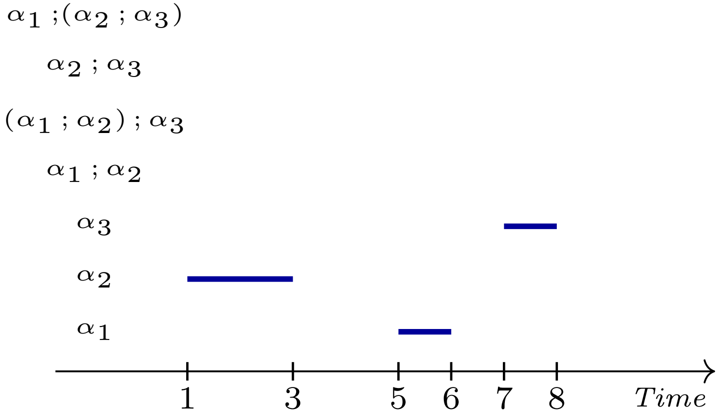
White W. M., Riedewald M., Gehrke J., Demers A. J.: What is “next” in event processing. PODS, 263–272. ACM, 2007.

Bucchi M., Grez A., Quintana A., Riveros C., Vansummeren S.: CORE: a complex event recognition engine. Proc. VLDB Endow., 15(9):1951–1964, 2022.

Alevizos E., Artikis A., Paliouras G.: Complex event recognition with symbolic register transducers. Proc. VLDB Endow., 17(11):3165–3177, 2024.

Associativity Requirement

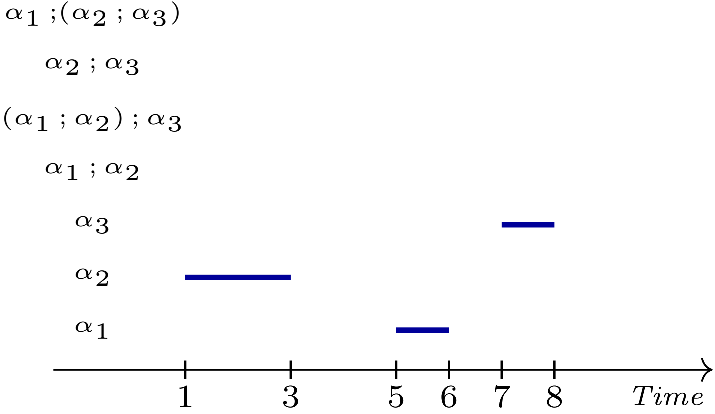
- Associativity: required for optimising hierarchial patterns.



Associativity Requirement

- Associativity: required for optimising hierarchial patterns.

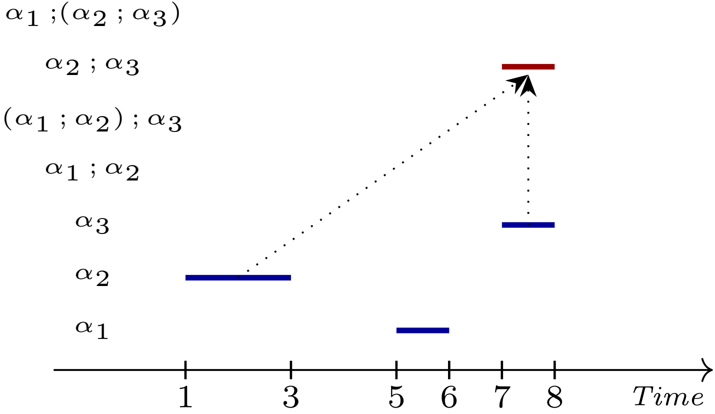
For a sequencing operator where $i_1 \otimes i_2 = i_2$:



Associativity Requirement

- Associativity: required for optimising hierarchial patterns.

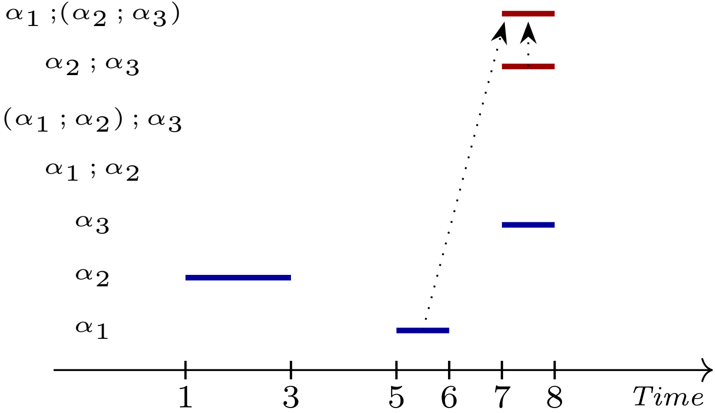
For a sequencing operator where $i_1 \otimes i_2 = i_2$:



Associativity Requirement

- Associativity: required for optimising hierarchial patterns.

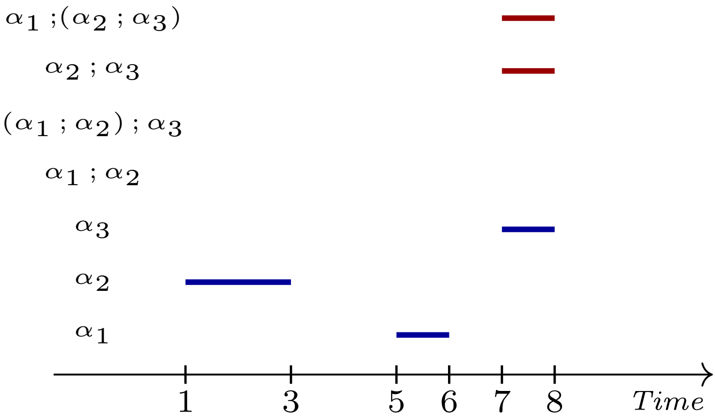
For a sequencing operator where $i_1 \otimes i_2 = i_2$:



Associativity Requirement

- Associativity: required for optimising hierarchial patterns.

