Incremental Event Calculus for Run-Time Reasoning

Efthimis Tsilionis eftsilio@di.uoa.gr, eftsilio@iit.demokritos.gr
Department of Informatics & Telecommunications,
National and Kapodistrian University of Athens, Greece
Institute of Informatics & Telecommunications, NCSR "Demokritos", Greece

Alexander Artikis a.artikis@unipi.gr
Department of Maritime Studies, University of Piraeus, Greece
Institute of Informatics & Telecommunications, NCSR "Demokritos", Greece

Georgios Paliouras paliourg@iit.demokritos.gr
Institute of Informatics & Telecommunications, NCSR "Demokritos", Greece

Abstract

We present a system for online, incremental composite event recognition. In streaming environments, the usual case is for data to arrive with a (variable) delay from, and to be revised by, the underlying sources. We propose $RTEC_{inc}$, an incremental version of RTEC, a composite event recognition engine with formal, declarative semantics, that has been shown to scale to several real-world data streams. RTEC deals with delayed arrival and revision of events by computing all queries from scratch. This is often inefficient since it results in redundant computations. Instead, $RTEC_{inc}$ deals with delays and revisions in a more efficient way, by updating only the affected queries. We examine $RTEC_{inc}$ theoretically, presenting a complexity analysis, and show the conditions in which it outperforms RTEC. Moreover, we compare $RTEC_{inc}$ and RTEC experimentally using real-world and synthetic datasets. The results are compatible with our theoretical analysis and show that $RTEC_{inc}$ outperforms RTEC in many practical cases.

1. Introduction

Streaming environments combine simple, derived events (SDEs) in order to recognize in real-time composite events (CEs) that satisfy a given pattern. These patterns are collections of simpler events, which are subject to temporal and atemporal constraints and may be combined with static background knowledge (Cugola & Margara, 2012; Alevizos et al., 2017; Giatrakos et al., 2020).

The Event Calculus for Run-Time Reasoning (RTEC) (Artikis et al., 2015) is a composite event recognition (CER) system that has been tested and proven efficient in numerous applications, such as urban traffic management, public space surveillance and maritime situational awareness (Patroumpas et al., 2017). RTEC is a dialect of the Event Calculus, which is a logic programming formalism for representing and reasoning about events and their effects (Kowalski & Sergot, 1986). Moreover, RTEC includes various optimisation techniques for computing the intervals of CEs in a data stream.

In real-life streaming tasks, the usual case is for the input events to arrive with variable delays to the CER system. In the maritime domain, for example, delays occur in the input events when the stations (terrestrial and satellite) that collect the position signals of vessels have to deal with a great amount of messages. Furthermore, revisions or retractions may be applied to input events; e.g. the start or end time of an event may be wrong and subsequently corrected by the event source.

Delays and retractions in the input stream are being handled by RTEC by means of windowing. RTEC employs overlapping windows in order to 'wait' for delayed events and retractions. A drawback is that CE intervals within a window are computed from scratch, without considering the CE intervals of the previous overlapping windows. This way the intervals of a CE will be re-calculated even if delays and retractions do not have an effect on them. In previous work (Tsilionis et al., 2019a), we presented an incremental version of RTEC, i.e. $RTEC_{inc}$, that overcomes this inefficiency for a type of CEs. In the present paper, we extend the incremental algorithm in order to handle all types of CEs supported by RTEC. In particular, the contributions of this paper are the following:

- We extend $RTEC_{inc}$ by supporting all types of CEs of the language of RTEC.
- We provide a detailed theoretical evaluation of $RTEC_{inc}$ and show the conditions in which it achieves lower computational complexity compared to RTEC.
- We compare $RTEC_{inc}$ and RTEC experimentally using real-world and synthetic datasets from different application domains. The results are compatible with our complexity analysis and show that $RTEC_{inc}$ is preferable over RTEC in many practical cases. The code of $RTEC_{inc}$ is publicly available[1]. Moreover, some of the employed datasets are available[2], allowing the reproducibility of our results.

The structure of the paper is as follows: Section 2 summarises the functionality of RTEC. Sections 3 and 4 elaborate on the algorithmic details of the incremental procedure and the complexity analysis. Section 5 presents our empirical analysis, while Section 6 discusses related work. Finally, in Section 7 we summarise the work and outline future work directions.

2. Background: Run-Time Event Calculus

2.1 Language

The time model used by RTEC is linear and includes integer time-points (Artikis et al., 2015). If F is a fluent — a property that is allowed to have different values at different points in time — the term $F{=}V$ denotes that fluent F has value V. Boolean fluents are a special case in which the possible values are true and false. holdsAt($F{=}V$, T) is a predicate representing that fluent F has value V at time-point T. holdsFor($F{=}V$, I) represents that I is the list of maximal intervals for which $F{=}V$ holds continuously. holdsAt and holdsFor are defined in such a way that, for any fluent F, holdsAt($F{=}V$, T) if and only if T belongs to one of the maximal intervals of I for which holdsFor($F{=}V$, I).

An event description in RTEC comprises rules that express: (a) event occurrences using the happensAt predicate, (b) the effects of events using the initiatedAt and terminatedAt predicates, (c) the values of fluents, with the use of the holdsAt and holdsFor predicates, as well as other,

[1] `https://github.com/eftsilio/Incremental_RTEC`

[2] The datasets concerning maritime activity in the area of Brest, France, are public and can be downloaded through the link provided in the github repository of $RTEC_{inc}$.

Table 1: Main predicates of RTEC.

Predicate	Meaning
happensAt$(E,\ T)$	Event E occurs at time T
holdsAt$(F{=}V,\ T)$	The value of fluent F is V at time T
holdsFor$(F{=}V,\ I)$	I is the list of maximal intervals for which $F{=}V$ holds continuously
initiatedAt$(F{=}V,\ T)$	At time T the simple fluent $F{=}V$ is initiated
terminatedAt$(F{=}V,\ T)$	At time T the simple fluent $F{=}V$ is terminated
relative_complement_all$(I',\ L,\ I)$	I is the list of maximal intervals produced by the relative complement of the list of maximal intervals I' with respect to every list of maximal intervals of list L
union_all$(L,\ I)$	I is the list of maximal intervals produced by the union of the lists of maximal intervals of list L
intersect_all$(L,\ I)$	I is the list of maximal intervals produced by the intersection of the lists of maximal intervals of list L

possibly atemporal, parameters. Table 1 summarises the RTEC predicates available to the event description developer. Variables start with an upper-case letter, while predicates and constants start with a lower-case letter. Event Calculus events express instantaneous SDEs and CEs, while fluent-value pairs express durative SDEs and CEs. In CER, the majority of CEs are durative and, therefore, the task is to compute the maximal intervals for which a fluent-value pair $F{=}V$ representing a CE has a particular value continuously. A fluent F may have at most one value at each time-point. This constraint is satisfied by built-in axioms of RTEC (presented in the Appendix). Fluents in RTEC are of two kinds: simple and statically determined. The incremental computation of the maximal intervals of simple and statically determined fluent-value pairs is the subject of this paper.

2.1.1 Simple Fluents

Simple fluents are defined by means of initiatedAt and terminatedAt rules. Below we present an abstract initiatedAt rule. terminatedAt rules have similar form.

$$\begin{array}{l} \mathsf{initiatedAt}(F{=}V,\ T) \leftarrow \\ \quad \mathsf{happensAt}(A,\ T), \\ \quad \mathsf{holdsAt}(B{=}V_B,\ T), \\ \quad \mathsf{not}\ \ \mathsf{happensAt}(C,\ T), \\ \quad \mathsf{not}\ \ \mathsf{holdsAt}(D{=}V_D,\ T). \end{array} \tag{1}$$

Rule (1) is a rule of conjunctions, meaning that all body literals should be satisfied in order for the rule to fire. not denotes negation by failure (Clark, 1977). The variable T, present at the head and all body literals, expresses that all literals are evaluated at the same time-point. Rule (1) is satisfied at time-point T if event A has occurred at T, there exists an interval of fluent B that includes T, there is no occurrence of event C at T and there is no interval of fluent D that includes T. We use the term *positive* to refer to events and fluents that must occur at or include T, e.g. A and B, and the term *negative* to refer to events and fluents that should not occur at or include T (symbol not), e.g. C and D. initiatedAt and terminatedAt rules of type (1) are not restricted in the number of body literals. The only requirement is the first body literal to be a *positive* happensAt predicate, which can then be followed by a possibly empty set of *positive/negative* happensAt and holdsAt predicates. An example simple fluent definition, from the maritime domain (Pitsikalis et al., 2019), is presented below:

$$\begin{array}{l}
\mathsf{initiatedAt}(\mathit{gap}(\mathit{Vessel}){=}\mathit{nearPorts},\ T) \leftarrow \\
\quad \mathsf{happensAt}(\mathit{gap\_start}(\mathit{Vessel}),\ T), \\
\quad \mathsf{holdsAt}(\mathit{withinArea}(\mathit{Vessel}, \mathit{nearPorts}){=}\mathsf{true},\ T). \\
\mathsf{initiatedAt}(\mathit{gap}(\mathit{Vessel}){=}\mathit{farFromPorts},\ T) \leftarrow \\
\quad \mathsf{happensAt}(\mathit{gap\_start}(\mathit{Vessel}),\ T), \\
\quad \mathsf{not}\ \ \mathsf{holdsAt}(\mathit{withinArea}(\mathit{Vessel}, \mathit{nearPorts}){=}\mathsf{true},\ T). \\
\mathsf{terminatedAt}(\mathit{gap}(\mathit{Vessel}){=}\mathit{nearPorts},\ T) \leftarrow \\
\quad \mathsf{happensAt}(\mathit{gap\_end}(\mathit{Vessel}),\ T). \\
\mathsf{terminatedAt}(\mathit{gap}(\mathit{Vessel}){=}\mathit{farFromPorts},\ T) \leftarrow \\
\quad \mathsf{happensAt}(\mathit{gap\_end}(\mathit{Vessel}),\ T).
\end{array} \tag{2}$$

The above set of rules formalises the notion of a ‘communication gap’. Communication gaps occur when a vessel is not emitting its position, either due to the absence of a nearby receiving station or on purpose. In maritime situational awareness, communication gaps are important since the vessel’s behavior is unknown and may indicate an intention of hiding (e.g. in cases of illegal fishing in a protected area). Notice that there is a distinction between gaps occurring near ports and those occurring in the open sea. The former are usually not significant in maritime monitoring. For a succinct formulation, we represent a communication gap as a multi-valued fluent, as opposed to Boolean fluents.

gap_start and *gap_end* are input events produced by a module annotating vessel position streams in real-time, indicating that a vessel stopped and re-started transmitting its position, respectively (Patroumpas et al., 2017). *withinArea*(*Vessel*, *nearPorts*) is a durative input event, denoting the periods of time a vessel is near a port. It is produced by online spatial processing of the position signals of the vessels (Santipantakis et al., 2018). According to rule-set (2), a communication gap is initiated for a *Vessel* if a *gap_start* has occurred near or far from ports, and terminated when a *gap_end* event is detected.

RTEC utilises the time-points produced by initiatedAt and terminatedAt rules to construct the maximal intervals during which a simple fluent has a particular value continuously. Therefore, to compute the intervals I for which $F{=}V$, i.e. holdsFor($F{=}V$, I), we find all time-points T_s at which $F{=}V$ is initiated by using initiatedAt rules, and then, for each T_s, we compute the first time-point T_f after T_s at which $F{=}V$ is terminated, by evaluating terminatedAt rules. This is an implementation of the *law of inertia*.

Table 2: Examples of the three interval manipulation constructs of RTEC.

Interval manipulation construct	I_A	I_B	I_F
$\mathsf{union\_all}([I_A, I_B],\ I_F)$	$[(5,20),(26,30)]$	$[(28,35)]$	$[(5,20),(26,35)]$
$\mathsf{intersect\_all}([I_A, I_B],\ I_F)$	$[(26,31)]$	$[(21,26),(30,40)]$	$[(30,31)]$
$\mathsf{relative\_complement\_all}(I_A,\ [I_B],\ I_F)$	$[(5,20),(26,30)]$	$[(1,4),(18,22)]$	$[(5,18),(26,30)]$

2.1.2 Statically Determined Fluents

In addition to simple fluents, an event description may include domain-specific holdsFor rules that define the values of a fluent F in terms of the intervals of other fluents. These fluents are called 'statically determined' and the rules that define them make use of interval manipulation constructs — see the last three items of Table 1. Consider the following abstract rule:

$$\begin{aligned}&\mathsf{holdsFor}(F{=}V,\ I_F) \leftarrow\\ &\quad \mathsf{holdsFor}(A{=}V_A,\ I_A),\\ &\quad \mathsf{holdsFor}(B{=}V_B,\ I_B),\\ &\quad \mathit{interval\_manipulation}([I_A, I_B],\ I_F).\end{aligned} \tag{3}$$

The evaluation of rule (3) will produce the maximal intervals I_F of $F{=}V$, by combining the maximal intervals I_A and I_B of $A{=}V_A$ and $B{=}V_B$ according to a given interval manipulation operation. The three interval manipulation constructs of RTEC are union_all, intersect_all and relative_complement_all. Table 2 displays examples of these constructs. A term of the form $(T_s,\ T_e)$ represents the closed-open interval $[T_s,\ T_e)$. I_F in $\mathsf{union\_all}([I_A,\ I_B],\ I_F)$ is a list of maximal intervals that includes each time-point that is part of the lists of maximal intervals I_A or I_B. I_F in $\mathsf{intersect\_all}([I_A,\ I_B],\ I_F)$ is a list of maximal intervals that includes each time-point that is part of the lists I_A and I_B. Finally, I_F in $\mathsf{relative\_complement\_all}(I_A,\ [I_B],\ I_F)$ is a list of maximal intervals including each time-point of I_A that is not part of I_B.

Consider the following example definition of a statically determined fluent from the maritime domain:

$$\begin{aligned}&\mathsf{holdsFor}(\mathit{rendezVous}(\mathit{Vessel_1}, \mathit{Vessel_2}){=}\mathsf{true},\ I) \leftarrow\\ &\quad \mathsf{holdsFor}(\mathit{proximity}(\mathit{Vessel_1}, \mathit{Vessel_2}){=}\mathsf{true},\ I_p),\\ &\quad \mathsf{holdsFor}(\mathit{lowSpeed}(\mathit{Vessel_1}){=}\mathsf{true},\ I_{l1}),\\ &\quad \mathsf{holdsFor}(\mathit{lowSpeed}(\mathit{Vessel_2}){=}\mathsf{true},\ I_{l2}),\\ &\quad \mathsf{holdsFor}(\mathit{stopped}(\mathit{Vessel_1}){=}\mathit{farFromPorts},\ I_{s1}),\\ &\quad \mathsf{holdsFor}(\mathit{stopped}(\mathit{Vessel_2}){=}\mathit{farFromPorts},\ I_{s2}),\\ &\quad \mathsf{union\_all}([I_{l1}, I_{s1}],\ I_1),\\ &\quad \mathsf{union\_all}([I_{l2}, I_{s2}],\ I_2),\\ &\quad \mathsf{intersect\_all}([I_1, I_2, I_p],\ I).\end{aligned} \tag{4}$$

According to rule (4) two vessels are assumed to perform a *rendezVous* if they are close to each other and are stopped in the open sea or moving with low speed. Maritime analysts are interested in these cases as they may be indicators of illegal ship-to-ship transfers. The *proximity* fluent is a durative input SDE denoting the periods of time in which two vessels are close to each other, and is produced by online spatial processing. *lowSpeed* and *stopped* are simple fluents denoting when a vessel is moving with a low speed or stopped, respectively.

The interval manipulation constructs of RTEC support the following type of specification: for all time-points T, $F{=}V$ holds at T if and only if a Boolean combination of fluent-value pairs holds at T. For many fluents, this is a much more concise (and cheaper) specification than the traditional style of Event Calculus representation, i.e. identifying all conditions in which the fluent is initiated and terminated, so that maximal intervals may then be computed using the domain-independent holdsFor.

2.2 Semantics and Operation

The language of RTEC supports non-monotonic reasoning and favours Clark completion since it assumes a closed-world description, meaning that the information provided for predicates is sufficient and necessary. Rules in RTEC are 'safe', i.e. every variable that appears in the head of the rule or in any negative literal in the body also appears in at least one positive literal in the body. CE definitions are (locally) stratified logic programs (Przymusinski, 1987). Stratification allows the mapping of all fluent-value pairs $F{=}V$ and all events to the non-negative integers. At level 0 we find the events and statically determined fluents that do not depend on any other events or fluents, and serve as input to the CER system (these are the explicit facts of our knowledge base). Simple fluents are output entities and thus, belong to higher strata. Events and fluents of level n are defined in terms of at least one event or fluent-value of level $n-1$ and a possibly empty set of events and fluent-values from levels lower than $n-1$. In other words, we restrict our attention to hierarchical definitions; in Figure 1 we present an example CE definition hierarchy from the maritime domain. The restriction to hierarchical definitions allows RTEC to perform recognition in a bottom-up manner, according to which the fluents and events at the bottom of the hierarchy are processed first and their intervals are cached. Subsequently, RTEC processes the events and fluents of the next level and caches their intervals, and reaches level-by-level the top of the hierarchy. This way, when evaluating the rules of level n, the intervals of the fluents and events of the body literals are simply fetched from the cache, thus avoiding unnecessary re-computations.

A CER system consumes a stream of SDEs which may not necessarily be temporally ordered, meaning that there may exist a (variable) delay between the time each SDE occurs and the time of arrival at the CER system. Furthermore, there may be revisions and retractions of SDEs. Consider for example the case where the parameters of an SDE were originally computed erroneously and subsequently revised, or the case where an SDE was reported by mistake, and the mistake was realized later (Anicic et al., 2012).