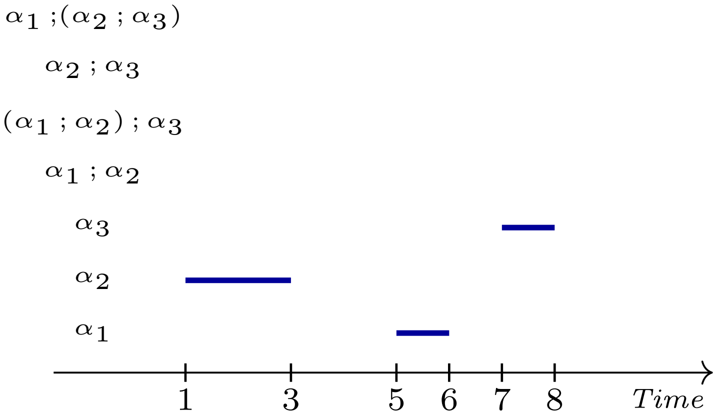
For a sequencing operator where $i_1 \otimes i_2 = i_2$:



- Associativity is violated → Activities $(\alpha_1 ; \alpha_2) ; \alpha_3$ and $\alpha_1 ;(\alpha_2 ; \alpha_3)$ are assigned different lists of intervals.

Sequencing Operator for RTEC

- Associativity: required for optimising hierarchial patterns.



- Our sequencing operator satisfies associativity → Activities $(\alpha_1 ; \alpha_2) ; \alpha_3$ and $\alpha_1 ; (\alpha_2 ; \alpha_3)$ are assigned the same list of intervals.

RTEC$_S$: RTEC with Sequencing

$$
\begin{aligned}
&\textbf{holdsFor}(\mathit{fishingTripStart}(Vl) = \mathsf{true},\, I) \leftarrow \\
&\quad \textbf{holdsFor}(\mathit{anchoredOrMoored}(Vl) = \mathsf{true},\, I_{am}), \\
&\quad \textbf{holdsFor}(\mathit{withinArea}(Vl, \mathit{fishing}) = \mathsf{true},\, I_f), \\
&\quad \mathsf{seq}(I_{am}, I_f, I).
\end{aligned}
$$

RTEC$_S$: RTEC with Sequencing

$$
\begin{aligned}
&\textbf{holdsFor}(\mathit{fishingTripStart}(\mathit{Vl}) = \mathrm{true}, I) \leftarrow \\
&\quad \textbf{holdsFor}(\mathit{anchoredOrMoored}(\mathit{Vl}) = \mathrm{true}, I_{am}), \\
&\quad \textbf{holdsFor}(\mathit{withinArea}(\mathit{Vl}, \mathit{fishing}) = \mathrm{true}, I_{f}), \\
&\quad \mathrm{seq}(I_{am}, I_{f}, I).
\end{aligned}
$$

Semantics

Pattern hierarchies in RTEC$_S$ are locally stratified logic programs.

RTEC$_S$: RTEC with Sequencing

$$
\begin{aligned}
&\textbf{holdsFor}(\mathit{fishingTripStart}(Vl) = \mathsf{true}, I) \leftarrow\\
&\quad \textbf{holdsFor}(\mathit{anchoredOrMoored}(Vl) = \mathsf{true}, I_{am}),\\
&\quad \textbf{holdsFor}(\mathit{withinArea}(Vl, \mathit{fishing}) = \mathsf{true}, I_f),\\
&\quad \mathsf{seq}(I_{am}, I_f, I).
\end{aligned}
$$

Semantics

Pattern hierarchies in RTEC$_S$ are locally stratified logic programs.

Correctness

If I_1 and I_2 contain the maximal intervals of activities a_1 and a_2, then $\mathsf{seq}(I_1, I_2, I)$ computes the list of maximal intervals I of $a_1\,;a_2$.

RTEC$_\mathsf{S}$: RTEC with Sequencing

$$
\begin{aligned}
&\textbf{holdsFor}(\mathit{fishingTripStart}(\mathit{Vl}) = \mathsf{true},\, I) \leftarrow\\
&\quad \textbf{holdsFor}(\mathit{anchoredOrMoored}(\mathit{Vl}) = \mathsf{true},\, I_{am}),\\
&\quad \textbf{holdsFor}(\mathit{withinArea}(\mathit{Vl}, \mathit{fishing}) = \mathsf{true},\, I_f),\\
&\quad \mathsf{seq}(I_{am}, I_f, I).
\end{aligned}
$$

Semantics

Pattern hierarchies in RTEC$_\mathsf{S}$ are locally stratified logic programs.

Correctness

If I_1 and I_2 contain the maximal intervals of activities a_1 and a_2, then $\mathsf{seq}(I_1, I_2, I)$ computes the list of maximal intervals I of $a_1\,;a_2$.

Complexity

RTEC_S evaluates $\mathsf{seq}(I_1, I_2, I)$ in time linear to the number of intervals in I_1 and I_2.

Experiments: Sequencing N Activities

Parameters		Reasoning Time (ms)		Computed Intervals	
N	D	RTEC$_S$	CORE	RTEC$_S$	CORE
3	10K	19	1K	500	148K
6	10K	23	7K	402	669K
12	10K	30	2K	35	109K
3	50K	82	109K	500	19M
6	50K	87	$>$600K	500	$>$30M
12	50K	107	$>$600K	500	$>$30M

- Pattern: $\alpha_1(Id)\,;\alpha_2(Id)\,;\ldots;\alpha_N(Id)$
  → A sequence of N different unary activities with the same argument.

Code, Data & Temporal Specifications: https://github.com/Periklismant/rtecs_ecai25_supplementary

Experiments: Sequencing N Activities

Parameters		Reasoning Time (ms)		Computed Intervals	
N	D	RTEC$_S$	CORE	RTEC$_S$	CORE
3	10K	19	1K	500	148K
6	10K	23	7K	402	669K
12	10K	30	2K	35	109K
3	50K	82	109K	500	19M
6	50K	87	>600K	500	>30M
12	50K	107	>600K	500	>30M

- Pattern: $\alpha_1(Id)\,;\alpha_2(Id)\,;\ldots;\alpha_N(Id)$
  → A sequence of N different unary activities with the same argument.

Code, Data & Temporal Specifications: https://github.com/Periklismant/rtecs_ecai25_supplementary

Experiments: Sequencing N Activities

Parameters		Reasoning Time (ms)		Computed Intervals	
N	D	RTEC$_S$	CORE	RTEC$_S$	CORE
3	10K	19	1K	500	148K
6	10K	23	7K	402	669K
12	10K	30	2K	35	109K
3	50K	82	109K	500	19M
6	50K	87	>600K	500	>30M
12	50K	107	>600K	500	>30M

- Pattern: $\alpha_1(Id)\,;\alpha_2(Id)\,;\ldots;\alpha_N(Id)$
  → A sequence of N different unary activities with the same argument.

Code, Data & Temporal Specifications: https://github.com/Periklismant/rtecs_ecai25_supplementary

Experiments: Sequencing with Hierarchical Patterns

	Reasoning Time (ms)			Computed Intervals		
N	RTEC$_{\mathrm{S}}$	RTEC$_{\mathrm{S\text{-}f}}$	CORE	RTEC$_{\mathrm{S}}$	RTEC$_{\mathrm{S\text{-}f}}$	CORE
3	31	39	2K	1.5K	1.5K	193K
6	63	84	18K	6.6K	6.6K	1.5M
12	240	400	48K	12.4K	12.4K	1.6M

- Patterns: $\alpha_1(Id)\,;\alpha_2(Id)\,;\ldots;\alpha_N(Id)$ and all its possible subpatterns, e.g.:
  - $\alpha_1(Id)\,;\alpha_2(Id)\,;\ldots;\alpha_{N-1}(Id)$
  - $\alpha_{N-2}(Id)\,;\alpha_{N-1}(Id)\,;\alpha_N(Id)$
  - etc.
- For $N = 12$, we have a hierarchy of 66 patterns.

Code, Data & Temporal Specifications: `https://github.com/Periklismant/rtecs_ecai25_supplementary`

Experiments: Sequencing with Hierarchical Patterns

	Reasoning Time (ms)			Computed Intervals		
N	RTEC$_{\mathrm{S}}$	RTEC$_{\mathrm{S\text{-}f}}$	CORE	RTEC$_{\mathrm{S}}$	RTEC$_{\mathrm{S\text{-}f}}$	CORE
3	31	39	2K	1.5K	1.5K	193K
6	63	84	18K	6.6K	6.6K	1.5M
12	240	400	48K	12.4K	12.4K	1.6M

- Patterns: $\alpha_1(Id)\,;\alpha_2(Id)\,;\ldots;\alpha_N(Id)$ and all its possible subpatterns, e.g.:
    - $\alpha_1(Id)\,;\alpha_2(Id)\,;\ldots;\alpha_{N-1}(Id)$
    - $\alpha_{N-2}(Id)\,;\alpha_{N-1}(Id)\,;\alpha_N(Id)$
    - etc.
- For $N = 12$, we have a hierarchy of 66 patterns.

Code, Data & Temporal Specifications: https://github.com/Periklismant/rtecs_ecai25_supplementary

Experiments: Sequencing with Hierarchical Patterns

	Reasoning Time (ms)			Computed Intervals		
N	RTEC$_{\text{S}}$	RTEC$_{\text{S-f}}$	CORE	RTEC$_{\text{S}}$	RTEC$_{\text{S-f}}$	CORE
3	31	39	2K	1.5K	1.5K	193K
6	63	84	18K	6.6K	6.6K	1.5M
12	240	400	48K	12.4K	12.4K	1.6M

- Patterns: $\alpha_1(Id)\,;\alpha_2(Id)\,;\ldots;\alpha_N(Id)$ and all its possible subpatterns, e.g.:
  - $\alpha_1(Id)\,;\alpha_2(Id)\,;\ldots;\alpha_{N-1}(Id)$
  - $\alpha_{N-2}(Id)\,;\alpha_{N-1}(Id)\,;\alpha_N(Id)$
  - etc.
- For $N = 12$, we have a hierarchy of 66 patterns.

Code, Data & Temporal Specifications: `https://github.com/Periklismant/rtecs_ecai25_supplementary`

Experiments: Real Maritime Dataset

Window Size		Reasoning Time (sec)	Computed Intervals
Days	D	RTEC$_S$	RTEC$_S$
1	73K	2	18K
2	145K	6	33K
4	272K	14	61K
8	545K	32	119K
16	1M	79	236K

- Patterns: Composite Maritime Activities
- Dataset:
  - 18M position signals
  - 5K vessels
  - area near the port of Brest, France
  - 6 months of data

Code, Data & Temporal Specifications: https://github.com/Periklismant/rtecs_ecai25_supplementary

Public Maritime Dataset: https://zenodo.org/records/1167595

Experiments: Real Maritime Dataset

Window Size		Reasoning Time (sec)	Computed Intervals
Days	D	RTEC$_S$	RTEC$_S$
1	73K	2	18K
2	145K	6	33K
4	272K	14	61K
8	545K	32	119K
16	1M	79	236K

- Patterns: Composite Maritime Activities
- Dataset:
  - 18M position signals
  - 5K vessels
  - area near the port of Brest, France
  - 6 months of data

Code, Data & Temporal Specifications: https://github.com/Periklismant/rtecs_ecai25_supplementary

Public Maritime Dataset: https://zenodo.org/records/1167595

Summary & Future Work

Summary:

- A sequencing operator that is:
  - interval-based,
  - compositional, and
  - associative.

Summary & Future Work

Summary:

- A sequencing operator that is:
  - interval-based,
  - compositional, and
  - associative.
- $\mathsf{RTEC_S}$: stream reasoning with sequencing.

Summary & Future Work

Summary:

- A sequencing operator that is:
  - interval-based,
  - compositional, and
  - associative.
- RTEC$_S$: stream reasoning with sequencing.
- Reproducible empirical evaluation with real and synthetic data.

Summary & Future Work

Summary:

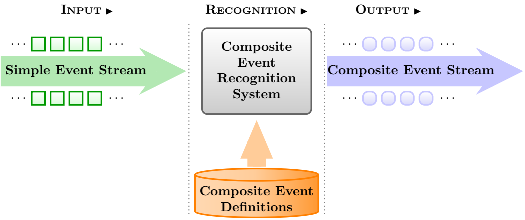
- A sequencing operator that is:
  - interval-based,
  - compositional, and
  - associative.
- $\mathrm{RTEC}_{\mathrm{S}}$: stream reasoning with sequencing.
- Reproducible empirical evaluation with real and synthetic data.