In RTEC, the CER process involves the computation of the maximal intervals of fluent-value pairs expressing CEs. This process takes place at specified query times $q_1, q_2, \ldots$. The recognition at each q_i is performed over the SDEs that fall within a specified interval, the 'working memory' or window ω. All SDEs outside the window are discarded and not considered during recognition. This means that at each q_i the CER depends only on the

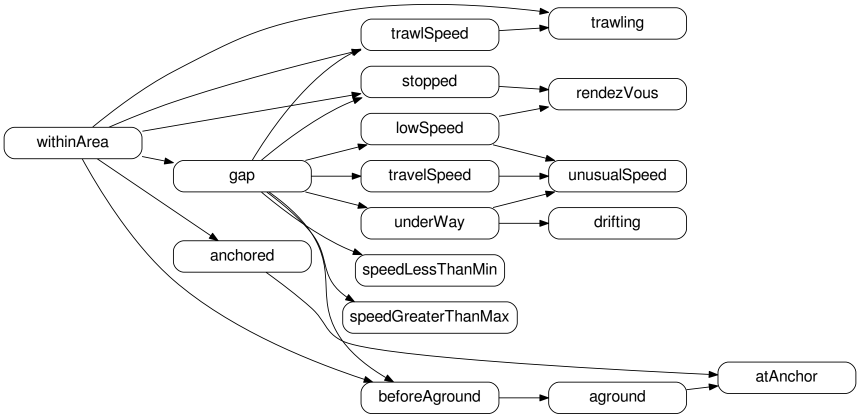


Figure 1: Hierarchy graph of maritime CE definitions (Pitsikalis et al., 2019). The nodes of the graph express fluents while an arrow from node X to node Y expresses that fluent X is a body literal in the definition of fluent Y.

SDEs that took place in the interval $(q_i - \omega, q_i]$. The size of ω as well as the temporal distance between two consecutive query times — the step $(q_i{-}q_{i-1})$ — are user-specified.

In order to deal with delays or retractions of SDEs, the user must set ω to be longer than the step, i.e. $q_i{-}\omega < q_{i-1} < q_i$. When ω is longer than the step, it is possible that an SDE occurs in the interval $(q_i{-}\omega,\ q_{i-1}]$ but arrives at RTEC in $(q_{i-1}, q_i]$; its effects are taken into account at query time q_i. Similarly, the effects of SDE retraction will be taken into consideration for SDEs that took place in $(q_i{-}\omega,\ q_{i-1}]$ and were revised in $(q_{i-1}, q_i]$. However, information may still be lost. Any SDEs arriving or retracted between q_{i-1} and q_i are discarded at q_i if they took place before or at $q_i{-}\omega$.

At each query time q_i, RTEC computes from scratch and stores the intervals of fluent-value pairs expressing CEs. Figure 2 illustrates the process of computing the initiation points of the fluent-value pair $F{=}V$ at two consecutive query times with the use of rule (1) (see page 3). In this example, the window ω is longer than the step; this way, we may consider delayed arrivals or retraction of events, as well as fluent intervals calculated or removed at q_i and falling in $(q_i{-}\omega,\ q_{i-1}]$. At the upper part of Figure 2 the upward arrows represent the initiation points calculated at the previous query time q_{i-1} with the use of rule (1). These points are the result of occurrences of event A and intervals of fluent B that include the occurrences of A. Additionally, event C did not occur at these time-points and the intervals of fluent D did not include these time-points.

At the bottom part of Figure 2, we present the initiation points produced by evaluating rule (1) at q_i. Notice the presence of delayed arrivals for events A and C (green dots), of a new interval computed for fluent D (green line), of a retraction of event C (red dot), and of an interval reduction for fluent B. The delayed arrival of event A along with the retraction of event C lead to a new initiation point. Two initiation points that were present at q_{i-1} are no longer present at q_i. The first, due to the new interval of fluent D or the diminished interval of fluent B (both have the same effect), and the second due to a delayed arrival of event C.

However, three out of the four initiation points calculated at q_i are identical among the two query times (see the vertical dashed lines). These initiation points are not affected by delays and retractions, but are re-computed. This is unnecessary and indicates the redundant computations performed by RTEC. Note that the same has to be done for termination points. In large event descriptions, expressing big CE hierarchies (recall e.g. Figure 1), this process can be very expensive. The example displayed in Figure 2 concerns simple fluents, but RTEC suffers from similar inefficiencies in the case of statically determined fluents. In the following sections we will present methods for the incremental evaluation of the maximal intervals of both types of fluents, addressing these inefficiencies.

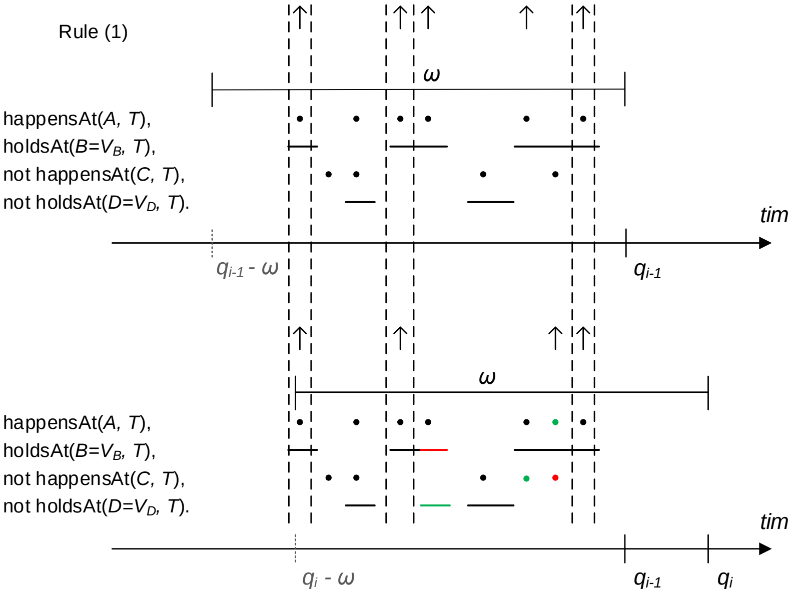


Figure 2: Example of initiation point computation. The upper part of the figure shows the initiation points of fluent-value pair $F{=}V$, as defined by rule (1), calculated at the previous query time q_{i-1}, while the bottom part shows the initiation points calculated at q_i. Dots represent event occurrences, unlabeled horizontal lines represent fluent intervals and arrows facing upwards represent initiation points. Green dots express event instances that arrived at RTEC at q_i. Green lines express fluent intervals that were calculated at q_i. Red dots (resp. lines) express event instances (fluent intervals) that were retracted at q_i. On the left of each part of the figure, we display the conditions of rule (1), indicating the type of event/fluent expressed by the corresponding row of dots/lines. For example, the first row of dots expresses occurrences of event A, the second row of lines expresses intervals of fluent B, and so on. Vertical dashed lines indicate the initiation points that are common between the two query times.

Table 3: Notation of incremental simple fluent computation.

Notation	Meaning
$se^{\mathsf{Q_i}}$	The set of starting/initiation and ending/termination points calculated at q_i and falling in $(q_i{-}\omega, q_{i-1}]$
$I^{\mathsf{Q_i}}$	The set of event occurrences and fluent intervals available at q_i and overlapping $(q_i{-}\omega, q_{i-1}]$
I^{+}	The set of event occurrences and fluent intervals inserted at q_i and overlapping $(q_i{-}\omega, q_{i-1}]$
I^{-}	The set of event occurrences and fluent intervals retracted at q_i and overlapping $(q_i{-}\omega, q_{i-1}]$

3. Incremental Evaluation of Simple Fluents

We present $RTEC_{inc}$, an extension of RTEC that includes a process for the incremental computation of the intervals of simple fluents. In what follows, we assume that windows are overlapping, i.e. that $q_i{-}\omega < q_{i-1} < q_i$. Moreover, incremental computation concerns the overlap between consecutive windows, i.e. $(q_i{-}\omega, q_{i-1}]$. Reasoning in $(q_{i-1}, q_i]$ may be performed using RTEC. Recall from Section 2.1.1 that the intervals of a simple fluent are produced on the basis of its initiation and termination points. Notice also that some of the initiation and termination points might not contribute to the intervals of the fluent at q_i but might contribute to future intervals. To make this more clear, assume that at the previous query time, q_{i-1}, an ending point, T_f, was calculated, but the absence of a starting point, T_s, prevented the construction of the interval $(T_s, T_f]$. Furthermore, consider that at q_i the delayed arrival of an event gives rise to the starting point $T_s < T_f$ and as result to the interval $(T_s, T_f]$. Thus, we must store all the initiation and termination points that fall inside the overlap of consecutive windows, i.e. $(q_i{-}\omega, q_{i-1}]$.

Table 3 summarizes the notation used throughout this section. Algorithm 1 shows the pseudo-code of recogniseSimpleFluent, the procedure for incrementally computing and storing the intervals of simple fluents in $RTEC_{inc}$. This procedure comprises several steps, some of which are in common with RTEC and are shown in bold. First, $RTEC_{inc}$ retrieves from the computer memory the initiation and termination points $se^{\mathsf{Q_{i-1}}}$ and the maximal intervals $I^{\mathsf{Q_{i-1}}}$ of $F{=}V$ computed at the previous query time q_{i-1} (see line 1).

Algorithm 1 recogniseSimpleFluent

```
1: $\boldsymbol{retrieve}(F{=}V, se^{\mathsf{Q_{i-1}}}, I^{\mathsf{Q_{i-1}}})$
2: $deleteSEpoints(F{=}V, se^{\mathsf{Q_{i-1}}})$
3: $computeSEpoints(F{=}V, se^{\mathsf{Q_{i-1}}}, se^{\mathsf{Q_i}})$
4: $\boldsymbol{makeIntervals}(se^{\mathsf{Q_i}}, I^{\mathsf{Q_i}})$
5: $symmetricDifference(I^{\mathsf{Q_i}}, I^{\mathsf{Q_{i-1}}}, I^{+}, I^{-})$
6: $\boldsymbol{assert}(F{=}V, se^{\mathsf{Q_i}}, I^{\mathsf{Q_i}}, I^{+}, I^{-})$
```

The second step (line 2 in Algorithm 1) deletes initiation and termination points, calculated at q_{i-1} and no longer holding at q_i. Next, the addition phase (line 3 in Algorithm 1) calculates new initiation and termination points, i.e. points that were not present at q_{i-1}. The details of the deletion and addition steps are presented in the following sub-sections. The time-points that survived the deletion phase and the new ones produced in the addition phase constitute the initiation/termination points of $F{=}V$ at q_i, i.e. $se^{\mathsf{Q_i}}$. The next step concerns the construction of the intervals of $F{=}V$ by means of the initiation and termination points. This process is done as in RTEC and leads to the intervals of $F{=}V$ at q_i, $I^{\mathsf{Q_i}}$ (line 4 in Algorithm 1).

At the end of this process, *RTEC*$_{inc}$ computes the intervals that were added (I^+), and retracted (I^-) for $F{=}V$ at q_i. This is necessary because $F{=}V$ itself may participate, positively or negatively, in the body of initiatedAt and terminatedAt rules of a fluent of a higher stratum. For example, consider again the hierarchy displayed in Figure 1 and notice that the intervals of the fluent *rendezVous* depend on the intervals of the fluents *stopped* and *lowSpeed*. In order to calculate incrementally the intervals of *rendezVous*, we need to know the intervals that were added (I^+) and retracted (I^-) for each one of the two constituent fluents. This is achieved by calculating the symmetric difference between the intervals of $F{=}V$ calculated at q_{i-1}, $I^{\mathsf{Q_{i-1}}}$, and the intervals calculated at q_i, $I^{\mathsf{Q_i}}$ (line 5 in Algorithm 1). Finally, the computed list of intervals $I^{\mathsf{Q_i}}$, along with the initiation/termination points calculated at q_i, $se^{\mathsf{Q_i}}$, and the intervals in I^+ and I^-, are stored (line 6 in Algorithm 1), replacing the ones computed at q_{i-1}.

In the following subsections, we delve into the details of the deletion and the addition phases, since these constitute the main differences between *RTEC*$_{inc}$ and RTEC, with respect to simple fluent recognition.

3.1 Deletion phase

In the deletion phase, *RTEC*$_{inc}$ examines which of the initiation and termination points falling in the overlapping part of two consecutive windows, $(q_i{-}\omega, q_{i-1}]$, still hold at q_i. An initiation/termination point may no longer hold for several reasons. In the case of negative happensAt literals (see e.g. literal C in rule (1) on page 3), the event may have occurred in the interval $(q_i{-}\omega, q_{i-1}]$ but arrived in $(q_{i-1}, q_i]$. If the time occurrence of this delayed event coincides with an initiation/termination point calculated at q_{i-1}, the latter must be retracted. In the case of negative fluents (see fluent D in rule (1)), an interval, not present at q_{i-1}, computed at q_i and falling in $(q_i{-}\omega, q_{i-1}]$ may include an initiation/termination point. Again, the initiation/termination point should be removed.

In the case of positive events (see event A in rule (1)), the time of occurrence of the event may have been reported by mistake at q_{i-1}, and by q_i the mistake may have been realized and the specific event occurrence retracted. If the time of this event occurrence coincides with the time of an initiation/termination point, then this point should be deleted. Finally, in the case of positive fluents (see fluent B in rule (1)), an interval, calculated at q_{i-1} and falling in $(q_i{-}\omega, q_{i-1}]$, that included an initiation/termination point may be deleted or reduced. As a consequence, the interval may no longer include the initiation/termination point and once again this point should be removed.

To determine which of the initiation/termination points of $F{=}V$ survive, we use a transformation of the rules expressing CE definitions. Rule (5) below e.g. is a transformation of rule (1):

$$\begin{aligned}
&\Big[\mathsf{initiatedAt}(F{=}V,\ T)\Big]^{se^{\mathsf{Q_{i-1}}}} &&\leftarrow\\
&\quad\Big[\mathsf{happensAt}(A,\ T)\Big]^{I^-} &&\vee\\
&\quad\Big[\mathsf{holdsAt}(B{=}V_B,\ T)\Big]^{I^-} &&\vee\\
&\quad\Big[\mathsf{happensAt}(C,\ T)\Big]^{I^+} &&\vee\\
&\quad\Big[\mathsf{holdsAt}(D{=}V_D,\ T)\Big]^{I^+}.
\end{aligned} \tag{5}$$

Rule (5) is a *delta* rule determining if an initiation point must be deleted; termination points are handled in a similar manner. As opposed to rule (1), this is a rule of disjunctions. Notice that the negative predicates of rule (1) occur positively in the body of rule (5). The superscripts in the head and the body literals indicate the set in which the time argument T is evaluated on. See Table 3 for the definitions of these sets. We evaluate rule (5) for each initiation point of $se^{\mathsf{Q_{i-1}}}$, i.e. for each initiation point calculated at q_{i-1} that falls in $(q_i{-}\omega, q_{i-1}]$. Since rule (5) consists of disjunctions, it suffices for the time argument of only one of the body literals to match the value of T in the head in order for the rule to be satisfied. Rule (5) fires, stating that an initiation point T in $se^{\mathsf{Q_{i-1}}}$ should be deleted, if and only if one of the following conditions is satisfied:

(a) T coincides with the time of a retracted occurrence of event A (set I^-).

(b) T belongs to a retracted interval of $B{=}V_B$ (set I^-).

(c) T matches the timestamp of a delayed arrival of event C (set I^+).

(d) T belongs to a new interval of $D{=}V_D$ calculated at q_i (set I^+), due to a delayed event arrival or retraction.

When all points in $se^{\mathsf{Q_{i-1}}}$ have been examined the deletion phase terminates. Figure 3 illustrates the deletion phase with the use of an example. The evaluation of rule (1) at q_{i-1} results in the initiation points shown in the upper part of Figure 3. The occurrences of event A belonging to intervals of $B{=}V_B$, not coinciding with occurrences of event C and not included in the intervals of $D{=}V_D$, leads to the calculation of these points at q_{i-1}. In the bottom part of Figure 3, rule (5) is used to examine which of these points should be removed. The vertical dashed lines indicate the points not surviving the deletion process. The deletion of each of these points is due to one of the four conditions outlined above. For example, the first initiation point is removed because the corresponding occurrence of A was retracted, while the last one was removed due to an interval of $D{=}V_D$, calculated at q_i, that includes this initiation point.

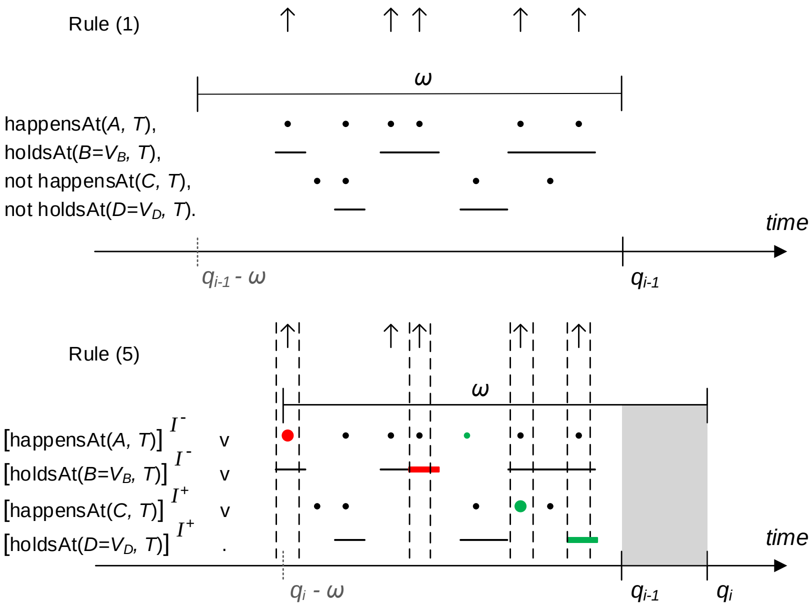


Figure 3: Illustration of the deletion phase. The upper part of the figure shows the initiation points calculated at q_{i-1} using rule (1). The bottom part displays the deletion process at q_i using the *delta* rule (5). The non-overlapping part of the two query times, $(q_{i-1}, q_i]$ is greyed out since the deletion phase concerns only the overlapping part, i.e. $(q_i{-}\omega, q_{i-1}]$. Dots represent event occurrences, unlabeled horizontal lines represent fluent intervals and arrows facing upwards represent initiation points. The black color signifies event occurrences and fluent intervals present both at q_{i-1} and q_i, the green color signifies delayed arrival of events and fluent interval computation at q_i, while the red color signifies event occurrences and fluent intervals retracted at q_i. On the left of each part of the figure, we display the rule conditions indicating the type of event/fluent expressed by the corresponding row of dots/lines. Enlarged dots and lines denote participation in the deletion process. The vertical dashed lines indicate the initiation points that will be removed.