Future Work:

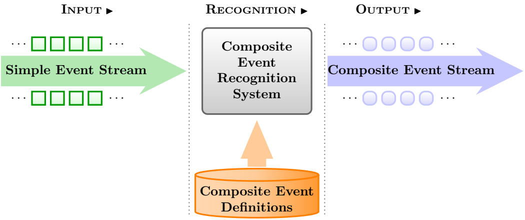
- Windowing.

Appendix

Composite Event Recognition



Composite Event Recognition



INPUT ▸

Simple Event Stream

RECOGNITION ▸

Composite Event Recognition System

Composite Event Definitions

OUTPUT ▸

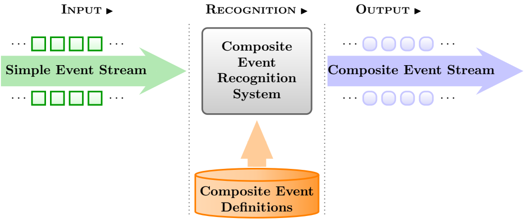
Composite Event Stream

● Under way
★ Trawling
● AIS messages
● Other vessels
Fishing area
2016-02-02 02:02:29

https://cer.iit.demokritos.gr

(maritime situational awareness)

Composite Event Recognition



INPUT ▸

RECOGNITION ▸

OUTPUT ▸

Simple Event Stream

Composite Event Recognition System

Composite Event Stream

Composite Event Definitions

● Under way
★ Trawling
● AIS messages
● Other vessels
Fishing area
2016-02-02 02:02:29

https://cer.iit.demokritos.gr

(maritime situational awareness)

- Composite activity patterns often require sequencing.
- Example: phases of a fishing trip.

Event Calculus

- A logic programming language for representing and reasoning about events and their effects.
- Key components:
  - event (typically instantaneous).
  - fluent: a property that may have different values at different points in time.

Robert A. Kowalski, Marek J. Sergot: A Logic-based Calculus of Events. New Gener. Comput. 4(1): 67-95, 1986.

Event Calculus

- A logic programming language for representing and reasoning about events and their effects.
- Key components:
  - event (typically instantaneous).
  - fluent: a property that may have different values at different points in time.
- Built-in representation of inertia:
  - $F = V$ holds at a particular time-point if $F = V$ has been *initiated* by an event at some earlier time-point, and not *terminated* by another event in the meantime.

Robert A. Kowalski, Marek J. Sergot: A Logic-based Calculus of Events. New Gener. Comput. 4(1): 67-95, 1986.

Event Calculus

- A logic programming language for representing and reasoning about events and their effects.
- Key components:
  - event (typically instantaneous).
  - fluent: a property that may have different values at different points in time.
- Built-in representation of inertia:
  - $F = V$ holds at a particular time-point if $F = V$ has been *initiated* by an event at some earlier time-point, and not *terminated* by another event in the meantime.
- Fluents definitions:
  - succinct and intuitive representations of composite activities.

Robert A. Kowalski, Marek J. Sergot: A Logic-based Calculus of Events. New Gener. Comput. 4(1): 67-95, 1986.

Event Calculus

- A logic programming language for representing and reasoning about events and their effects.
- Key components:
  - event (typically instantaneous).
  - fluent: a property that may have different values at different points in time.
- Built-in representation of inertia:
  - $F = V$ holds at a particular time-point if $F = V$ has been *initiated* by an event at some earlier time-point, and not *terminated* by another event in the meantime.
- Fluents definitions:
  - succinct and intuitive representations of composite activities.
  - inefficient representations for stream reasoning.

Robert A. Kowalski, Marek J. Sergot: A Logic-based Calculus of Events. New Gener. Comput. 4(1): 67-95, 1986.

Run-Time Event Calculus (RTEC)

The Run-Time Event Calculus:

- Composite activity pattern matching over large-scale streams of events.

Artikis A., Sergot M. and Paliouras G., An Event Calculus for Event Recognition. In IEEE Transactions on Knowledge and Data Engineering (TKDE), 27(4), 895–908, 2015.

Run-Time Event Calculus (RTEC)

The Run-Time Event Calculus:

- Composite activity pattern matching over large-scale streams of events.

Inertial Activities:

$$\begin{array}{l}\textbf{initiatedAt}(\mathit{withinArea}(\mathit{Vl}, \mathit{AreaType}) = \mathrm{true}, T) \leftarrow \\ \quad \textbf{happensAt}(\mathit{entersArea}(\mathit{Vl}, \mathit{AreaID}), T), \\ \quad \mathit{areaType}(\mathit{AreaID}, \mathit{AreaType}).\end{array}$$

Artikis A., Sergot M. and Paliouras G., An Event Calculus for Event Recognition. In IEEE Transactions on Knowledge and Data Engineering (TKDE), 27(4), 895–908, 2015.

Run-Time Event Calculus (RTEC)

The Run-Time Event Calculus:

- Composite activity pattern matching over large-scale streams of events.

Inertial Activities:

$$\begin{aligned}&\textbf{initiatedAt}(\mathit{withinArea}(\mathit{Vl}, \mathit{AreaType}) = \text{true}, T) \leftarrow\\ &\quad \textbf{happensAt}(\mathit{entersArea}(\mathit{Vl}, \mathit{AreaID}), T),\\ &\quad \mathit{areaType}(\mathit{AreaID}, \mathit{AreaType}).\end{aligned}$$

$$\begin{aligned}&\textbf{terminatedAt}(\mathit{withinArea}(\mathit{Vl}, \mathit{AreaType}) = \text{true}, T) \leftarrow\\ &\quad \textbf{happensAt}(\mathit{leavesArea}(\mathit{Vl}, \mathit{AreaID}), T),\\ &\quad \mathit{areaType}(\mathit{AreaID}, \mathit{AreaType}).\end{aligned}$$

Artikis A., Sergot M. and Paliouras G., An Event Calculus for Event Recognition. In IEEE Transactions on Knowledge and Data Engineering (TKDE), 27(4), 895–908, 2015.

Run-Time Event Calculus (RTEC)

The Run-Time Event Calculus:

- Composite activity pattern matching over large-scale streams of events.

Statically Determined Activities:

$$
\begin{aligned}
&\textbf{holdsFor}(\mathit{anchoredOrMoored}(\mathit{Vl}) = \mathrm{true}, I) \leftarrow \\
&\quad \textbf{holdsFor}(\mathit{stopped}(\mathit{Vl}) = \mathit{farFromPorts}, I_{sf}), \\
&\quad \textbf{holdsFor}(\mathit{withinArea}(\mathit{Vl}, \mathit{anchorage}) = \mathrm{true}, I_a), \\
&\quad \textbf{intersect\_all}([I_{sf}, I_a], I_{sfa}), \\
&\quad \textbf{holdsFor}(\mathit{stopped}(\mathit{Vl}) = \mathit{nearPorts}, I_{sn}), \\
&\quad \textbf{union\_all}([I_{sfa}, I_{sn}], I).
\end{aligned}
$$

Artikis A., Sergot M. and Paliouras G., An Event Calculus for Event Recognition. In IEEE Transactions on Knowledge and Data Engineering (TKDE), 27(4), 895–908, 2015.

Run-Time Event Calculus (RTEC)

The Run-Time Event Calculus:

- Composite activity pattern matching over large-scale streams of events.
- RTEC has several benefits compared to the state of the art:
  - relational & hierarchical patterns with background knowledge.
  - inertial & statically determined activities.
  - scalability to large real-world scenarios.

Artikis A., Sergot M. and Paliouras G., An Event Calculus for Event Recognition. In IEEE Transactions on Knowledge and Data Engineering (TKDE), 27(4), 895–908, 2015.

Run-Time Event Calculus (RTEC)

The Run-Time Event Calculus:

- Composite activity pattern matching over large-scale streams of events.
- RTEC has several benefits compared to the state of the art:
  - relational & hierarchical patterns with background knowledge.
  - inertial & statically determined activities.
  - scalability to large real-world scenarios.
- However, RTEC does not express sequencing.

Artikis A., Sergot M. and Paliouras G., An Event Calculus for Event Recognition. In IEEE Transactions on Knowledge and Data Engineering (TKDE), 27(4), 895–908, 2015.

Run-Time Event Calculus (RTEC)

The Run-Time Event Calculus:

- Composite activity pattern matching over large-scale streams of events.

- RTEC has several benefits compared to the state of the art:
  - relational & hierarchical patterns with background knowledge.
  - inertial & statically determined activities.
  - scalability to large real-world scenarios.

- However, RTEC does not express sequencing.
- Contribution: a sequencing operator for RTEC that preserves:
  - interval-based semantics for durative activities.

Artikis A., Sergot M. and Paliouras G., An Event Calculus for Event Recognition. In IEEE Transactions on Knowledge and Data Engineering (TKDE), 27(4), 895–908, 2015.

Run-Time Event Calculus (RTEC)

The Run-Time Event Calculus:

- Composite activity pattern matching over large-scale streams of events.
- RTEC has several benefits compared to the state of the art:
  - relational & hierarchical patterns with background knowledge.
  - inertial & statically determined activities.
  - scalability to large real-world scenarios.
- However, RTEC does not express sequencing.
- Contribution: a sequencing operator for RTEC that preserves:
  - interval-based semantics for durative activities.
  - correctness over pattern hierarchies.

Artikis A., Sergot M. and Paliouras G., An Event Calculus for Event Recognition. In IEEE Transactions on Knowledge and Data Engineering (TKDE), 27(4), 895–908, 2015.

Run-Time Event Calculus (RTEC)

The Run-Time Event Calculus:

- Composite activity pattern matching over large-scale streams of events.
- RTEC has several benefits compared to the state of the art:
  - relational & hierarchical patterns with background knowledge.
  - inertial & statically determined activities.
  - scalability to large real-world scenarios.
- However, RTEC does not express sequencing.
- Contribution: a sequencing operator for RTEC that preserves:
  - interval-based semantics for durative activities.
  - correctness over pattern hierarchies.
  - efficiency over large-scale activity streams.

Artikis A., Sergot M. and Paliouras G., An Event Calculus for Event Recognition. In IEEE Transactions on Knowledge and Data Engineering (TKDE), 27(4), 895–908, 2015.

Abstract Sequencing Model

An abstract sequencing model is a tuple $(T, \prec, \mathsf{S}, \otimes)$, where

- T is a set of time-stamps.

White W. M., Riedewald M., Gehrke J., Demers A. J.: What is “next” in event processing. PODS, 263–272. ACM, 2007.

Abstract Sequencing Model

An abstract sequencing model is a tuple $(T, \prec, \mathsf{S}, \otimes)$, where

- T is a set of time-stamps.
- $\prec$ is a partial order on T.

White W. M., Riedewald M., Gehrke J., Demers A. J.: What is "next" in event processing. PODS, 263–272. ACM, 2007.

Abstract Sequencing Model

An abstract sequencing model is a tuple $(T, \prec, \mathsf{S}, \otimes)$, where

- T is a set of time-stamps.
- $\prec$ is a partial order on T.
- $\mathsf{S} : T \times 2^T \rightarrow 2^T$ is a successor function.
  $\mathsf{S}(t, \mathcal{F})$ is the set of immediate successor time-stamps of t in set $\mathcal{F}$.

White W. M., Riedewald M., Gehrke J., Demers A. J.: What is “next” in event processing. PODS, 263–272. ACM, 2007.

Abstract Sequencing Model

An abstract sequencing model is a tuple $(T, \prec, \mathsf{S}, \otimes)$, where

- T is a set of time-stamps.
- $\prec$ is a partial order on T.
- $\mathsf{S} : T \times 2^T \rightarrow 2^T$ is a successor function.
  $\mathsf{S}(t, \mathcal{F})$ is the set of immediate successor time-stamps of t in set $\mathcal{F}$.
- $\otimes : T \times T \rightarrow T$ is a composition operator.
  $t_1 \otimes t_2$ is the time-stamp of the sequence of two activities with time-stamps t_1 and t_2.

White W. M., Riedewald M., Gehrke J., Demers A. J.: What is “next” in event processing. PODS, 263–272. ACM, 2007.

Sequencing in Cayuga

The sequencing model of Cayuga is $(T_{cy}, \prec_{cy}, \mathsf{S}_{cy}, \otimes_{cy})$, where

- T_{cy}: intervals over the positive integers.

White W. M., Riedewald M., Gehrke J., Demers A. J.: What is “next” in event processing. PODS, 263–272. ACM, 2007.