3.2 Addition phase

Once the deletion phase has completed, the addition phase commences. The addition phase consists of the calculation of new initiation and termination points, i.e. points that were not present at q_{i-1}. The new time-points may belong to the overlapping part of the two consecutive windows, $(q_i{-}\omega, q_{i-1}]$, or to the non-overlapping part, $(q_{i-1}, q_i]$. We focus on the overlapping part, since it differentiates the method of computation of $RTEC_{inc}$ from that of RTEC. Computation in the non-overlapping part, $(q_{i-1}, q_i]$, is identical in the two algorithms, since initiation/termination points belonging to this part are not affected by delays or retractions.

Consider rule (1) again and assume that at q_{i-1} the rule did not fire, but all body predicates except the first one were satisfied at time-point T. At q_i a delayed arrival of event A at time-point T will activate the rule. Similarly, deletions of event occurrences or fluent intervals may lead to the satisfaction of a rule. For example, if rule (1) did not fire at

$$
\begin{array}{lclc}
\mathsf{initiatedAt}(F{=}V,\ T) \leftarrow & & \mathsf{initiatedAt}(F{=}V,\ T) \leftarrow & \\
\quad \Big[\mathsf{happensAt}(A,\ T)\Big]^{I^{+}}, & & \quad \Big[\mathsf{happensAt}(A,\ T)\Big]^{I^{\mathsf{Q_i}} \setminus I^{+}}, & \\
\quad \Big[\mathsf{holdsAt}(B{=}V_B,\ T)\Big]^{I^{\mathsf{Q_i}}}, & \text{(a)} & \quad \Big[\mathsf{holdsAt}(B{=}V_B,\ T)\Big]^{I^{+}}, & \text{(b)} \\
\quad \mathsf{not}\ \Big[\mathsf{happensAt}(C,\ T)\Big]^{I^{\mathsf{Q_i}}}, & & \quad \mathsf{not}\ \Big[\mathsf{happensAt}(C,\ T)\Big]^{I^{\mathsf{Q_i}}}, & \\
\quad \mathsf{not}\ \Big[\mathsf{holdsAt}(D{=}V_D,\ T)\Big]^{I^{\mathsf{Q_i}}}. & & \quad \mathsf{not}\ \Big[\mathsf{holdsAt}(D{=}V_D,\ T)\Big]^{I^{\mathsf{Q_i}}}. & \\
 & & & \\
\mathsf{initiatedAt}(F{=}V,\ T) \leftarrow & & \mathsf{initiatedAt}(F{=}V,\ T) \leftarrow & \\
\quad \Big[\mathsf{happensAt}(C,\ T)\Big]^{I^{-}}, & & \quad \Big[\mathsf{happensAt}(A,\ T)\Big]^{I^{\mathsf{Q_i}} \setminus I^{+}}, & \\
\quad \Big[\mathsf{happensAt}(A,\ T)\Big]^{I^{\mathsf{Q_i}} \setminus I^{+}}, & \text{(c)} & \quad \Big[\mathsf{holdsAt}(D{=}V_D,\ T)\Big]^{I^{-}}, & \text{(d)} \\
\quad \Big[\mathsf{holdsAt}(B{=}V_B,\ T)\Big]^{I^{\mathsf{Q_i}} \setminus I^{+}}, & & \quad \Big[\mathsf{holdsAt}(B{=}V_B,\ T)\Big]^{I^{\mathsf{Q_i}} \setminus I^{+}}, & \\
\quad \mathsf{not}\ \Big[\mathsf{holdsAt}(D{=}V_D,\ T)\Big]^{I^{\mathsf{Q_i}}}. & & \quad \mathsf{not}\ \Big[\mathsf{happensAt}(C,\ T)\Big]^{I^{\mathsf{Q_i}} \cup I^{-}}. &
\end{array}
\tag{6}
$$

q_{i-1} due to the fact that event C occurred at T, but at q_i the specific occurrence of event C was retracted, the rule would fire.

To calculate the new initiation points, we use the *delta* rules presented in (6) (termination points are handled similarly). The rules in (6) are examined in the given order. The superscripts of these rules correspond to the set in which the time argument T is evaluated and are presented in Table 3. In rule (6)(a), event A is evaluated over the occurrences that arrived at $RTEC_{inc}$ at q_i (set I^{+}). The time-points in set I^{+} are examined against all the intervals of $B{=}V_B$ (set $I^{\mathsf{Q_i}}$) overlapping the interval $(q_i{-}\omega, q_{i-1}]$. If an interval of $B{=}V_B$ includes a time-point in set I^{+}, then this time-point should not coincide with any occurrence of event C, and should not overlap any of the intervals of $D{=}V_D$. If all of these conditions are satisfied, then the rule fires and gives rise to a new initiation point. Figure 4(a) shows the calculation of an initiation point belonging to part $(q_i{-}\omega, q_{i-1}]$ by means of rule (6)(a).

Rule (6)(b) is similar to (6)(a), but has a small modification which ensures that derivations are not repeated. In this rule, only the intervals computed at q_i are considered for $B{=}V_B$ (set I^{+}). However, event A is matched against the occurrences at q_i, excluding the occurrences that were inserted to the system at q_i (set $I^{\mathsf{Q_i}} \setminus I^{+}$). This means that we examine only time-points falling in $(q_i{-}\omega, q_{i-1}]$ and present to the system from the previous query time q_{i-1}, excluding the ones, if any, that were retracted at q_i. This is important in order to avoid repeating evaluations. If in rule (6)(b) we used all the time-points of event A at q_i, then we would have to repeat evaluations, since the inserted time-points of event A at q_i (set I^{+}) are included in the occurrences of the event at q_i. The negative body literals are evaluated as in rule (6)(a). Figure 4(b) shows the calculation of an initiation point belonging to $(q_i{-}\omega, q_{i-1}]$ by using rule (6)(b).

Rule (6)(c) examines if a retracted occurrence of event C (set I^{-}) can lead to a new initiation point. Recall from rule (1) that we demand the absence of event C in order for the rule to be satisfied. Thus, if at q_i there are deleted occurrences of event C, this means

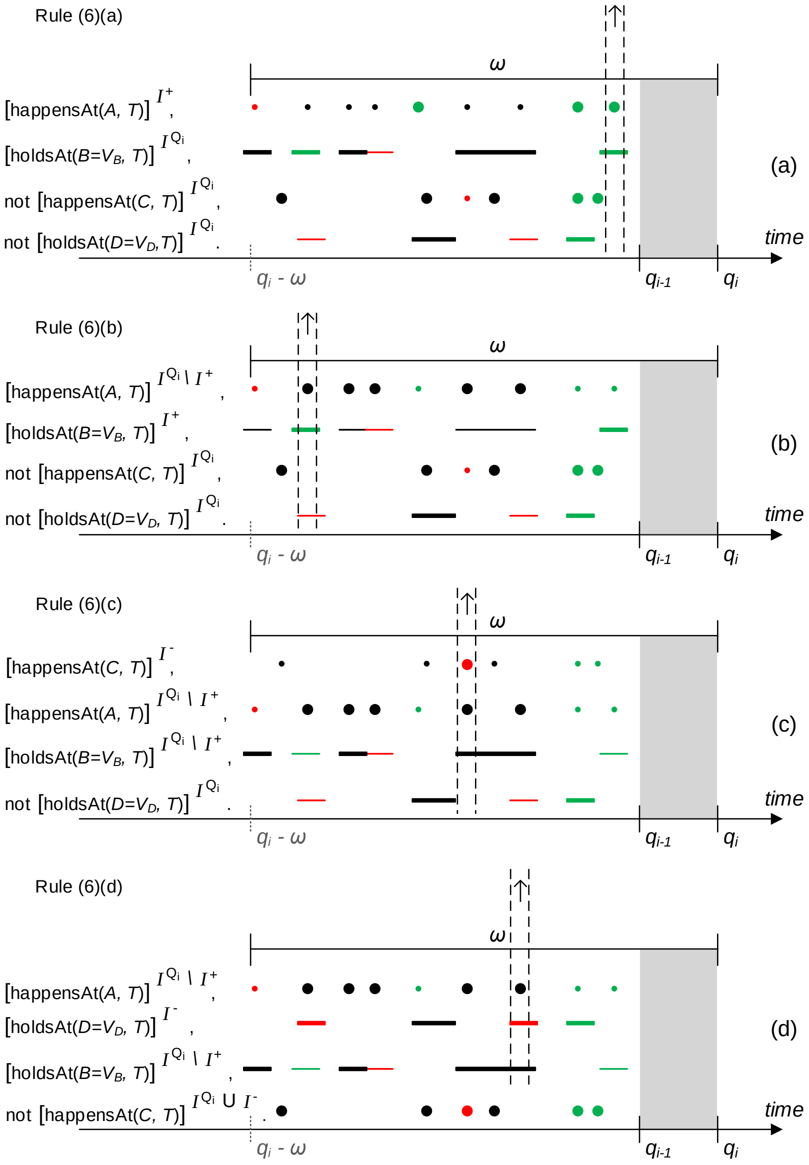

Rule (6)(a)
ω
[happensAt(A, T)] I+,
[holdsAt(B=VB, T)] IQi,
not [happensAt(C, T)] IQi,
not [holdsAt(D=VD,T)] IQi.
(a)
time
qi - ω
qi-1
qi
Rule (6)(b)
ω
[happensAt(A, T)] IQi \ I+,
[holdsAt(B=VB, T)] I+,
not [happensAt(C, T)] IQi,
not [holdsAt(D=VD, T)] IQi.
(b)
time
qi - ω
qi-1
qi
Rule (6)(c)
ω
[happensAt(C, T)] I-,
[happensAt(A, T)] IQi \ I+,
[holdsAt(B=VB, T)] IQi \ I+,
not [holdsAt(D=VD, T)] IQi.
(c)
time
qi - ω
qi-1
qi
Rule (6)(d)
ω
[happensAt(A, T)] IQi \ I+,
[holdsAt(D=VD, T)] I-,
[holdsAt(B=VB, T)] IQi \ I+,
not [happensAt(C, T)] IQi U I-.
(d)
time
qi - ω
qi-1
qi

Figure 4: Illustration of the addition phase. Dots represent event occurrences, unlabeled horizontal lines represent fluents intervals and arrows facing upwards represent initiation points. The black color signifies event occurrences and fluent intervals present both at q_{i-1} and q_i, the green color signifies events arriving at, and fluent intervals computed at q_i, while the red color signifies event occurrences and fluent intervals retracted at q_i. Enlarged dots and lines denote participation in the addition phase. The vertical dashed lines indicate the time-points and intervals that give rise to a new initiation point. Each of the four illustrations corresponds to a *delta* rule of rule-set (6). The non-overlapping part of the two query times, $(q_{i-1}, q_i]$, is greyed out in all illustrations since we focus on the overlapping part, i.e. $(q_i{-}\omega, q_{i-1}]$.

that event C did not actually occur at the previously reported time-points, and thus an initiation point of $F{=}V$ should have been calculated. Notice that the conditions of the rule have been re-ordered for performance reasons. Now, event C is evaluated first. This favors computation since the time-points in set I^- are usually few. Figure 4(c) shows the calculation of an initiation point belonging to $(q_i{-}\omega, q_{i-1}]$ by using rule (6)(c).

Rule (6)(d) examines the deleted intervals of $D{=}V_D$. At q_{i-1} an interval of $D{=}V_D$ may have prevented the calculation of an initiation point. If at q_i this interval of $D{=}V_D$ is deleted, the rule will produce a new initiation point. We employ the same optimisations as in the previous two rules, but we also introduce a new one concerning negative literals. In rule (6)(c) we examined event C over the time-points in I^-. If any of these points led to new initiation points, then these derivations should not be repeated in rule (6)(d). In order to achieve this, we evaluate event C negatively over all its occurrences at q_i including the deleted ones (set $I^{\mathsf{Q_i}} \cup I^-$). In other words, we take into account the occurrences in I^-, which are not present at q_i in order to avoid repeating derivations. Figure 4(d) shows the calculation of an initiation point belonging to $(q_i{-}\omega, q_{i-1}]$ by means of rule (6)(d).

In each of the four *delta* rules, a body literal is evaluated over the set I^+ or I^-. In practice these sets are small, compared to the set of all event occurrences and fluent intervals, i.e. the set $I^{\mathsf{Q_i}}$. By using these small sets, the evaluation is faster and the performance is improved compared to the recomputation from scratch of RTEC. The introduced optimisations also ensure that a new initiation point can be produced by only one of the four *delta* rules.

3.3 Complexity analysis

We present a worst-case complexity analysis of $RTEC_{inc}$ concerning simple fluents. We focus on the cost of computing maximal simple fluent intervals in the overlap of two consecutive query times, i.e. $(q_i{-}\omega, q_{i-1}]$. We omit the non-overlapping part, $(q_{i-1}, q_i]$, since the cost is the same in RTEC and $RTEC_{inc}$. Moreover, we make assumptions that lead to the worst overall cost of the incremental procedure as opposed to the worst case of one of its phases. We make this choice because the assumptions leading to the worst case of one phase, e.g. the addition phase, do not necessarily lead to the worst case of the other (e.g. deletion) phase.

Table 4 summarizes the notation of the complexity analysis. We assume that there are e events and fluents in the body of an initiatedAt/terminatedAt rule defining a simple fluent. Incremental reasoning makes sense only in the presence of delays and retractions falling in the overlap of two consecutive windows. Therefore, we assume that each of the e body literals of a rule will have n delayed insertions and retractions, i.e. $n > 0$.

3.3.1 Analysis of $RTEC_{inc}$

Deletion phase. In the deletion phase of the incremental procedure, the initiation and termination points $se^{\mathsf{Q_{i-1}}}$ calculated at q_{i-1} and falling in $(q_i{-}\omega, q_{i-1}]$ are examined in order to determine if they hold at q_i. This is achieved by the use of *delta* rules of type (5) presented on page 11. Assuming that there are l rules of type (1) (page 3) defining a fluent, then we will have l *delta* rules of type (5). In these *delta* rules each of the initiation/termination points calculated at q_{i-1} is checked against the delayed insertions of positive body events and fluents, as well as against the retractions of negative events and fluents. Evaluating

Table 4: Complexity analysis notation.

Notation	Meaning
m_{OV}	Number of time-points in $(q_i{-}\omega, q_{i-1}]$
e	Number of event and fluent types
n	Number of event occurrences or fluent intervals inserted and retracted at $(q_{i-1}, q_i]$ and overlapping $(q_i{-}\omega, q_{i-1}]$
l	Number of simple fluent rules

a happensAt predicate expressing an event in the body of a *delta* rule of type (5) requires retrieving the event's inserted or retracted time-points from the computer memory, and checking whether the initiation/termination point under investigation coincides with one of these inserted/retracted time-points. Similarly, evaluating a holdsAt predicate requires retrieving from the computer memory the inserted or retracted intervals of the fluent, and checking whether the initiation/termination point belongs to these intervals. In the worst case, all body literals of the *delta* rules of type (5) will be evaluated, and thus, the cost of the deletion phase is bound by:

$$\mathcal{O}\Bigg(l \times |se^{\mathsf{Q_{i-1}}}| \times e \times n\Bigg) \quad , \tag{7}$$

where l is the number of the delta rules, $|se^{\mathsf{Q_{i-1}}}|$ represents the number of initiation/termination points calculated at q_{i-1} and falling in $(q_i{-}\omega, q_{i-1}]$, e is the number of events and fluents in the body of a *delta* rule, and n the number of insertions/deletions of an event or fluent within $(q_i{-}\omega, q_{i-1}]$, as recorded in $(q_{i-1}, q_i]$ (see Table 4). Notice that in practice l and e are small, while $|se^{\mathsf{Q_{i-1}}}|$ and n are only a small subset of a sliding window.

Addition phase. The purpose of the addition phase is to compute the initiation/termination points that are the result of delayed insertions and retractions, by using *delta* rules of type (6). Each initiatedAt/terminatedAt rule contains e events and fluents. Therefore, each of the l rules of type (1) is transformed to e *delta* rules of type (6), leading to $l \times e$ *delta* rules in total. In each of these rules, only one event or fluent is evaluated over its delayed insertions or retractions. Next, the insertions/retractions of this event or fluent are checked against all the occurrences and intervals of the remaining body events and fluents in order to produce a new initiation/termination point. In the worst case, each insertion/retraction of an event and each inserted/retracted interval of a fluent will contribute a new initiation/termination point, and thus, the cost of computing the initiation/termination points by using all $l \times e$ *delta* rules is bound by:

$$\mathcal{O}\Bigg(l \times e^2 \times n \times |I^{\mathsf{Q_i}}|\Bigg) \quad , \tag{8}$$

where $|I^{\mathsf{Q_i}}|$ is the number of event occurrences or fluent intervals in $(q_i{-}\omega, q_{i-1}]$.

The addition phase is more expensive than the deletion phase. During the addition phase, we have to evaluate more *delta* rules than the deletion phase. Furthermore, in the evaluation of a *delta* rule in the addition phase we examine the delayed insertions or retractions of the first body literal against all the occurrences or intervals of the remaining body literals; in contrast, in the deletion phase the body literals are evaluated only over their delayed insertions and retractions. The number of delayed insertions and retractions is usually less than the number of all event occurrences and fluent intervals.

Calculation of inserted and retracted fluent intervals. The final steps of the incremental procedure consist of constructing intervals from starting and ending points, and calculating the inserted and retracted intervals of fluents (see Algorithm 1 on page 9). The cost of the former step is the same as that of RTEC. For the latter step, the cost of the symmetric difference (see line 5 of Algorithm 1) of the intervals computed at q_{i-1}, $I^{\mathsf{Q_{i-1}}}$, and the intervals computed at q_i, $I^{\mathsf{Q_i}}$, is limited by the sum of the sizes of the two lists, given that each list of maximal intervals is sorted. Since the intervals calculated at q_{i-1} are at most $|se^{\mathsf{Q_{i-1}}}|/2$ and the intervals calculated at q_i are at most $|se^{\mathsf{Q_i}}|/2$, the cost of the symmetric difference is bound by $\mathcal{O}(\frac{|se^{\mathsf{Q_i}}|+|se^{\mathsf{Q_{i-1}}}|}{2})$.

3.3.2 Comparison of RTEC and $RTEC_{inc}$

We examine the conditions in which $RTEC_{inc}$ is preferable over RTEC. The worst case scenario for RTEC does not coincide with the worst case scenario for $RTEC_{inc}$ and vice versa. Hence, we perform a comparison for the worst case of each CER engine.