Sequencing in Cayuga

The sequencing model of Cayuga is $(T_{cy}, \prec_{cy}, \mathsf{S}_{cy}, \otimes_{cy})$, where

- T_{cy}: intervals over the positive integers.
- $i_1 \prec_{cy} i_2$ iff i_1 ends before the start of i_2.

White W. M., Riedewald M., Gehrke J., Demers A. J.: What is “next” in event processing. PODS, 263–272. ACM, 2007.

Sequencing in Cayuga

The sequencing model of Cayuga is $(T_{cy}, \prec_{cy}, \mathsf{S}_{cy}, \otimes_{cy})$, where

- T_{cy}: intervals over the positive integers.
- $i_1 \prec_{cy} i_2$ iff i_1 ends before the start of i_2.
- $i_2 \in \mathsf{S}_{cy}(i_1, I_2)$ iff $i_2 \in I_2$, i_2 is after i_1 and no interval in I_2 is after i_1 and ends before i_2.

White W. M., Riedewald M., Gehrke J., Demers A. J.: What is "next" in event processing. PODS, 263–272. ACM, 2007.

Sequencing in Cayuga

The sequencing model of Cayuga is $(T_{cy}, \prec_{cy}, \mathsf{S}_{cy}, \otimes_{cy})$, where

- T_{cy}: intervals over the positive integers.
- $i_1 \prec_{cy} i_2$ iff i_1 ends before the start of i_2.
- $i_2 \in \mathsf{S}_{cy}(i_1, I_2)$ iff $i_2 \in I_2$, i_2 is after i_1 and no interval in I_2 is after i_1 and ends before i_2.
- If $i_1 \prec_{cy} i_2$, we have $i_1 \otimes_{cy} i_2 = (s(i_1), e(i_2))$.

White W. M., Riedewald M., Gehrke J., Demers A. J.: What is "next" in event processing. PODS, 263–272. ACM, 2007.

Sequencing in Cayuga

The sequencing model of Cayuga is $(T_{cy}, \prec_{cy}, \mathsf{S}_{cy}, \otimes_{cy})$, where

- T_{cy}: intervals over the positive integers.
- $i_1 \prec_{cy} i_2$ iff i_1 ends before the start of i_2.
- $i_2 \in \mathsf{S}_{cy}(i_1, I_2)$ iff $i_2 \in I_2$, i_2 is after i_1 and no interval in I_2 is after i_1 and ends before i_2.
- If $i_1 \prec_{cy} i_2$, we have $i_1 \otimes_{cy} i_2 = (s(i_1), e(i_2))$.

The sequencing operator of Cayuga for two activities α_1 and α_2 and a stream S is:

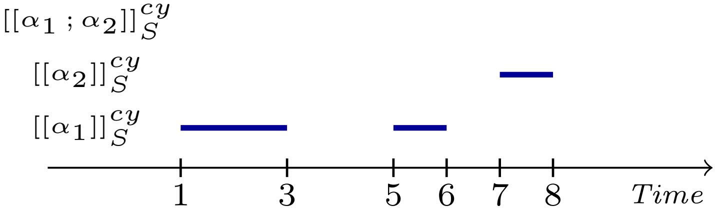
$$[[\alpha_1 ; \alpha_2]]_S^{cy} = \{ i_1 \otimes_{cy} i_2 \mid i_1 \in [[\alpha_1]]_S^{cy} \wedge i_2 \in \mathsf{S}_{cy}(i_1, [[\alpha_2]]_S^{cy}) \}$$

White W. M., Riedewald M., Gehrke J., Demers A. J.: What is "next" in event processing. PODS, 263–272. ACM, 2007.

Sequencing in Cayuga: Example

The sequencing operator of Cayuga for two activities α_1 and α_2 and a stream S is:

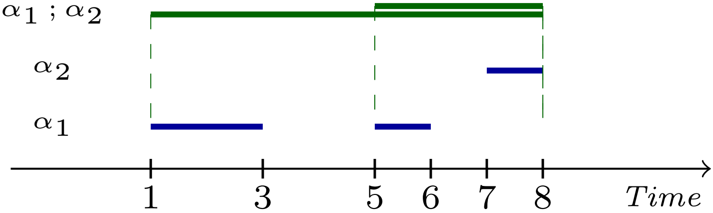
$$[[\alpha_1\,;\alpha_2]]_S^{cy} = \{ i_1 \otimes_{cy} i_2 \mid i_1 \in [[\alpha_1]]_S^{cy} \wedge i_2 \in \mathsf{S}_{cy}(i_1, [[\alpha_2]]_S^{cy}) \}$$



Sequencing in Cayuga: Example

The sequencing operator of Cayuga for two activities α_1 and α_2 and a stream S is:

$$[[\alpha_1\,;\alpha_2]]_S^{cy} = \{ i_1 \otimes_{cy} i_2 \mid i_1 \in [[\alpha_1]]_S^{cy} \wedge i_2 \in \mathsf{S}_{cy}(i_1, [[\alpha_2]]_S^{cy}) \}$$



Sequencing: Requirements for RTEC

Requirement (Maximal Disjoint Intervals)

Consider a sequencing operator ";", a stream S and activities α_1 and α_2. $[[\alpha_1\, ;\, \alpha_2]]_S$ is composed of maximal disjoint intervals (MDIs).

Sequencing: Requirements for RTEC

Requirement (Maximal Disjoint Intervals)

Consider a sequencing operator ";", a stream S and activities α_1 and α_2. $[[\alpha_1 ; \alpha_2]]_S$ is composed of maximal disjoint intervals (MDIs).

Requirement (Associativity)

Given a stream S and activities α_1, α_2 and α_3, $[[(\alpha_1;\alpha_2);\alpha_3]]_S$ is equal to $[[\alpha_1;(\alpha_2;\alpha_3)]]_S$.

Sequencing: Requirements for RTEC

Requirement (Maximal Disjoint Intervals)

Consider a sequencing operator ";", a stream S and activities α_1 and α_2. $[[\alpha_1\,;\alpha_2]]_S$ is composed of maximal disjoint intervals (MDIs).