Worst case for RTEC. In evaluation from scratch, the worst case is that every time-point of the overlap, $(q_i{-}\omega, q_{i-1}]$, is promoted to an initiation/termination point by all initiatedAt/terminatedAt rules. Therefore, the cost of RTEC is bound by:

$$\mathcal{O}\bigg(\, l \times m_{OV}^2 \times e \bigg) \quad , \tag{9}$$

where m_{OV} represents the number of time-points in the overlap $(q_i{-}\omega, q_{i-1}]$.

In the setting of the worst case for RTEC, the first body literal of an initiatedAt/terminatedAt rule has m_{OV} occurrences. Furthermore, these occurrences must coincide with the m_{OV} occurrences of each of the remaining positive body events and belong to the $m_{OV}/2$ intervals of the remaining positive body fluents. Additionally, there are no occurrences/intervals of negative events and fluents or all their occurrences/intervals are retracted at q_i. Based on the above, and performing the appropriate substitutions and simplifications in the total cost of $RTEC_{inc}$ and in formula (9) of RTEC, we derive inequality (10), that expresses the conditions in which $RTEC_{inc}$ is preferable to RTEC:

$$\frac{n \times e}{m_{OV}} < 1 \tag{10}$$

In other words, the most important factor for improving performance effectively is the ratio of delayed insertions/retractions to the degree of overlap.

The worst case of RTEC is extreme. However, the analysis of this case allows us to unravel the factors that most influence the performance of incremental evaluation. Intuitively, when there are few delays, $RTEC_{inc}$ is the best option.

Worst case for $RTEC_{inc}$. In incremental evaluation, the worst case is when every time-point in the overlap leads to an initiation/termination point at q_{i-1}, and at q_i all these points are retracted. In this case, RTEC is the preferred choice, since the cost of RTEC is close to zero, as there are no time-points that can be promoted to initiation/termination points. Deleting all initiation/termination points calculated at q_{i-1} is an inevitable operation for $RTEC_{inc}$, baring the cost of Eq. (7). In such extreme cases, where n is equal to (or approximates) m_{OV}, RTEC is the preferred option.

A typical case. Based on the experience of the authors in a range of CER applications, including the two domains used in this paper (see Section 5), the following conditions hold:

- The number of initiatedAt/terminatedAt rules (l) is in the same order of magnitude as the number of fluent and event types (e).
- $|se^{\mathsf{Q_{i-1}}}| \approx |se^{\mathsf{Q_i}}| \ll m_{OV}$. To aid the presentation, we will represent the average number of starting and ending points of all windows as $|se|$.
- The number of delays/retractions n is also a small fraction of the the window overlap, i.e. $n = \frac{m_{OV}}{c}$, where c is a constant.

By performing the appropriate substitutions in the total cost of $RTEC_{inc}$ and in formula (9) of RTEC, under these assumptions and minor simplifications, we conclude with the following inequality:

$$n < \frac{|se|}{e} \implies c \times \frac{|se|}{e} > m_{OV} \quad . \tag{11}$$

We observe again that the number of delays and retractions is the most crucial factor in order for $RTEC_{inc}$ to improve performance, where few delays favor $RTEC_{inc}$.

4. Incremental Evaluation of Statically Determined Fluents

In this section, we present the incremental evaluation of statically determined fluents. Recall e.g. rule (3) in Section 2.1.2 (page 5) that expresses a statically determined fluent definition. To simplify the presentation, when we refer to the intervals of a fluent we omit the value of the fluent, e.g. the intervals of $F{=}V$ are represented as I_F. The evaluation of rule (3) will produce the intervals of $F{=}V$, I_F, by combining the intervals I_A and I_B according to the given interval manipulation construct (see Table 1 on page 3). We will use rule (3) to illustrate the presentation that follows, as well as the example intervals of $A{=}V_A$ and $B{=}V_B$ that are displayed in Table 6.

We will describe the incremental process for each of the three manipulation constructs, focusing on the overlapping part of the two query times, $(q_i - \omega, q_{i-1}]$, and assuming without loss of generality that the definition of a statically determined fluent depends only on two fluents (as in rule (3)). This is because a holdsFor rule with k body fluents may be rewritten as a set of holdsFor rules with two fluents, each connected by means of auxiliary fluents.

Table 5 presents the notation concerning the analysis of statically determined fluents. Moreover, in the sections that follow, we will use the equation below, stating that the intervals $I_F^{\mathsf{Q_i}}$ of $F{=}V$ at q_i, overlapping $(q_i - \omega, q_{i-1}]$, are equal to the intervals produced by first calculating the relative complement of $I_F^{\mathsf{Q_{i-1}}}$ with respect to I_F^-, i.e. by calculating the intervals that are not retracted at q_i, and then computing the union of these intervals with the intervals I_F^+ that are inserted at q_i.

Table 5: Notation for interval manipulation and comparison.

Notation	Description
$I_A \cup_T I_B$	union_all operation of RTEC
$I_A \cap_T I_B$	intersect_all operation of RTEC
$I_A \setminus_T I_B$	relative_complement_all operation of RTEC
$I_A \subseteq_T I_B$	Temporal subset; it denotes that the time-points of the intervals of I_A are included in the intervals of I_B.

Table 6: Running example for the evaluation of rule (3)

	$A{=}V_A$	$B{=}V_B$
$I^{\mathsf{Q_{i-1}}}$	$[(10,15),(23,30),(40,50),(60,70)]$	$[(17,21),(26,35),(43,47),(54,65)]$
I^{-}	$[(10,12),(27,30),(43,47)]$	$[(19,21),(43,47),(60,65)]$
I^{+}	$[(54,58),(80,90),(95,100)]$	$[(37,41),(82,87),(105,120)]$
$I^{\mathsf{Q_{i-1}}}_{A \cup_T B}$	$[(10,15),(17,21),(23,35),(40,50),(54,70)]$	
$I^{\mathsf{Q_{i-1}}}_{A \cap_T B}$	$[(26,30),(43,47),(60,65)]$	
$I^{\mathsf{Q_{i-1}}}_{A \setminus_T B}$	$[(10,15),(23,26),(40,43),(47,50),(65,70)]$	
$I^{\mathsf{Q_{i-1}}}_{B \setminus_T A}$	$[(17,21),(30,35),(54,60)]$	

$$I_F^{\mathsf{Q_i}} = (I_F^{\mathsf{Q_{i-1}}} \setminus_T I_F^{-}) \cup_T I_F^{+} \tag{12}$$

4.1 Interval Union

First, we deal with the case of union_all and assume that at the previous query time, q_{i-1}, the list of intervals of the body fluents of rule (3) is not empty, that is, $I_A^{\mathsf{Q_{i-1}}}, I_B^{\mathsf{Q_{i-1}}} \neq \varnothing$. If this is not the case, then incremental evaluation will not improve performance. Furthermore, we choose to analyze the case where each body fluent has retractions and insertions, that is, $I_A^{-}, I_B^{-}, I_A^{+}, I_B^{+} \neq \varnothing$, as this is the most expensive case in terms of processing time.

4.1.1 RTEC

If we compute everything from scratch, as RTEC does, the evaluation of rule (3) leads to the following computation:

$$I_F^{\mathsf{Q_i}} = I_A^{\mathsf{Q_i}} \cup_T I_B^{\mathsf{Q_i}} \tag{13}$$

The size of $I_A^{\mathsf{Q_i}}$ and $I_B^{\mathsf{Q_i}}$ is, in the worst case, $|I_A^{\mathsf{Q_i}}| = |I_A^{\mathsf{Q_{i-1}}}| + |I_A^{-}| + |I_A^{+}|$ and $|I_B^{\mathsf{Q_i}}| = |I_B^{\mathsf{Q_{i-1}}}| + |I_B^{-}| + |I_B^{+}|$ respectively. To clarify, the worst case is for a deletion to break an existing interval to two parts and for an insertion to introduce a new interval, rather than extending an existing one. Additionally, in the worst case the lists of maximal intervals of fluents A and B are non-overlapping, and since the two lists are temporally sorted, the cost of Eq. (13) is limited by the sum of the sizes of the two lists:

$$\mathcal{O}\Bigg(|I_A^{\mathsf{Q_{i-1}}}| + |I_A^-| + |I_A^+| + |I_B^{\mathsf{Q_{i-1}}}| + |I_B^-| + |I_B^+|\Bigg) \tag{14}$$

4.1.2 *RTEC$_{inc}$*

In *RTEC$_{inc}$*, we want to produce the intervals of union_all at q_i by using the intervals of union_all calculated at q_{i-1} along with the deletions and insertions of the two body fluents at q_i. The incremental evaluation of union_all is the following:

$$I_F^{\mathsf{Q_i}} = \Bigg[(I^{\mathsf{Q_{i-1}}}_{A\cup_T B}) \underset{\underset{\textcircled{5}}{\downarrow}}{\setminus_T} \Bigg[\Big[(I_A^- \underset{\underset{\textcircled{1}}{\downarrow}}{\cup_T} I_B^-) \underset{\underset{\textcircled{2}}{\downarrow}}{\setminus_T} (I^{\mathsf{Q_{i-1}}}_{A\cap_T B})\Big] \underset{\underset{\textcircled{4}}{\downarrow}}{\cup_T} (I_A^- \underset{\underset{\textcircled{3}}{\downarrow}}{\cap_T} I_B^-)\Bigg]\Bigg] \underset{\underset{\textcircled{7}}{\downarrow}}{\cup_T} (I_A^+ \underset{\underset{\textcircled{6}}{\downarrow}}{\cup_T} I_B^+) \tag{15}$$

The interested reader may refer to the Appendix for the derivation of the incremental formula from the non-incremental one of RTEC. $I^{\mathsf{Q_{i-1}}}_{A\cup_T B}$ represents the result of rule (3) at q_{i-1}, and the term $I^{\mathsf{Q_{i-1}}}_{A\cap_T B}$ represents the common time-points of the two body fluents at q_{i-1}, i.e. their intersection. Both these terms are known at q_i. The circled numbers express just one of the many possible orders of execution and serve as pointers when we want to refer to a specific operation. Moreover, some of these operations may be performed in parallel (e.g. operations ②, ③ and ⑥). First the intervals that must be removed from $I^{\mathsf{Q_{i-1}}}_{A\cup_T B}$ are considered (see operations ①–⑤ in Eq. (15)). We distinguish between two cases:

(a) The time-points belonging to the retracted intervals of both body fluents, A and B (see operation ③ in Eq. (15)), must be removed from the intervals of F that were calculated at the previous query time q_{i-1} (operation ⑤ in Eq. (15)).

(b) The time-points belonging to the retracted intervals of one body fluent, and not belonging to the intervals of the other body fluent that were computed at q_{i-1}(operations ① and ② in Eq. (15)), must be deleted from the intervals of F (operation ⑤ in Eq. (15)).

Subsequently, the union of the insertions of the two body fluents (operation ⑥ in Eq. (15)) is added to the outcome of operation ⑤ in Eq. (15) (operation ⑦ in Eq. (15)). In Table 7 we illustrate the evaluation of Eq. (15) with the aid of the example of Table 6, where we provide the output of each operation.

We proceed with the complexity analysis of Eq. (15). Without loss of generality, assume that $|I_A^{\mathsf{Q_{i-1}}}| \leq |I_B^{\mathsf{Q_{i-1}}}|$. We assume the worst case scenario for all the operations, including those involving retractions and insertions. For the operation $I_A^- \cup_T I_B^-$ the worst case is for the intervals to be disjoint and thus for the output list to be of size $|I_A^-| + |I_B^-|$. This implies that the output of operation $I_A^- \cap_T I_B^-$ will be $\varnothing$. We selected this case as it leads both to the largest cost and output for operation ⑤. In Table 8 we present the cost of each operation of Eq. (15) as well as the size of the output list. The latter facilitates understanding the steps of the complexity analysis, since the output list of an operation participates in a subsequent operation.

Notice that in the complexity analysis presented in Table 8 the cost of operation ④ is zero since the output of operation ③ is empty. Furthermore, the output of the final operation (⑦) is not of interest since it is not used in any other operation. The overall cost of Eq. (15) is produced by summing the cost of all operations and is the following:

$$\mathcal{O}\left(2|I^{\mathsf{Q}_{\mathsf{i-1}}}_{A\cup_T B}| + 3|I^{\mathsf{Q}_{\mathsf{i-1}}}_{A\cap_T B}| + 5(|I^-_A| + |I^-_B|) + 2(|I^+_A| + |I^+_B|)\right) \tag{16}$$

Table 7: Evaluation of Eq. (15) using the running example of Table 6

Operation	**Result**
①	$[(10,12),(19,21),(27,30),(43,47),(60,65)]$
②	$[(10,12),(19,21)]$
③	$[(43,47)]$
④	$[(10,12),(19,21),(43,47)]$
⑤	$[(12,15),(17,19),(23,35),(40,43),(47,50),(54,70)]$
⑥	$[(37,41),(54,58),(80,90),(95,100),(105,120)]$
⑦	$[(12,15),(17,19),(23,35),(37,43),(47,50),$ $(54,70),(80,90),(95,100),(105,120)]$

Table 8: Complexity analysis of Eq. (15)

Operation	**Cost**	**Output**
①	$\lvert I^-_A\rvert + \lvert I^-_B\rvert$	$\lvert I^-_A\rvert + \lvert I^-_B\rvert$
②	$\lvert I^{\mathsf{Q}_{\mathsf{i-1}}}_{A\cap_T B}\rvert + \lvert I^-_A\rvert + \lvert I^-_B\rvert$	$\lvert I^{\mathsf{Q}_{\mathsf{i-1}}}_{A\cap_T B}\rvert + \lvert I^-_A\rvert + \lvert I^-_B\rvert$
③	$\lvert I^-_A\rvert + \lvert I^-_B\rvert$	$\varnothing$
④	0	$\lvert I^{\mathsf{Q}_{\mathsf{i-1}}}_{A\cap_T B}\rvert + \lvert I^-_A\rvert + \lvert I^-_B\rvert$
⑤	$\lvert I^{\mathsf{Q}_{\mathsf{i-1}}}_{A\cup_T B}\rvert + \lvert I^{\mathsf{Q}_{\mathsf{i-1}}}_{A\cap_T B}\rvert + \lvert I^-_A\rvert + \lvert I^-_B\rvert$	$\lvert I^{\mathsf{Q}_{\mathsf{i-1}}}_{A\cap_T B}\rvert + \lvert I^{\mathsf{Q}_{\mathsf{i-1}}}_{A\cap_T B}\rvert + \lvert I^-_A\rvert + \lvert I^-_B\rvert$
⑥	$\lvert I^+_A\rvert + \lvert I^+_B\rvert$	$\lvert I^+_A\rvert + \lvert I^+_B\rvert$
⑦	$\lvert I^{\mathsf{Q}_{\mathsf{i-1}}}_{A\cup_T B}\rvert + \lvert I^{\mathsf{Q}_{\mathsf{i-1}}}_{A\cap_T B}\rvert +$ $\lvert I^-_A\rvert + \lvert I^-_B\rvert + \lvert I^+_A\rvert + \lvert I^+_B\rvert$	-

4.1.3 Comparison of RTEC and $RTEC_{inc}$.

Our goal is to identify the conditions under which the incremental evaluation of statically determined fluents using union_all is more efficient than the computation from scratch (RTEC).

Table 9: $RTEC_{inc}$ vs RTEC: union_all. In all lines below, $|I^-_A| = |I^-_B| = 0$.

$\boldsymbol{\vert I^+_A\vert}$	$\boldsymbol{\vert I^+_B\vert}$	**Conditions in which $\boldsymbol{RTEC_{inc}}$ outperforms RTEC**
$\neq 0$	$\neq 0$	$\vert I^{\mathsf{Q_{i-1}}}_{A\cup_T B}\vert < 2\vert I^{\mathsf{Q_{i-1}}}_{AB}\vert - 2\vert I^{-+}_{AB}\vert$
0	$\neq 0$	$\vert I^{\mathsf{Q_{i-1}}}_{A\cup_T B}\vert < 2\vert I^{\mathsf{Q_{i-1}}}_{AB}\vert$
$\neq 0$	0	$\vert I^{\mathsf{Q_{i-1}}}_{A\cup_T B}\vert < 2\vert I^{\mathsf{Q_{i-1}}}_{AB}\vert$
0	0	$\top$

To achieve this, we compare the cost of RTEC, as given by Eq. (14), and that of $RTEC_{inc}$—see Eq. (16). To facilitate the presentation:

- We will use $|I^{\mathsf{Q_{i-1}}}_{AB}|$, as opposed to $|I^{\mathsf{Q_{i-1}}}_{A}|$ and $|I^{\mathsf{Q_{i-1}}}_{B}|$, to represent the number of intervals of the two body fluents at the previous query time.
- We will sometimes use $|I^{-+}_{AB}|$, as opposed to $|I^-_A|$, $|I^-_B|$, $|I^+_A|$ and $|I^+_B|$, to represent the number of deletions/insertions of the two body fluents.

$RTEC_{inc}$ is preferable to RTEC when:

$$\begin{aligned} \mathcal{O}\left(2|I^{\mathsf{Q_{i-1}}}_{A\cup_T B}| + 3|I^{\mathsf{Q_{i-1}}}_{A\cap_T B}| + 14|I^{-+}_{AB}|\right) &< \mathcal{O}\left(2|I^{\mathsf{Q_{i-1}}}_{AB}| + 4|I^{-+}_{AB}|\right) \Rightarrow \\ |I^{\mathsf{Q_{i-1}}}_{A\cup_T B}| &< |I^{\mathsf{Q_{i-1}}}_{AB}| - \frac{3}{2}|I^{\mathsf{Q_{i-1}}}_{A\cap_T B}| - 5|I^{-+}_{AB}| \end{aligned} \tag{17}$$

Unfortunately, this inequality does not hold in the presence of deletions, even if we assume the best case for the individual operations of Eq. (15). In the case of the absence of deletions, i.e. when $I^-_A = I^-_B = \varnothing$, Eq. (15), expressing the incremental evaluation of union_all, is simplified to the considerably cheaper following operations:

$$I^{\mathsf{Q_{i-1}}}_{A\cup_T B} \cup_T \left[I^+_A \cup_T I^+_B\right] \tag{18}$$

Table 9 displays the conditions in which $RTEC_{inc}$ is preferable to RTEC. Figure 5 illustrates, with the use of an example, the case displayed in the first line of Table 9. $RTEC_{inc}$ is preferred over RTEC as the number of intervals of the two body fluents increases and the number of delayed insertions decreases. The cases in which RTEC is preferable can be found in the complete comparison Table presented in the Appendix.

4.2 Interval Intersection

In this section, we present the incremental evaluation of the interval manipulation construct intersect_all and the benefits in performance.

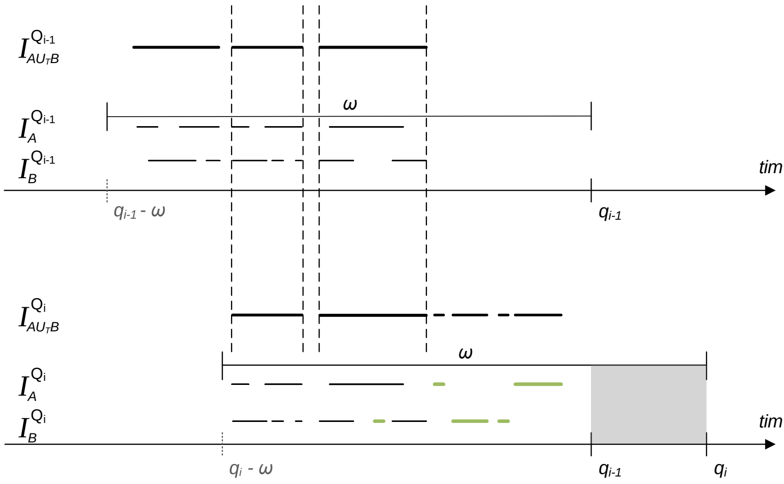


Figure 5: An example where $RTEC_{inc}$ outperforms RTEC. The upper part shows the result of union_all at q_{i-1}, while the bottom part displays the result of union_all at q_i. ω expresses the sliding window. Unlabelled horizontal lines represent fluent intervals. Green lines express fluent intervals inserted in $(q_{i-1}, q_i]$. Vertical dashed lines denote the common intervals of $I_{A\cup_T B}$ at both query times. The non-overlapping part of the two query times, $(q_{i-1}, q_i]$, in the bottom part of the figure is greyed out.

4.2.1 RTEC

If we compute everything from scratch, the cost of evaluating intersect_all is bound by Eq. (14) again. In the worst case, the cost of intersect_all is limited by the sum of the sizes of the two lists of intervals.

4.2.2 *RTEC_{inc}*

We express the intervals of intersect_all at q_i by taking into consideration the result of intersection at q_{i-1}, as well as the intervals of the two body fluents of rule (3) (page 5) that were not part of intersection at q_{i-1}. These body fluent intervals, represented as $I^{\mathsf{Q_{i-1}}}_{A\setminus_T B}$ and $I^{\mathsf{Q_{i-1}}}_{B\setminus_T A}$, may contribute to the evaluation of intersect_all at q_i in the case of delayed insertions. The incremental evaluation of intersect_all is as follows:

$$\begin{aligned} I_F^{\mathsf{Q_i}} = &\Big[I^{\mathsf{Q_{i-1}}}_{A\cap_T B} \underset{\underset{\textcircled{2}}{\downarrow}}{\setminus_T} \Big[(I_A^- \underset{\underset{\textcircled{1}}{\downarrow}}{\cup_T} I_B^-) \Big] \Big] \qquad \underset{\underset{\textcircled{10}}{\downarrow}}{\cup_T} \\ &\Big[\Big[I_B^+ \underset{\underset{\textcircled{5}}{\downarrow}}{\cap_T} \Big[(I^{\mathsf{Q_{i-1}}}_{A\setminus_T B} \underset{\underset{\textcircled{3}}{\downarrow}}{\setminus_T} I_A^-) \underset{\underset{\textcircled{4}}{\downarrow}}{\cup_T} I_A^+ \Big] \Big] \underset{\underset{\textcircled{9}}{\downarrow}}{\cup_T} \Big[I_A^+ \underset{\underset{\textcircled{8}}{\downarrow}}{\cap_T} \Big[(I^{\mathsf{Q_{i-1}}}_{B\setminus_T A} \underset{\underset{\textcircled{6}}{\downarrow}}{\setminus_T} I_B^-) \underset{\underset{\textcircled{7}}{\downarrow}}{\cup_T} I_B^+ \Big] \Big] \Big] \end{aligned} \tag{19}$$

The derivation of the incremental formula of $RTEC_{inc}$ from the non-incremental one of RTEC is presented in the Appendix. Terms $I^{\mathsf{Q_{i-1}}}_{A\setminus_T B}$, $I^{\mathsf{Q_{i-1}}}_{B\setminus_T A}$ and $I^{\mathsf{Q_{i-1}}}_{A\cap_T B}$ are known at q_i. The circled numbers express a sequence of operations leading to the evaluation of Eq. (19).

Table 10: Evaluation of Eq. (19) with the use of the running example of Table 6

Operation	**Result**
①	$[(10,12),(19,21),(27,30),(43,47),(60,65)]$
②	$[(26,27)]$
③	$[(12,15),(23,26),(40,43),(47,50),(65,70)]$
④	$[(12,15),(23,26),(40,43),(47,50),(54,58),(65,70),(80,90),(95,100)]$
⑤	$[(40,41),(82,87)]$
⑥	$[(17,19),(30,35),(54,60)]$
⑦	$[(17,19),(30,35),(37,41),(54,60),(82,87),(105,120)]$
⑧	$[(54,58),(82,87)]$
⑨	$[(40,41),(54,58),(82,87)]$
⑩	$[(26,27),(40,41),(54,58),(82,87)]$

In Eq. (19), we first delete from $I^{\mathsf{Q_{i-1}}}_{A\cap_T B}$ the retracted intervals of $A{=}V_A$ and $B{=}V_B$ (see operations ①–② in Eq. (19)). Next, we compute the inserted intervals of $B{=}V_B$ that belong to $I^{\mathsf{Q_{i-1}}}_{A\setminus_T B}$, after excluding the deletions of $A{=}V_A$, or to the insertions of $A{=}V_A$ (operations ③–⑤). We do the same for the insertions of $A{=}V_A$ (operations ⑥–⑧). Finally, the union of these operations provides the intersection of the intervals of the two fluents at q_i (operations ⑨–⑩). Table 10 illustrates the incremental evaluation of $\mathsf{intersect\_all}$ with the use of the running example displayed in Table 6 on page 19.

For the complexity analysis, we consider, as usual, the highest overall cost and the largest possible output. The conditions that lead to the highest cost are the following:

1. $I^{\mathsf{Q_{i-1}}}_{A\cap_T B}, I^{\mathsf{Q_{i-1}}}_{A\setminus_T B}, I^{\mathsf{Q_{i-1}}}_{B\setminus_T A} \neq \varnothing$.

2. $I^{-}_{A} \cap_T I^{-}_{B} = \varnothing$.

3. $I^{\mathsf{Q_{i-1}}}_{A\cap_T B}$ remains unaffected after operation ② of Eq. (19).

4. I^{-}_{A} splits all the intervals of $I^{\mathsf{Q_{i-1}}}_{A\setminus_T B}$ in operation ③ and I^{-}_{B} splits the intervals of $I^{\mathsf{Q_{i-1}}}_{B\setminus_T A}$ in operation ⑥.

5. I^{+}_{B} overlaps completely $I^{\mathsf{Q_{i-1}}}_{A\setminus_T B} \setminus_T I^{-}_{A}$ and partially I^{+}_{A}. Accordingly I^{+}_{A} overlaps completely $I^{\mathsf{Q_{i-1}}}_{B\setminus_T A} \setminus_T I^{-}_{B}$ and partially I^{+}_{B}.

A detailed analysis of the cost of each operation of Eq. (19) is presented in Table 11, in a similar way as in Section 4.1. The output of operation ⑨ contains the term $\frac{|I^{+}_{A}|+|I^{+}_{B}|}{2}$, since operations ⑤ and ⑧ produce some common intervals, which must be considered only once.

Table 11: Complexity analysis of Eq. (19)

Operation	**Cost**	**Output**
①	$\lvert I_A^-\rvert + \lvert I_B^-\rvert$	$\lvert I_A^-\rvert + \lvert I_B^-\rvert$
②	$\lvert I^{\mathsf{Q_{i-1}}}_{A\cap_T B}\rvert + \lvert I_A^-\rvert + \lvert I_B^-\rvert$	$\lvert I^{\mathsf{Q_{i-1}}}_{A\cap_T B}\rvert$
③	$\lvert I^{\mathsf{Q_{i-1}}}_{A\setminus_T B}\rvert + \lvert I_A^-\rvert$	$\lvert I^{\mathsf{Q_{i-1}}}_{A\setminus_T B}\rvert + \lvert I_A^-\rvert$
④	$\lvert I^{\mathsf{Q_{i-1}}}_{A\setminus_T B}\rvert + \lvert I_A^-\rvert + \lvert I_A^+\rvert$	$\lvert I^{\mathsf{Q_{i-1}}}_{A\setminus_T B}\rvert + \lvert I_A^-\rvert + \lvert I_A^+\rvert$
⑤	$\lvert I^{\mathsf{Q_{i-1}}}_{A\setminus_T B}\rvert + \lvert I_A^-\rvert + \lvert I_A^+\rvert + \lvert I_B^+\rvert$	$\lvert I^{\mathsf{Q_{i-1}}}_{A\setminus_T B}\rvert + \lvert I_A^-\rvert + \lvert I_A^+\rvert$
⑥	$\lvert I^{\mathsf{Q_{i-1}}}_{B\setminus_T A}\rvert + \lvert I_B^-\rvert$	$\lvert I^{\mathsf{Q_{i-1}}}_{B\setminus_T A}\rvert + \lvert I_B^-\rvert$
⑦	$\lvert I^{\mathsf{Q_{i-1}}}_{B\setminus_T A}\rvert + \lvert I_B^-\rvert + \lvert I_B^+\rvert$	$\lvert I^{\mathsf{Q_{i-1}}}_{B\setminus_T A}\rvert + \lvert I_B^-\rvert + \lvert I_B^+\rvert$
⑧	$\lvert I^{\mathsf{Q_{i-1}}}_{B\setminus_T A}\rvert + \lvert I_B^-\rvert + \lvert I_B^+\rvert + \lvert I_A^+\rvert$	$\lvert I^{\mathsf{Q_{i-1}}}_{B\setminus_T A}\rvert + \lvert I_B^-\rvert + \lvert I_B^+\rvert$
⑨	$\lvert I^{\mathsf{Q_{i-1}}}_{A\setminus_T B}\rvert + \lvert I_A^-\rvert + \lvert I_A^+\rvert +$ $\lvert I^{\mathsf{Q_{i-1}}}_{B\setminus_T A}\rvert + \lvert I_B^-\rvert + \lvert I_B^+\rvert$	$\lvert I^{\mathsf{Q_{i-1}}}_{A\setminus_T B}\rvert + \lvert I_A^-\rvert + \lvert I^{\mathsf{Q_{i-1}}}_{B\setminus_T A}\rvert +$ $\lvert I_B^-\rvert + \frac{\lvert I_A^+\rvert + \lvert I_B^+\rvert}{2}$
⑩	$\lvert I^{\mathsf{Q_{i-1}}}_{A\cap_T B}\rvert + \lvert I^{\mathsf{Q_{i-1}}}_{A\setminus_T B}\rvert + \lvert I_A^-\rvert +$ $\lvert I^{\mathsf{Q_{i-1}}}_{B\setminus_T A}\rvert + \lvert I_B^-\rvert + \frac{\lvert I_A^+\rvert + \lvert I_B^+\rvert}{2}$	-

Those intervals are the result of the intersection of I_A^+ and I_B^+. By summing the costs of all operations, we compute the bound of incremental intersect_all:

$$\mathcal{O}\Bigg(2\lvert I^{\mathsf{Q_{i-1}}}_{A\cap_T B}\rvert + 5(\lvert I^{\mathsf{Q_{i-1}}}_{A\setminus_T B}\rvert + \lvert I^{\mathsf{Q_{i-1}}}_{B\setminus_T A}\rvert) + \tfrac{9}{2}(\lvert I_A^+\rvert + \lvert I_B^+\rvert) + 7(\lvert I_A^-\rvert + \lvert I_B^-\rvert)\Bigg) \tag{20}$$

4.2.3 Comparison of RTEC and *RTEC$_{inc}$*.

We compare the cost of RTEC as given by Eq. (14), and that of *RTEC$_{inc}$*— see Eq. (20). To facilitate the presentation, we simplify notation as follows:

- We will use $\lvert I^{\mathsf{Q_{i-1}}}_{AB}\rvert$, as opposed to $\lvert I^{\mathsf{Q_{i-1}}}_{A}\rvert$ and $\lvert I^{\mathsf{Q_{i-1}}}_{B}\rvert$, to represent the number of intervals of the two body fluents at the previous query time.

- We will use $\lvert I^{\mathsf{Q_{i-1}}}_{A\setminus_T B}\rvert$ to denote $\lvert I^{\mathsf{Q_{i-1}}}_{A\setminus_T B}\rvert$ and $\lvert I^{\mathsf{Q_{i-1}}}_{B\setminus_T A}\rvert$.

- We will sometimes use $\lvert I^{-+}_{AB}\rvert$, as opposed to $\lvert I_A^-\rvert$, $\lvert I_B^-\rvert$, $\lvert I_A^+\rvert$ and $\lvert I_B^+\rvert$, to represent the number of deletions/insertions of the two body fluents.

Table 12: *RTEC$_{inc}$* vs RTEC: intersect_all.

$\|I_A^-\|$	$\|I_A^+\|$	$\|I_B^-\|$	$\|I_B^+\|$	**Conditions in which *RTEC$_{inc}$* outperforms RTEC**
$\neq 0$	0	$\neq 0$	$\neq 0$	$\|I^{\mathsf{Q_{i-1}}}_{A\cap_T B}\| < \|I^{\mathsf{Q_{i-1}}}_{AB}\| - \frac{3}{2}\|I^{\mathsf{Q_{i-1}}}_{A\setminus_T B}\| - 2\|I^{-+}_{AB}\|$
0	0	$\neq 0$	$\neq 0$	$\|I^{\mathsf{Q_{i-1}}}_{A\cap_T B}\| < \|I^{\mathsf{Q_{i-1}}}_{AB}\| - \|I^{\mathsf{Q_{i-1}}}_{A\setminus_T B}\|$
0	$\neq 0$	0	$\neq 0$	$\|I^{\mathsf{Q_{i-1}}}_{A\cap_T B}\| < 2\|I^{\mathsf{Q_{i-1}}}_{AB}\| - 8\|I^{\mathsf{Q_{i-1}}}_{A\setminus_T B}\| - \frac{7}{2}\|I^{-+}_{AB}\|$
$\neq 0$	0	0	$\neq 0$	$\|I^{\mathsf{Q_{i-1}}}_{A\cap_T B}\| < \|I^{\mathsf{Q_{i-1}}}_{AB}\| - \frac{3}{2}\|I^{\mathsf{Q_{i-1}}}_{A\setminus_T B}\| - \frac{3}{2}\|I^{-+}_{AB}\|$
0	0	0	$\neq 0$	$\|I^{\mathsf{Q_{i-1}}}_{A\cap_T B}\| < 2\|I^{\mathsf{Q_{i-1}}}_{AB}\| - 2\|I^{\mathsf{Q_{i-1}}}_{A\setminus_T B}\|$
$\neq 0$	$\neq 0$	$\neq 0$	0	$\|I^{\mathsf{Q_{i-1}}}_{A\cap_T B}\| < \|I^{\mathsf{Q_{i-1}}}_{AB}\| - \frac{3}{2}\|I^{\mathsf{Q_{i-1}}}_{A\setminus_T B}\| - 2\|I^{-+}_{AB}\|$
0	$\neq 0$	$\neq 0$	0	$\|I^{\mathsf{Q_{i-1}}}_{A\cap_T B}\| < \|I^{\mathsf{Q_{i-1}}}_{AB}\| - \frac{3}{2}\|I^{\mathsf{Q_{i-1}}}_{A\setminus_T B}\| - \frac{3}{2}\|I^{-+}_{AB}\|$
$\neq 0$	0	$\neq 0$	0	$\|I^{\mathsf{Q_{i-1}}}_{A\cap_T B}\| < 2\|I^{\mathsf{Q_{i-1}}}_{AB}\| - 2\|I^{-+}_{AB}\|$
0	0	$\neq 0$	0	$\|I^{\mathsf{Q_{i-1}}}_{A\cap_T B}\| < 2\|I^{\mathsf{Q_{i-1}}}_{AB}\|$
$\neq 0$	$\neq 0$	0	0	$\|I^{\mathsf{Q_{i-1}}}_{A\cap_T B}\| < \|I^{\mathsf{Q_{i-1}}}_{AB}\| - \|I^{\mathsf{Q_{i-1}}}_{A\setminus_T B}\|$
0	$\neq 0$	0	0	$\|I^{\mathsf{Q_{i-1}}}_{A\cap_T B}\| < 2\|I^{\mathsf{Q_{i-1}}}_{AB}\| - 2\|I^{\mathsf{Q_{i-1}}}_{A\setminus_T B}\|$
$\neq 0$	0	0	0	$\|I^{\mathsf{Q_{i-1}}}_{A\cap_T B}\| < 2\|I^{\mathsf{Q_{i-1}}}_{AB}\|$
0	0	0	0	$\top$

Then, *RTEC$_{inc}$* is preferable to RTEC when:

$$\begin{aligned} \mathcal{O}\Big(2|I^{\mathsf{Q_{i-1}}}_{AB}| + 4|I^{-+}_{AB}|\Big) \;&>\; \mathcal{O}\Big(2|I^{\mathsf{Q_{i-1}}}_{A\cap_T B}| + 10|I^{\mathsf{Q_{i-1}}}_{A\setminus_T B}| + 23|I^{-+}_{AB}|\Big) \quad \Rightarrow \\ |I^{\mathsf{Q_{i-1}}}_{A\cap_T B}| \;&<\; |I^{\mathsf{Q_{i-1}}}_{AB}| - 5|I^{\mathsf{Q_{i-1}}}_{A\setminus_T B}| - \frac{19}{2}|I^{-+}_{AB}| \end{aligned} \tag{21}$$

Unfortunately, the above inequality cannot be satisfied. In other words, when the conditions 1–5 stated above hold, RTEC is always the best option. Table 12 presents those cases in which *RTEC$_{inc}$* outperforms RTEC. In Figure 6, we display an example that illustrates one of the cases. The complete comparison can be found in the Appendix.