Requirement (Associativity)

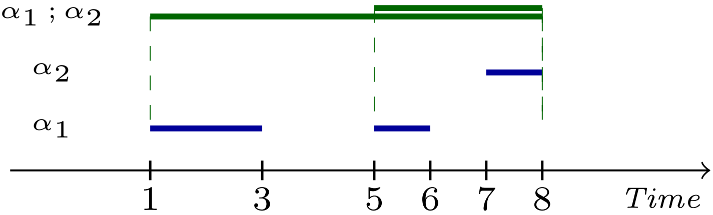
Given a stream S and activities α_1, α_2 and α_3, $[[(\alpha_1;\alpha_2);\alpha_3]]_S$ is equal to $[[\alpha_1;(\alpha_2;\alpha_3)]]_S$.

- These requirements imply compositionality.

MDI Requirement: Example

Requirement (Maximal Disjoint Intervals)

Consider a sequencing operator ";", a stream S and activities α_1 and α_2. $[[\alpha_1 ; \alpha_2]]_S$ is composed of maximal disjoint intervals (MDIs).



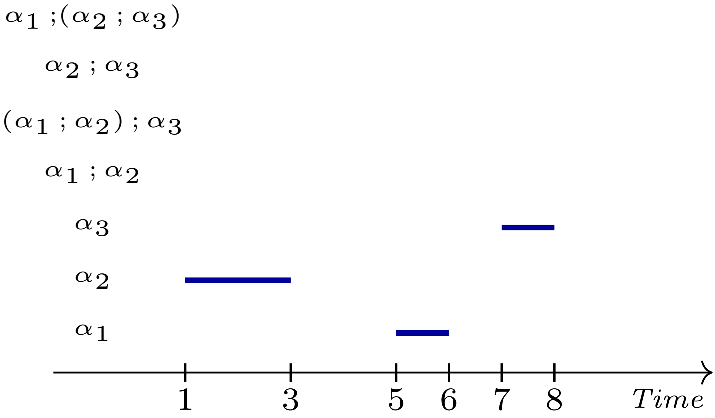
- Cayuga's sequencing operator violates the MDI requirement.

Associativity: Example

Requirement (Associativity)

Given a stream S and activities α_1, α_2 and α_3, $[[(\alpha_1;\alpha_2);\alpha_3]]_S$ is equal to $[[\alpha_1;(\alpha_2;\alpha_3)]]_S$.

- For a sequencing model where $i_1 \otimes i_2 = i_2$:

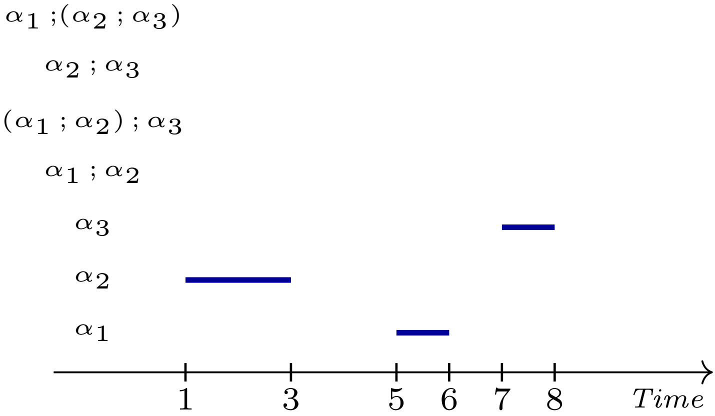


Associativity: Example

Requirement (Associativity)

Given a stream S and activities α_1, α_2 and α_3, $[[(\alpha_1;\alpha_2);\alpha_3]]_S$ is equal to $[[\alpha_1;(\alpha_2;\alpha_3)]]_S$.

- For a sequencing model where $i_1 \otimes i_2 = i_2$:

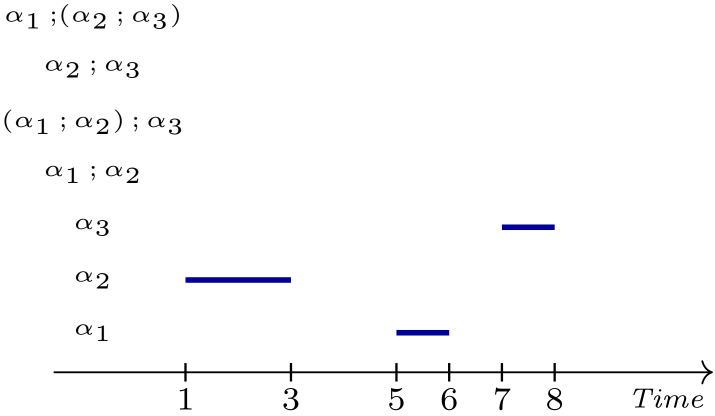


Associativity: Example

Requirement (Associativity)

Given a stream S and activities α_1, α_2 and α_3, $[[(\alpha_1;\alpha_2);\alpha_3]]_S$ is equal to $[[\alpha_1;(\alpha_2;\alpha_3)]]_S$.

- For a sequencing model where $i_1 \otimes i_2 = i_2$:

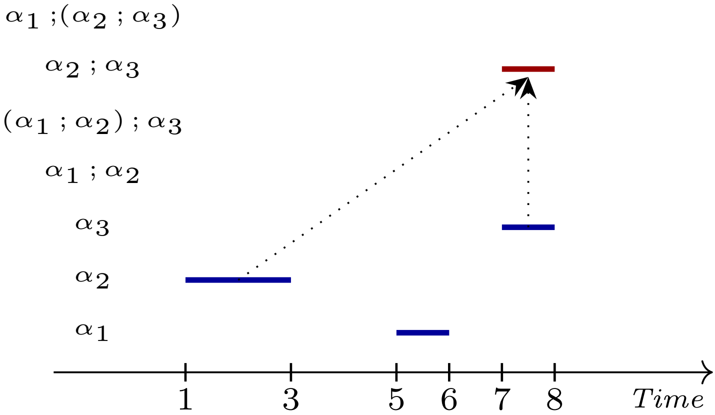


Associativity: Example

Requirement (Associativity)

Given a stream S and activities α_1, α_2 and α_3, $[[(\alpha_1;\alpha_2);\alpha_3]]_S$ is equal to $[[\alpha_1;(\alpha_2;\alpha_3)]]_S$.