In the absence of deletions, as in the example of Figure 6, Eq. (19) may be simplified to:

$$I^{\mathsf{Q_{i-1}}}_{A\cap_T B} \cup_T \Big[I^+_A \cap_T I^{\mathsf{Q_i}}_B\Big] \cup_T \Big[I^+_B \cap_T I^{\mathsf{Q_i}}_A\Big] \tag{22}$$

This is a much cheaper computation of intersect_all as opposed to Eq (19).

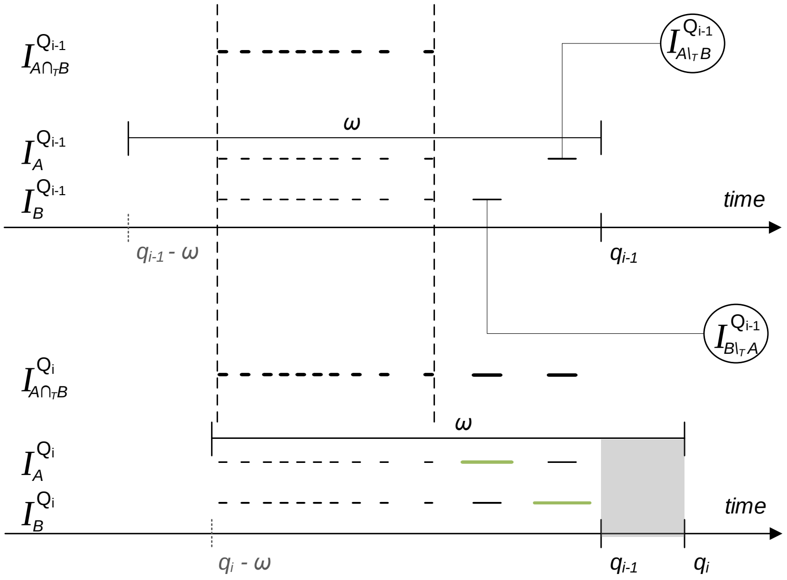


Figure 6: An example where $RTEC_{inc}$ outperforms RTEC. This corresponds to the third line of Table 12. ω expresses the sliding window. The upper part of the figure shows the result of intersect_all at q_{i-1}, while the bottom part displays the result of intersect_all at q_i. Unlabelled horizontal lines represent fluent intervals. Green lines express fluent intervals inserted in $(q_{i-1}, q_i]$. Vertical dashed lines denote the common intervals of the two body fluents, $I_{A\cap_T B}$, at both query times.

4.3 Relative Complement

The last interval manipulation construct of RTEC is relative_complement_all. We examine the case of the relative complement of $A{=}V_A$ with respect to $B{=}V_B$.

4.3.1 RTEC

Since we refer to two fluents, the computation from scratch of RTEC is bound once again by Eq. (14) (Artikis et al., 2015).

4.3.2 $RTEC_{inc}$

In the incremental case, we express the new intervals produced by relative_complement_all by taking into consideration not only the result of relative_complement_all at q_{i-1} but also the intervals at q_{i-1} that hold for fluent B and not for A, as well as their common intervals. We represent these two sets of intervals as $I_{B\setminus_T A}^{\mathsf{Q}_{i-1}}$ and $I_{A\cap_T B}^{\mathsf{Q}_{i-1}}$, respectively. The incremental formula of relative_complement_all is the following:

$$\begin{aligned} I_F^{\mathsf{Q}_{\mathsf{i}}} = &\left[I^{\mathsf{Q}_{\mathsf{i}-1}}_{A\setminus_T B} \underset{\textcircled{2}}{\underset{\downarrow}{\setminus_T}} (I_A^- \underset{\textcircled{1}}{\underset{\downarrow}{\cup_T}} I_B^+) \right] \quad \underset{\textcircled{10}}{\underset{\downarrow}{\cup_T}} \\ &\left[\left[I_A^+ \underset{\textcircled{5}}{\underset{\downarrow}{\setminus_T}} \left[(I^{\mathsf{Q}_{\mathsf{i}-1}}_{B\setminus_T A} \underset{\textcircled{3}}{\underset{\downarrow}{\setminus_T}} I_B^-) \underset{\textcircled{4}}{\underset{\downarrow}{\cup_T}} I_B^+\right]\right] \underset{\textcircled{9}}{\underset{\downarrow}{\cup_T}} \left[I_B^- \underset{\textcircled{8}}{\underset{\downarrow}{\cap_T}} \left[(I^{\mathsf{Q}_{\mathsf{i}-1}}_{A\cap_T B} \underset{\textcircled{6}}{\underset{\downarrow}{\setminus_T}} I_A^-) \underset{\textcircled{7}}{\underset{\downarrow}{\cup_T}} I_A^+\right]\right]\right] \end{aligned} \tag{23}$$

The derivation of Eq. (23) from the non-incremental one of RTEC may be found in the Appendix. First, the intervals that have been inserted for $B{=}V_B$ or deleted from $A{=}V_A$ and belong to $I^{\mathsf{Q}_{\mathsf{i}-1}}_{A\setminus_T B}$, must be retracted (see operations ①–② in Eq. (23)). Next, we calculate which of the inserted intervals of $A{=}V_A$ are not in $I^{\mathsf{Q}_{\mathsf{i}-1}}_{B\setminus_T A}$, after removing the deletions of $B{=}V_B$, and/or in the insertions of $B{=}V_B$ (see operations ③–⑤). Subsequently, we include in the result the retracted intervals of $B{=}V_B$ that are among the inserted intervals of $A{=}V_A$ or belong to $I^{\mathsf{Q}_{\mathsf{i}-1}}_{A\cap_T B}$, after discarding the deletions of $A{=}V_A$ (see operations ⑥–⑧). Finally, the union of the results of these operations constitutes the result of relative complement of $A{=}V_A$ with respect to $B{=}V_B$ at q_i (operations ⑨–⑩). Table 13 illustrates the evaluation of Eq. (23) with the use of our running example (see Table 6 on page 19).

The worst case, in terms of complexity, concerning the evaluation of incremental relative_complement_all, arises when the following conditions hold:

1. $I^{\mathsf{Q}_{\mathsf{i}-1}}_{A\cap_T B}, I^{\mathsf{Q}_{\mathsf{i}-1}}_{A\setminus_T B}, I^{\mathsf{Q}_{\mathsf{i}-1}}_{B\setminus_T A} \neq \varnothing$.

2. $I_A^- \cap_T I_B^+ = \varnothing$.

3. $I^{\mathsf{Q}_{\mathsf{i}-1}}_{A\setminus_T B}$ remains unaffected after operation ② of Eq. (23).

4. $I^{\mathsf{Q}_{\mathsf{i}-1}}_{B\setminus_T A}$ remains unaffected after operation ③ of Eq. (23).

5. I_A^- splits all the intervals of $I^{\mathsf{Q}_{\mathsf{i}-1}}_{A\cap_T B}$ (operation ⑥ of Eq. (23)).

6. $I^{\mathsf{Q}_{\mathsf{i}-1}}_{B\setminus_T A}, I_B^+ \subseteq_T I_A^+$.

7. $I_B^- \subseteq_T (I^{\mathsf{Q}_{\mathsf{i}-1}}_{A\cap_T B} \setminus_T I_A^-)$ and $I_B^- \cap_T I_A^+ = \varnothing$. This condition implies that $I^{\mathsf{Q}_{\mathsf{i}-1}}_{A\cap_T B}$ is produced by the intervals of $A{=}V_A$ and $B{=}V_B$ at q_{i-1} that are exactly the same.

Table 14 presents the steps of the complexity analysis. The cost of incremental relative_complement_all is bound by:

$$\mathcal{O}\left(2|I^{\mathsf{Q}_{\mathsf{i}-1}}_{A\setminus_T B}| + 5(|I^{\mathsf{Q}_{\mathsf{i}-1}}_{B\setminus_T A}| + |I^{\mathsf{Q}_{\mathsf{i}-1}}_{A\cap_T B}|) + 5|I_A^+| + 6|I_B^+| + 7|I_A^-| + 2|I_B^-|\right) \tag{24}$$

4.3.3 Comparison of RTEC and *RTEC$_{inc}$*.

We compare the cost of RTEC as given by Eq. (14), and that of *RTEC$_{inc}$*. To facilitate the presentation we make the usual simplifications in notation:

Table 13: Evaluation of Eq. (23) with the use of the running example of Table 6

Operation	**Result**
①	$[(10,12),(27,30),(37,41),(43,47),(82,87),(105,120)]$
②	$[(12,15),(23,26),(41,43),(47,50),(65,70)]$
③	$[(17,19),(30,35),(54,60)]$
④	$[(17,19),(30,35),(37,41),(54,60),(82,87),(105,120)]$
⑤	$[(80,82),(87,90),(95,100)]$
⑥	$[(26,27),(60,65)]$
⑦	$[(26,27),(54,58),(60,65),(80,90),(95,100)]$
⑧	$[(60,65)]$
⑨	$[(60,65),(80,82),(87,90),(95,100)]$
⑩	$[(12,15),(23,26),(41,43),(47,50),(60,70),(80,82),(87,90),(95,100)]$

Table 14: Complexity analysis of Eq. (24)

Operation	**Cost**	**Output**
①	$\lvert I_A^-\rvert + \lvert I_B^+\rvert$	$\lvert I_A^-\rvert + \lvert I_B^+\rvert$
②	$\lvert I_{A\setminus_T B}^{\mathsf{Q_{i-1}}}\rvert + \lvert I_A^-\rvert + \lvert I_B^+\rvert$	$\lvert I_{A\setminus_T B}^{\mathsf{Q_{i-1}}}\rvert$
③	$\lvert I_{B\setminus_T A}^{\mathsf{Q_{i-1}}}\rvert + \lvert I_B^-\rvert$	$\lvert I_{B\setminus_T A}^{\mathsf{Q_{i-1}}}\rvert$
④	$\lvert I_{B\setminus_T A}^{\mathsf{Q_{i-1}}}\rvert + \lvert I_B^+\rvert$	$\lvert I_{B\setminus_T A}^{\mathsf{Q_{i-1}}}\rvert + \lvert I_B^+\rvert$
⑤	$\lvert I_{B\setminus_T A}^{\mathsf{Q_{i-1}}}\rvert + \lvert I_B^+\rvert + \lvert I_A^+\rvert$	$\lvert I_{B\setminus_T A}^{\mathsf{Q_{i-1}}}\rvert + \lvert I_B^+\rvert + \lvert I_A^+\rvert$
⑥	$\lvert I_{A\cap_T B}^{\mathsf{Q_{i-1}}}\rvert + \lvert I_A^-\rvert$	$\lvert I_{A\cap_T B}^{\mathsf{Q_{i-1}}}\rvert + \lvert I_A^-\rvert$
⑦	$\lvert I_{A\cap_T B}^{\mathsf{Q_{i-1}}}\rvert + \lvert I_A^-\rvert + \lvert I_A^+\rvert$	$\lvert I_{A\cap_T B}^{\mathsf{Q_{i-1}}}\rvert + \lvert I_A^-\rvert + \lvert I_A^+\rvert$
⑧	$\lvert I_{A\cap_T B}^{\mathsf{Q_{i-1}}}\rvert + \lvert I_A^-\rvert + \lvert I_A^+\rvert + \lvert I_B^-\rvert$	$\lvert I_{A\cap_T B}^{\mathsf{Q_{i-1}}}\rvert + \lvert I_A^-\rvert$
⑨	$\lvert I_{B\setminus_T A}^{\mathsf{Q_{i-1}}}\rvert + \lvert I_B^+\rvert + \lvert I_A^+\rvert + \lvert I_{A\cap_T B}^{\mathsf{Q_{i-1}}}\rvert + \lvert I_A^-\rvert$	$\lvert I_{B\setminus_T A}^{\mathsf{Q_{i-1}}}\rvert + \lvert I_{A\cap_T B}^{\mathsf{Q_{i-1}}}\rvert + \lvert I_A^-\rvert + \lvert I_A^+\rvert + \lvert I_B^+\rvert$
⑩	$\lvert I_{A\setminus_T B}^{\mathsf{Q_{i-1}}}\rvert + \lvert I_{B\setminus_T A}^{\mathsf{Q_{i-1}}}\rvert + \lvert I_{A\cap_T B}^{\mathsf{Q_{i-1}}}\rvert + \lvert I_A^-\rvert + \lvert I_A^+\rvert + \lvert I_B^+\rvert$	-

Table 15: *RTEC*

$\lvert I_A^-\rvert$	$\lvert I_A^+\rvert$	$\lvert I_B^-\rvert$	$\lvert I_B^+\rvert$	**Conditions in which $\boldsymbol{RTEC_{inc}}$ outperforms RTEC**
$\neq 0$	0	$\neq 0$	$\neq 0$	$\lvert I^{\mathsf{Q_{i-1}}}_{A\setminus_T B}\rvert < \lvert I^{\mathsf{Q_{i-1}}}_{AB}\rvert - \frac{3}{2}\lvert I^{\mathsf{Q_{i-1}}}_{A\cap_T B}\rvert - \frac{5}{2}\lvert I^{-+}_{AB}\rvert$
0	0	$\neq 0$	$\neq 0$	$\lvert I^{\mathsf{Q_{i-1}}}_{A\setminus_T B}\rvert < \lvert I^{\mathsf{Q_{i-1}}}_{AB}\rvert - \lvert I^{\mathsf{Q_{i-1}}}_{A\cap_T B}\rvert$
$\neq 0$	$\neq 0$	0	$\neq 0$	$\lvert I^{\mathsf{Q_{i-1}}}_{A\setminus_T B}\rvert < \lvert I^{\mathsf{Q_{i-1}}}_{AB}\rvert - \frac{3}{2}\lvert I^{\mathsf{Q_{i-1}}}_{B\setminus_T A}\rvert - 3\lvert I^{-+}_{AB}\rvert$
0	$\neq 0$	0	$\neq 0$	$\lvert I^{\mathsf{Q_{i-1}}}_{A\setminus_T B}\rvert < \lvert I^{\mathsf{Q_{i-1}}}_{AB}\rvert - \frac{3}{2}\lvert I^{\mathsf{Q_{i-1}}}_{B\setminus_T A}\rvert - 2\lvert I^{-+}_{AB}\rvert$
$\neq 0$	0	0	$\neq 0$	$\lvert I^{\mathsf{Q_{i-1}}}_{A\setminus_T B}\rvert < 2\lvert I^{\mathsf{Q_{i-1}}}_{AB}\rvert - 2\lvert I^{-+}_{AB}\rvert$
0	0	0	$\neq 0$	$\lvert I^{\mathsf{Q_{i-1}}}_{A\setminus_T B}\rvert < 2\lvert I^{\mathsf{Q_{i-1}}}_{AB}\rvert$
$\neq 0$	0	$\neq 0$	0	$\lvert I^{\mathsf{Q_{i-1}}}_{A\setminus_T B}\rvert < \lvert I^{\mathsf{Q_{i-1}}}_{AB}\rvert - \frac{3}{2}\lvert I^{\mathsf{Q_{i-1}}}_{A\cap_T B}\rvert - \frac{3}{2}\lvert I^{-+}_{AB}\rvert$
0	0	$\neq 0$	0	$\lvert I^{\mathsf{Q_{i-1}}}_{A\setminus_T B}\rvert < 2\lvert I^{\mathsf{Q_{i-1}}}_{AB}\rvert - 2\lvert I^{\mathsf{Q_{i-1}}}_{A\cap_T B}\rvert$
$\neq 0$	$\neq 0$	0	0	$\lvert I^{\mathsf{Q_{i-1}}}_{A\setminus_T B}\rvert < \lvert I^{\mathsf{Q_{i-1}}}_{AB}\rvert - \lvert I^{\mathsf{Q_{i-1}}}_{B\setminus_T A}\rvert - \frac{\lvert I^{-+}_{AB}\rvert}{2}$
0	$\neq 0$	0	0	$\lvert I^{\mathsf{Q_{i-1}}}_{A\setminus_T B}\rvert < 2\lvert I^{\mathsf{Q_{i-1}}}_{AB}\rvert - 2\lvert I^{\mathsf{Q_{i-1}}}_{B\setminus_T B}\rvert - \frac{\lvert I^{-+}_{AB}\rvert}{2}$
$\neq 0$	0	0	0	$\lvert I^{\mathsf{Q_{i-1}}}_{A\setminus_T B}\rvert < 2\lvert I^{\mathsf{Q_{i-1}}}_{AB}\rvert$
0	0	0	0	$\top$

- We will use $|I^{\mathsf{Q_{i-1}}}_{AB}|$, as opposed to $|I^{\mathsf{Q_{i-1}}}_{A}|$ and $|I^{\mathsf{Q_{i-1}}}_{B}|$, to represent the number of intervals of the two body fluents at the previous query time.

- We will use $|I^{-+}_{AB}|$, as opposed to $|I^-_A|$, $|I^-_B|$, $|I^+_A|$ and $|I^+_B|$, to represent the number of deletions/insertions of the two body fluents.

RTEC

$$\begin{gathered}\mathcal{O}\Big(2|I^{\mathsf{Q_{i-1}}}_{AB}| + 4|I^{-+}_{AB}|\Big) > \mathcal{O}\Big(2|I^{\mathsf{Q_{i-1}}}_{A\setminus_T B}| + 5(|I^{\mathsf{Q_{i-1}}}_{B\setminus_T A}| + |I^{\mathsf{Q_{i-1}}}_{A\cap_T B}|) + 20|I^{-+}_{AB}|\Big) \quad \Rightarrow \\ |I^{\mathsf{Q_{i-1}}}_{A\setminus_T B}| < |I^{\mathsf{Q_{i-1}}}_{AB}| - \frac{5}{2}(|I^{\mathsf{Q_{i-1}}}_{B\setminus_T A}| + |I^{\mathsf{Q_{i-1}}}_{A\cap_T B}|) - 8|I^{-+}_{AB}|\end{gathered} \tag{25}$$

The above inequality can be satisfied only if the deletion and insertion sets of the two body fluents of rule (3) (page 5) are empty. In other words, in the wost case, RTEC is the preferred option over *RTEC*

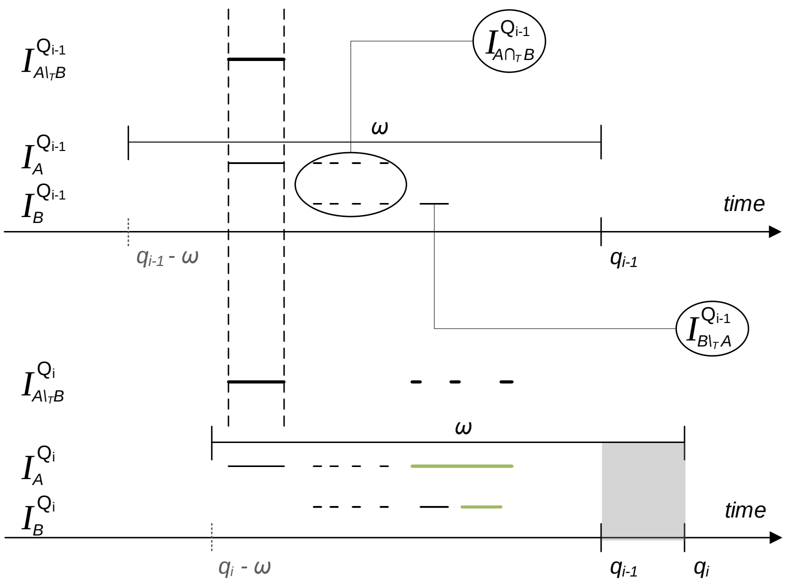


Figure 7: An example where *RTEC$_{inc}$* outperforms RTEC with respect to interval manipulation construct relative_complement_all. The upper part of the figure shows the result of relative_complement_all at q_{i-1}, while the bottom part the result of relative_complement_all at q_i. Unlabelled horizontal lines represent fluent intervals. Green lines express fluent intervals inserted in $(q_{i-1}, q_i]$. Vertical dashed lines denote the common intervals of $I_{A\setminus_T B}$ at both query times.

In the absence of deletions, we simplify the incremental evaluation of relative_complement_all to the much cheaper formula below:

$$\left[I^{\mathsf{Q}_{i-1}}_{A\setminus_T B} \setminus_T I^{+}_{B}\right] \cup_T \left[I^{+}_{A} \setminus_T I^{\mathsf{Q}_i}_{B}\right] \tag{26}$$

5. Empirical Analysis

In this section, we present our empirical analysis on real and synthetic datasets from the fields of maritime monitoring and fleet management.

5.1 Experimental Setup

RTEC and *RTEC$_{inc}$* are pure Prolog implementations — the source code of both systems is publicly available[3,1]. RTEC already uses various optimisation techniques for efficient stream reasoning, such as a rule compilation stage that rewrites the rules in order to take advantage of Prolog's built-in indexing mechanism. Details about RTEC may be found in Artikis et al. (2015) and the manual of RTEC[4]. *RTEC$_{inc}$* inherits all optimisation techniques

[3] `https://github.com/aartikis/RTEC`

[4] `https://github.com/aartikis/RTEC/blob/master/RTEC_manual.pdf`

of RTEC. A difference in the implementation of $RTEC_{inc}$ with respect to RTEC concerns the use of Prolog's internal database. The assertion and retraction of clauses is performed by using the internal database in $RTEC_{inc}$ and the usual way, i.e. with the use of *assertz* and *retract* predicates, in RTEC.

The datasets used in the empirical analysis are stored as CSV files, where each line corresponds to an input event (indicative datasets are available with the code of $RTEC_{inc}$[1]). We simulated a streaming behavior by consuming the input events little by little, i.e. reading small chunks periodically from the CSV files according to slide step specifications. The experiments were designed with the goal of highlighting the gain in performance $RTEC_{inc}$ can bring to Composite Event Recognition (CER) in comparison to RTEC. To demonstrate the effect of the overlap between consecutive query times, we employ five temporal windows with a constant step in all experiments. The experiments were performed on a computer with 24 cores (Intel(R) Xeon(R) CPU E5-2630 v2 @ 2.60GHz) and 252 GB of RAM, running Debian GNU/Linux 9 with Kernel 4.9.0-7-amd64 and YAP Prolog 6.2.2.

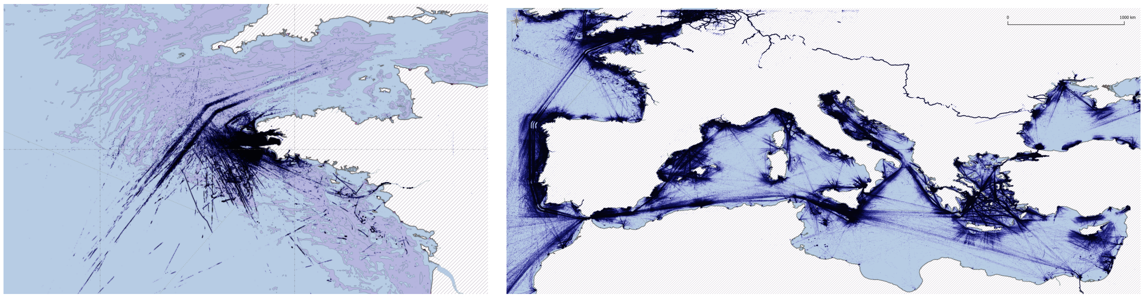
5.2 Maritime Situational Awareness

Maritime situational awareness concerns the recognition of compostite maritime events, such as ship-to-ship transfer, fishing and dangerous sailing, and is typically achieved by monitoring the messages vessels emit while sailing at sea. These messages are exchanged through the Automatic Identification System (AIS[5]) and contain information about the position, heading, speed, etc of vessels at different points in time. The timestamp of an AIS message refers to the time of the measurement on board. By using temporally sorted AIS messages one may reconstruct the trajectory of a vessel. Moreover, it has been shown that the trajectory of a vessel may be compressed by retaining only a subset of the AIS messages, called 'critical points', which allow for an accurate trajectory reconstruction (Patroumpas et al., 2017). These critical points convey information about the start/end of sailing at a low/high speed, changes in speed/heading, notifications about communication gaps, turns, etc. Furthermore, the critical points can be spatially processed in order to discover various relations, such as the entrance or exit of a vessel in an area of interest, and the proximity of two vessels (Santipantakis et al., 2018). The critical points along with the spatial relations constitute the input (SDEs) to our system.

The composite event description used in our empirical analysis includes sixteen simple fluents and five statically determined fluents, forming a hierarchy displayed in Figure 1 on page 7. The definitions of the statically determined fluents contain three operations of union_all, three operations of intersect_all, and two operations of relative_complement_all.

Terrestrial and satellite stations collect and forward the AIS messages emitted by the vessels to the CER system. The stations have to deal with a large number of messages and due to this high traffic significant delays may occur. The CER system must handle in an efficient way the delayed arrival of events in order to minimize the recognition loss. Our empirical evaluation includes delayed SDEs, but not SDE retraction, as SDEs are not retracted in maritime monitoring. However, delayed fluent interval retraction (and assertion) may take place in the higher levels of the fluent hierarchy (see Figure 1 on page 7). Consider, for example, the simple fluent *gap* in Figure 1. Assume that at the

[5] http://www.imo.org/en/OurWork/Safety/Navigation/Pages/AIS.aspx



(a) Brest, France: Synthetic delays dataset

(b) Europe: Natural delays dataset

Figure 8: Position signals of the two datasets (Pitsikalis et al., 2019).

current query time q_i new intervals are computed for *gap*. This may lead to the retraction of intervals for fluents *stopped* and *lowSpeed* since *gap* participates negatively in the body of these fluents. Furthermore, the retraction of intervals for the fluents *stopped* and *lowSpeed* will lead to the retraction of intervals for the fluent *rendezVous*.

We employed two datasets for our empirical analysis; the first is a publicly available dataset[6] concerning the area of Brest, France, and the second concerns all European seas, and was provided to us by IMIS Global[7], our partner in the datAcron project[8]. The geographical coverage of the datasets is displayed in Figure 8. Both datasets are temporally sorted. As a result, they did not include delays. For the dataset of Europe, we additionally obtained the AIS messages with timestamps that reveal the time the message left the terrestrial/satellite station. These timestamps differ from the timestamps recorded on board. By using the timestamps of the terrestrial/satellite stations and sorting the dataset we are able to retain the natural delays. For the Brest dataset these timestamps were not available and the experimental evaluation was performed by introducing synthetic delays.

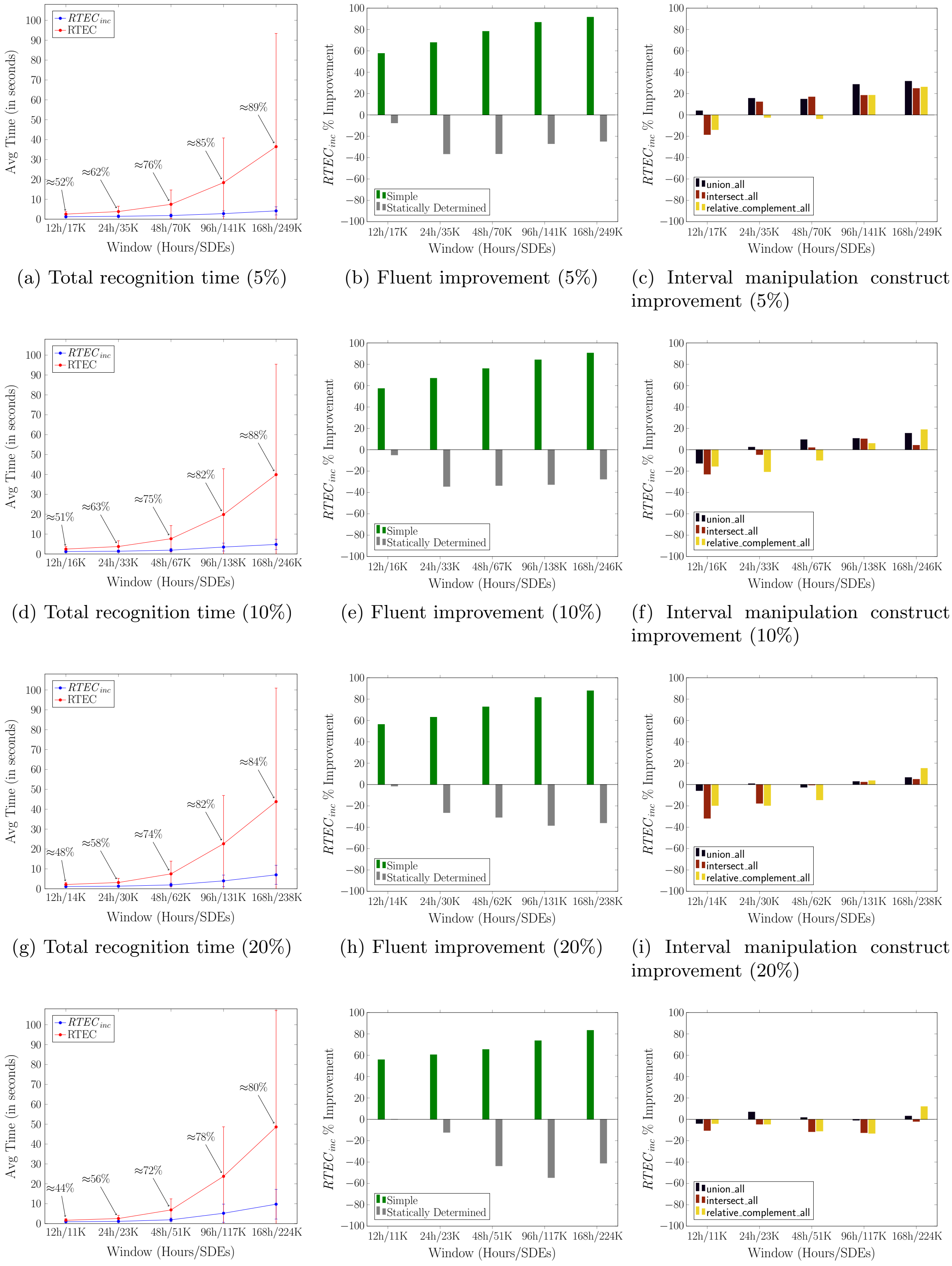
5.2.1 Synthetic delays

The first dataset concerns approx. $5K$ vessels sailing in the Atlantic Ocean around the port of Brest, France, and spans a period from 1 October 2015 to 31 March 2016 (Ray et al., 2019). The dataset consists of approx. $5M$ critical and spatial SDEs. The delays in this dataset cannot be recovered and an approach of artificially injecting delays was adopted. We performed a series of 5 experiments, each time differentiating the amount of SDEs being delayed. We selected uniformly 5%, 10%, 20%, 40% and 80% of the total events to be delayed. We used a uniform distribution for selecting events, since we assume that each event has the same probability to be delayed. Regarding the magnitude of the delay, in the real world, delays of few hours are more probable to happen, even though delays of over 16 hours have been observed in the maritime domain (Camossi et al., 2017; Ray et al., 2016). In order to mimic reality, as much as possible, we used a Gamma distribution with shape parameter $k = 2$ and scale parameter $\theta = 2$. Thus, a small delay had a higher probability

[6] `https://zenodo.org/record/1167595/#.X5aQwC8RrOQ`

[7] `https://imisglobal.com`

[8] `http://datacron-project.eu`



(a) Total recognition time (5%)

(b) Fluent improvement (5%)

(c) Interval manipulation construct improvement (5%)

(d) Total recognition time (10%)

(e) Fluent improvement (10%)

(f) Interval manipulation construct improvement (10%)

(g) Total recognition time (20%)

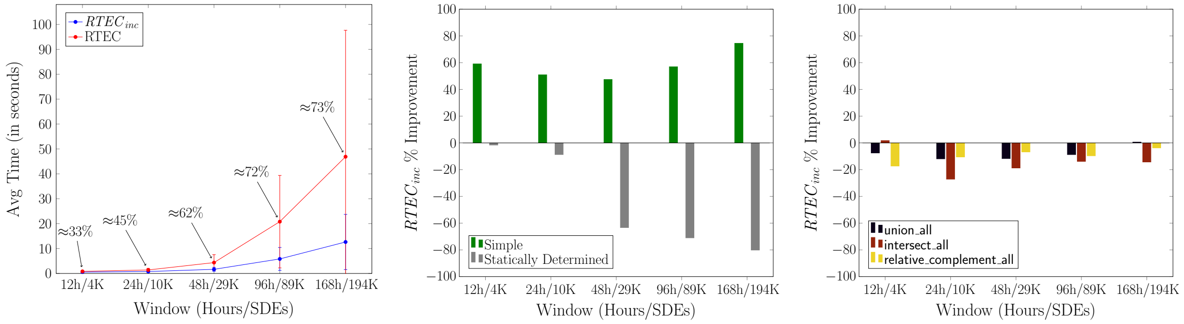
(h) Fluent improvement (20%)

(i) Interval manipulation construct improvement (20%)

(j) Total recognition time (40%)

(k) Fluent improvement (40%)

(l) Interval manipulation construct improvement (40%)



(m) Total recognition time (80%) (n) Fluent improvement (80%) (o) Interval manipulation construct improvement (80%)

Figure 9: Results on the synthetic delays dataset (slide step of 12 hours). Each row of diagrams corresponds to a different amount of delayed SDEs (5%, 10%, 20%, 40%, 80%).

to be imposed in a selected event. The average delay time in all settings was approximately 96 hours.

Even though this dataset spans a period of six months, it does not contain a large number of AIS messages for each vessel. We employed large temporal windows, that usually are not found in practice, in order to stress test the CER systems. Figure 9 presents the results where the sliding window varies from 12 to 168 hours while the slide step is constant and equal to 12 hours. Each row of diagrams of Figure 9 corresponds to a different amount of delayed events while the diagram columns refer to (a) the total recognition time, (b) the improvement $RTEC_{inc}$ brings in the processing time of each fluent type and (c) the improvement $RTEC_{inc}$ achieves in the processing time of each interval manipulation construct. The times displayed in the first column of diagrams of Figure 9 correspond to the average recognition time. In the second and third columns of the diagrams of Figure 9, positive values indicate an improvement in performance achieved by $RTEC_{inc}$ while negative values indicate a performance decrease. In all the diagrams of Figure 9 we state for each sliding window the average number of SDEs falling inside. It should be noted that for a different percentage of delayed events, the average number of SDEs may be different for the same window size. For example, when 5% of the SDEs are delayed, the 168-hours window includes $249K$ SDEs on average, while when 10% of the SDEs are delayed, the 168-hours window contains $246K$ SDEs. This is due to the fact that some of the delayed events arrive to the CER system too late to be taken into consideration.

Regarding recognition time, $RTEC_{inc}$ achieves a performance gain for all the percentages of delayed events and window sizes. This performance gain is highlighted for each sliding window with the arrows in each diagram of the first column of Figure 9. The numbers on top of the arrows denote the approximate improvement $RTEC_{inc}$ brings to the overall recognition. The performance gain becomes more profound as the sliding window size increases. Furthermore, $RTEC_{inc}$ has more predictable performance, as indicated by the standard deviations displayed in the left diagrams of Figure 9. $RTEC_{inc}$ improves performance even in the case of the 12-hours window, where there is no overlap in consecutive windows. This is mainly due to the fact that $RTEC_{inc}$ recognizes the cases where interval computation is unnecessary and avoids them.

The second column of Figure 9 shows the improvement of $RTEC_{inc}$ as regards the computation of the intervals of simple and statically determined fluents. In the case of statically determined fluents, RTEC has a better performance than $RTEC_{inc}$, regardless of the amount of delayed SDEs and the window size. The third column of Figure 9 refers to the comparison of the three interval manipulation constructs. RTEC's evaluation of the interval manipulation constructs usually demands less amount of time. In the complexity analysis we showed the conditions under which the incremental version of the three interval manipulation constructs can outperform the non-incremental one. These conditions are usually hard to satisfy, mainly for intersect_all and relative_complement_all. The most decisive factor for the performance of $RTEC_{inc}$ is the number of intervals the body fluents of a rule have from the previous query time. The higher the number of these intervals, the more probable it is for the incremental procedure to improve performance. In the Brest dataset, the SDEs referring to a vessel are infrequent, and as a consequence the produced intervals of a fluent referring to a vessel may not be great in number.