- For a sequencing model where $i_1 \otimes i_2 = i_2$:

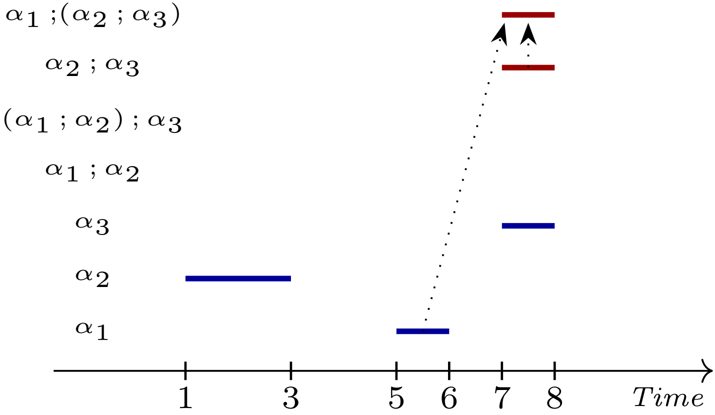


Associativity: Example

Requirement (Associativity)

Given a stream S and activities α_1, α_2 and α_3, $[[(\alpha_1;\alpha_2);\alpha_3]]_S$ is equal to $[[\alpha_1;(\alpha_2;\alpha_3)]]_S$.

- For a sequencing model where $i_1 \otimes i_2 = i_2$:

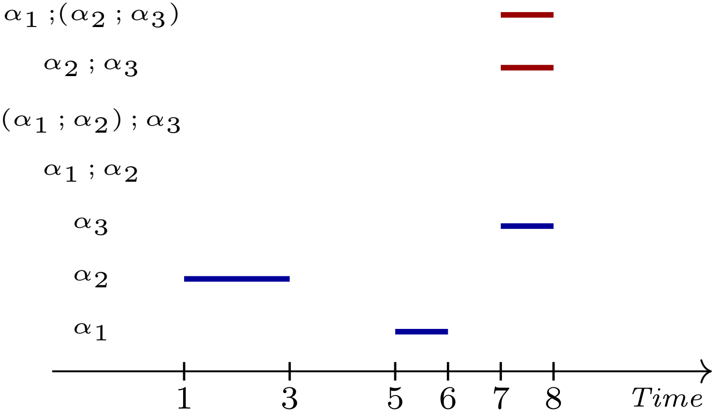


Associativity: Example

Requirement (Associativity)

Given a stream S and activities α_1, α_2 and α_3, $[[(\alpha_1;\alpha_2);\alpha_3]]_S$ is equal to $[[\alpha_1;(\alpha_2;\alpha_3)]]_S$.

- For a sequencing model where $i_1 \otimes i_2 = i_2$:



- Associativity fails: $[[(\alpha_1 ; \alpha_2) ; \alpha_3]]_S \neq [[\alpha_1 ;(\alpha_2 ; \alpha_3)]]_S$

Sequencing Operator for RTEC

The sequencing model of RTEC is $(T_{rt}, \prec_{rt}, \mathsf{A}, \otimes_{rt})$, where

Sequencing Operator for RTEC

The sequencing model of RTEC is $(T_{rt}, \prec_{rt}, \mathsf{A}, \otimes_{rt})$, where

- Time-stamps T_{rt}: intervals over the positive integers.

Sequencing Operator for RTEC

The sequencing model of RTEC is $(T_{rt}, \prec_{rt}, \mathrm{A}, \otimes_{rt})$, where

- Time-stamps T_{rt}: intervals over the positive integers.
- Partial Order: $i_1 \prec_{rt} i_2$ iff i_1 ends before the start of i_2.

Sequencing Operator for RTEC

The sequencing model of RTEC is $(T_{rt}, \prec_{rt}, \mathsf{A}, \otimes_{rt})$, where

- Time-stamps T_{rt}: intervals over the positive integers.
- Partial Order: $i_1 \prec_{rt} i_2$ iff i_1 ends before the start of i_2.
- Adjacency Mapping: $i_2 = \mathsf{A}(i_1, I_1, I_2)$ iff
  - $i_2 \in I_2$,
  - i_2 is after i_1, and
  - no interval in I_1 or I_2 is between i_1 and i_2.

Sequencing Operator for RTEC

The sequencing model of RTEC is $(T_{rt}, \prec_{rt}, \mathrm{A}, \otimes_{rt})$, where

- Time-stamps T_{rt}: intervals over the positive integers.
- Partial Order: $i_1 \prec_{rt} i_2$ iff i_1 ends before the start of i_2.
- Adjacency Mapping: $i_2 = \mathrm{A}(i_1, I_1, I_2)$ iff
  - $i_2 \in I_2$,
  - i_2 is after i_1, and
  - no interval in I_1 or I_2 is between i_1 and i_2.
- Composition Operator: $i_1 \otimes_{rt} i_2 = (s(i_1), e(i_2))$, if $i_1 \prec_{rt} i_2$.

Sequencing Operator for RTEC

The sequencing model of RTEC is $(T_{rt}, \prec_{rt}, \mathsf{A}, \otimes_{rt})$, where

- Time-stamps T_{rt}: intervals over the positive integers.
- Partial Order: $i_1 \prec_{rt} i_2$ iff i_1 ends before the start of i_2.
- Adjacency Mapping: $i_2 = \mathsf{A}(i_1, I_1, I_2)$ iff
  - $i_2 \in I_2$,
  - i_2 is after i_1, and
  - no interval in I_1 or I_2 is between i_1 and i_2.
- Composition Operator: $i_1 \otimes_{rt} i_2 = (s(i_1), e(i_2))$, if $i_1 \prec_{rt} i_2$.

The sequencing operator of RTEC computes:

$$[[\alpha_1 \,;\alpha_2]]^{rt}_S = \{i_1 \otimes_{rt} i_2 \mid i_1 \in [[\alpha_1]]^{rt}_S \wedge i_2 = \mathsf{A}(i_1, [[\alpha_1]]^{rt}_S, [[\alpha_2]]^{rt}_S) \wedge i_2 \neq \emptyset\}$$

This sequencing operator is compatible with RTEC because:

- $[[\alpha_1 \,;\alpha_2]]^{rt}_S$ consists of maximal disjoint intervals.
- It is associative, i.e., $[[(\alpha_1 \,;\alpha_2) \,;\alpha_3]]^{rt}_S = [[\alpha_1 \,;(\alpha_2 \,;\alpha_3)]]^{rt}_S$, allowing optimised pattern hierarchies.