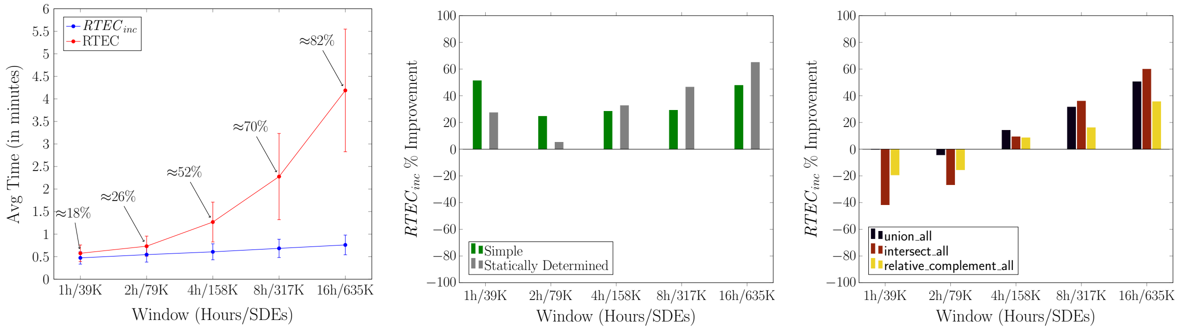
Even in the cases where $RTEC_{inc}$ improves the performance in all interval manipulation constructs, the overall performance concerning statically determined fluents is worse than RTEC. This lies in the fact that $RTEC_{inc}$ operates on binary rules to produce the intervals of a statically determined fluent, leading to costly memory operations. RTEC avoids this issue as it is not restricted to binary rules for statically determined fluents.

$RTEC_{inc}$ manages to produce the intervals of simple fluents more efficiently than RTEC. This is the main reason why $RTEC_{inc}$ achieves overall better performance than RTEC. Our complexity analysis showed that the most important factor is the ratio of delayed events to the size of the overlap. The lower this ratio is the higher the gain of $RTEC_{inc}$. By inspecting the performance of $RTEC_{inc}$ in the same window but for different percentages of delayed events, one may verify that as the number of delayed events increases the improvement in processing time also decreases.

5.2.2 Natural delays

The second dataset concerns $34K$ vessels sailing in the European seas in January 2016 (see Figure 8(b) on page 33). The compression-annotation of the AIS messages and subsequently the spatial processing yielded in total $\approx 17M$ SDEs. As already mentioned, this dataset retains the natural delays and permits us to test the recognition systems in real life conditions. Additionally, it allows us to test the scalability of the systems given the large number of vessels and SDEs.

Figure 10 displays our experimental results. We used five different sliding windows, varying from 1 to 16 hours, and a constant step of 1 hour to examine the effect of the overlap in performance. The number of SDEs increases dramatically even for smaller windows compared to the Brest dataset. Both $RTEC_{inc}$ and RTEC are capable of supporting real-time CER in this demanding dataset. $RTEC_{inc}$ manages again to improve performance in all window sizes and as the overlap of consecutive query times increases, this improvement is more pronounced (Figure 10(a)). Again, the arrows demonstrate the improvement $RTEC_{inc}$



(a) Total recognition time

(b) Fluent improvement

(c) Imc improvement

Figure 10: Results with the natural delays dataset (slide step of 1 hour).

achieves in each sliding window. Additionally, in contrast to RTEC, $RTEC_{inc}$ exhibits a near linear increase in processing time, as the size of the window increases.

In Figure 10(b) we present $RTEC_{inc}$'s improvement in processing time for statically determined and simple fluents, compared to RTEC. In contrast to the experiments performed with the synthetic delays dataset, $RTEC_{inc}$ now manages to improve performance in all windows for statically determined fluents. The great increase in SDEs allows the incremental procedure to achieve the observed performance gain. In Figure 10(c) the incremental operations achieve better performance in 4, 8 and 16-hours windows, illustrating that as the overlap increases the effects of the incremental process on performance are more pronounced.

Regarding the processing of simple fluents, $RTEC_{inc}$ again significantly enhances the performance of RTEC. As the overlap of consecutive query times increases, the performance gain of $RTEC_{inc}$ over RTEC increases. This result validates further our complexity analysis and highlights the importance of the ratio of delayed events to overlapping events.

5.3 Fleet Management

The European economy relies to a great extent on commercial vehicle fleets. According to the European Automobile Manufacturers Association[9], there were over 54 Million commercial vehicles in use in Europe in 2015, and this number is growing every year. Commercial vehicles are equipped with devices emitting information regarding their location and operational status, such as speed and fuel level. Fleet management applications collect the information emitted from moving vehicles in order to improve the management and planning of transportation services, and enable informed decision-making. Detecting CEs from such data streams can be beneficial for the drivers of commercial vehicles, since they can be informed about their performance, and even prevent dangerous situations. Additionally, the analysis of data generated by such a fleet of vehicles can help the owners maximize the performance of the fleet.

[9] http://www.acea.be/statistics/article/vehicles-in-use-europe-2017

In the context of the Track & Know[10] project, we developed an online fleet management system for the recognition of CEs, that improves the operating efficiency of a commercial fleet. Our system utilises the GPS (Global Positioning System) traces of moving vehicles along with information emitted by installed accelerometer devices, such as abrupt acceleration, and information concerning the level of fuel in a vehicle's tank provided by a fuel sensor. These traces are enriched with weather and Point-of-Interest (POI) information by a dedicated component for data enrichment (Tsilionis et al., 2019b). The enriched data stream is transmitted to the CER module, in order to recognize various types of vehicle activity. All such activities have been formalized in collaboration with the domain experts of the Track & Know project.

The event description includes four simple fluents. The rule-set below presents a simple fluent formalisation that detects opportunities for refueling:

$$\begin{aligned}
&\mathsf{initiatedAt}(\mathit{reFuelOpportunity}(V){=}\mathsf{true},\ T) \leftarrow\\
&\quad \mathsf{happensAt}(\mathit{closeToGas}(V),\ T),\\
&\quad \mathsf{holdsAt}(\mathit{highSpeed}(V){=}\mathsf{true},\ T),\\
&\quad \mathsf{happensAt}(\mathit{fuelLevel}(V, L),\ T),\\
&\quad \mathit{threshold}(V, \mathit{fuel}, V_{\mathit{tank}}),\ \ L < V_{\mathit{tank}}/2.\\
&\mathsf{terminatedAt}(\mathit{reFuelOpportunity}(V){=}\mathsf{true},\ T) \leftarrow\\
&\quad \mathsf{happensAt}(\mathit{fuelLevel}(V, L),\ T),\\
&\quad \mathit{threshold}(V, \mathit{fuel}, V_{\mathit{tank}}),\ \ L \geq V_{\mathit{tank}}/2.
\end{aligned} \tag{27}$$

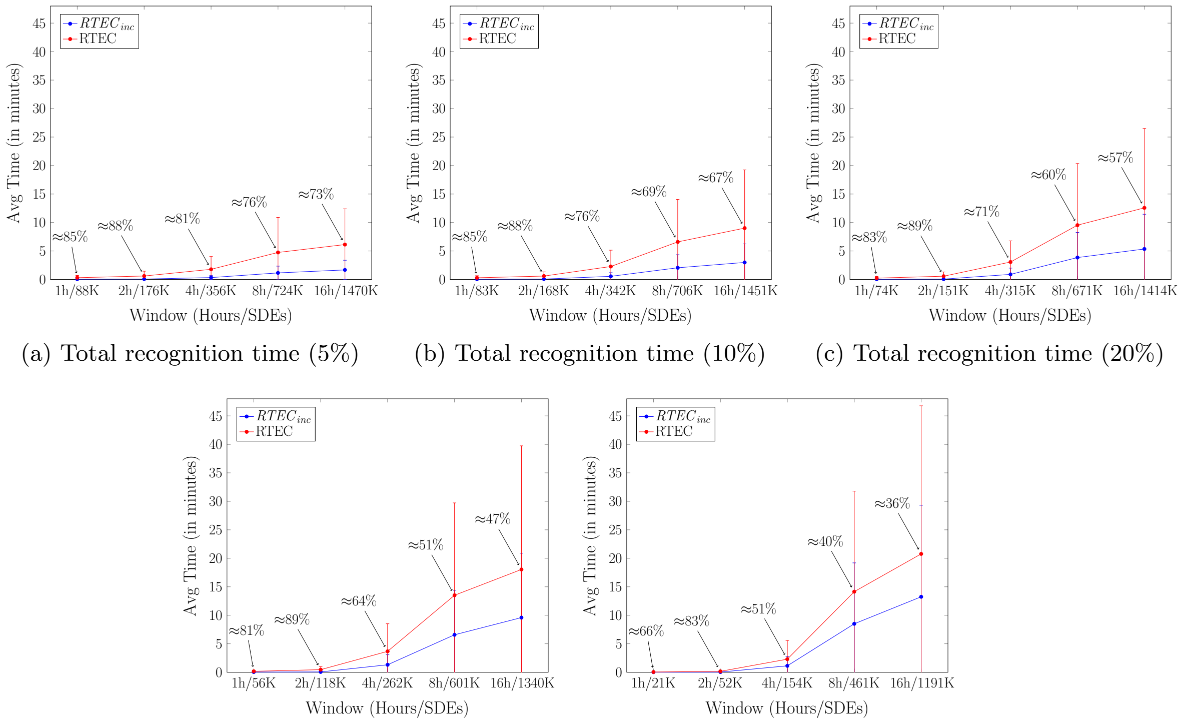
Refueling opportunities can be helpful for companies owning commercial fleets, since they place emphasis on fuel consumption. $\mathit{closeToGas}(V)$ is a spatial relation computed by the data enrichment module, indicating that a vehicle V is close to a gas station. $\mathit{highSpeed}(V)$ denotes that a vehicle V is moving with a speed greater than a user-specified threshold. $\mathit{fuelLevel}(V, L)$ is an SDE emitted by the fuel sensor of each vehicle. $\mathit{threshold}(V, \mathit{fuel}, V_{\mathit{tank}})$ records the tank size of vehicles. According to rule-set (27), our system starts flagging that vehicle V should refuel when it is close to a gas station, its speed is above the user-specified threshold (implying uneconomic driving), and the fuel level is lower than half of the tank size. Moreover, we stop flagging the need to refuel when the fuel is more than half of the tank size.

We evaluated $RTEC_{inc}$ on real-world positional data of vehicles, provided by Vodafone Innovus[11], our partner in the Track & Know project, which offers fleet management services. The dataset concerns approx. $6K$ vehicles traveling in the European road network and spans from 30 June 2018 to 8 August 2018. Each record of the data is enriched with weather data and proximity to POIs, leading to approx. $70M$ SDEs. This large number of SDEs allows us to test further the scalability of the CER engines.

Since the original dataset is temporally sorted, a strategy of artificially injecting delays was followed as that described in Section 5.2.1. We performed five experiments, each time varying the amount of SDEs being delayed (5%, 10%, 20%, 40% and 80%). The average delay time, in all settings, is approximately 8 hours.

[10] `https://trackandknowproject.eu/`

[11] `https://www.vodafoneinnovus.com/`



(a) Total recognition time (5%)

(b) Total recognition time (10%)

(c) Total recognition time (20%)

(d) Total recognition time (40%)

(e) Total recognition time (80%)

Figure 11: Results on the fleet management dataset (slide step of 1 hour). Each diagram corresponds to a different amount of delayed SDEs (5%, 10%, 20%, 40%, 80%).

Figure 11 presents the results regarding recognition time, where the sliding window varies from 1 to 16 hours and the slide step is constant and equal to 1 hour. Each diagram corresponds to a different amount of delayed events. $RTEC_{inc}$ outperforms RTEC in all experiments. The performance improvement becomes more profound as the overlap between consecutive windows increases. Both RTEC and $RTEC_{inc}$ exhibit variations in their performance; as the number of delayed events increases the intervals of a fluent are more susceptible to changes. Moreover, as the percentage of delayed events decreases it is more probable for $RTEC_{inc}$ to achieve performance gain.

6. Related Work

The goal of CER systems is to identify CEs of interest given as input a stream of time-stamped SDEs, such as events coming from sensors of vessels or vehicles. In the literature, numerous CER systems and languages have been proposed. See (Cugola & Margara, 2012; Alevizos et al., 2017; Giatrakos et al., 2020) for three surveys. These systems differ in their architectures, data models, pattern languages and processing mechanisms (Grez et al., 2019, 2020). For example, many CER methods are based on automata (Demers et al., 2007; Zhang et al., 2014; Schultz-Møller et al., 2009; Apache FlinkCEP[12]). The automaton model is used

[12] https://ci.apache.org/projects/flink/flink-docs-stable/dev/libs/cep.html

to describe the semantics of the language as well as the recognition algorithm. Apart from automata, some CER systems employ tree-based models (Liu et al., 2011; Mei & Madden, 2009). Again, tree-based formalisms are used for both modeling and pattern matching, i.e. they may describe the event patterns and the applied recognition algorithm.

Logic-based approaches to CER have also been attracting considerable attention, since they exhibit a formal, declarative semantics, and at the same time support efficient reasoning (Fern et al., 2002; Dousson & Maigat, 2007; Cugola & Margara, 2010; Paschke & Bichler, 2008; Brandt et al., 2018). RTEC is a logic programming implementation of the Event Calculus (Kowalski & Sergot, 1986), that has been used for CER in various application domains, such as maritime situational awareness (Patroumpas et al., 2017; Pitsikalis et al., 2019) and fleet management (Tsilionis et al., 2019b). The use of the Event Calculus facilitates considerably the development of succinct CE definitions by taking advantage of the built-in representation of inertia, favors the interaction between CE developer and domain expert and supports code maintenance. RTEC explicitly represents CE intervals (unlike e.g. Dousson & Maigat, 2007; Cugola & Margara, 2010; Beck et al., 2018) and thus avoids the related logical problems (Paschke, 2006). Furthermore, RTEC supports not only complex temporal constraints, but also complex atemporal constraints, as well as reasoning over background knowledge. This is in contrast to other CER systems proposed in the literature (Arasu et al., 2006; Dousson & Maigat, 2007; Cugola & Margara, 2010; Anicic et al., 2012). Moreover, and in contrast to state-of-the-art recognition systems, such as the Esper[13] engine and SASE[14] (Zhang et al., 2014), RTEC can naturally express hierarchical knowledge by means of well-structured specifications, and consequently employ caching techniques to avoid unnecessary re-computations.

Concerning the Event Calculus literature, a key feature of RTEC is that it includes a windowing technique. No other Event Calculus system (including Chittaro & Montanari, 1996; Cervesato & Montanari, 2000; Miller & Shanahan, 2002; Paschke & Bichler, 2008; Lee & Palla, 2012; Montali et al., 2013) 'forgets' or represents concisely the data stream history. Another feature of RTEC is the ability to deal with out-of-order and/or retracted SDE streams through its windowing mechanism. The processing of events under the assumption that the stream is temporally sorted is a serious limitation of several other CER systems (Gyllstrom et al., 2006; Cugola & Margara, 2010, 2012; Dindar et al., 2011; Li et al., 2011; Beck et al., 2018), since they cannot update or recognize new CEs as a result of delayed arrival or retraction of SDEs. One limitation of RTEC is the computation from scratch strategy, which results to inefficiencies. In this work, we presented an incremental version of RTEC, i.e. $RTEC_{inc}$, that overcomes this issue.

The techniques presented for incremental reasoning are based on the incremental maintenance of materialised views in deductive databases. A materialised view in a deductive database is the process of computing the effects of explicit facts on rules of a program and storing the result (Gupta et al., 1993; Motik et al., 2019). Explicit facts along with the implicit facts derived from the rules form the knowledge base of the database. Explicit facts can be seen as the SDEs that are the input in a CER system. The explicit facts can be updated through additions or deletions and thus, a new materialisation has to be produced. However, the update of explicit facts may or may not affect the inferences produced after

[13] http://www.espertech.com/esper/

[14] http://sase.cs.umass.edu//

rule evaluation. The goal of incremental maintenance is to compute the new materialisation, using the updates and the state of the database before the updates. The result is the minimization of the overall cost. In order for this operation to be efficient, incremental maintenance has to detect and deal only with those rule instances that require update. Notice that incremental materialisation is not always preferable over materialisation from scratch. Consider, for example, the case in which large amounts of explicit facts are inserted and deleted from the database (Motik et al., 2019; Gupta et al., 1993).

An update may cause a rule that fired before the update to stop firing or vice versa. The changes in the inferences induced by the updates have to be applied to the old materialization, in order to produce the new one. Moreover, the changes introduced by the updates are expressed as rules, i.e. rewritings of the original rules of the program capturing the difference between consecutive materialisations. In the literature, they are called *delta* rules and in Section 3, based on an initiatedAt rule, we provide the *delta* rules that compute the changes that have to be made in the initiation points of a fluent. Different incremental maintenance techniques have been proposed for the evaluation of the *delta* rules.

Ceri and Widom (1991) have presented a technique for maintaining views in relational databases with the use of production rules (similar to the *delta* rules). The rewriting technique has also been used to show that a recursive relation can be expressed as a non-recursive first-order query with respect to insertions (Dong & Topor, 1992; Dong & Su, 1994; Dong et al., 1995). Dong and Su (1995) discuss non-recursive methods to update the transitive closure of various graphs with respect to deletions and insertions. However, none of these approaches handles negation.

Another approach towards incremental maintenance is the counting algorithm (Gupta et al., 1993; Motik et al., 2019). In counting, every fact (explicit or implicit) is associated with a counter that denotes the number of times it has been derived. The counter is stored in the materialised view. When an update takes place, the counting algorithm uses the changes made to the explicit facts, as well as the previous state of the materialisation, in order to compute the changes that have to be made in the inferences. These changes are reflected in the counter of each fact. When a counter becomes zero the fact must be deleted from the database. On the other hand, when the counter is positive, the fact must remain or be inserted in the view. The limitations of the counting algorithm are the inability to handle recursive rules and the potential memory overhead caused by storing the counters (Hu et al., 2018; Motik et al., 2019).

The Delete/Rederive (DRed) (Gupta et al., 1993; Motik et al., 2019) algorithm is a different approach that deals with recursion and does not depend on additional information, in order to decide if a fact has to be removed from or be inserted in the view. DRed first 'over-deletes' all implicit facts that depend on updated facts, that is, it evaluates all rules whose body contains an updated fact. This is called the over-deletion step. Then, it adopts a re-derivation step, during which the algorithm tries to find alternative derivations for each deleted fact. The body of each rule, whose head can be matched to the deleted fact, is evaluated over the new materialisation and if the evaluation succeeds the fact is inserted again to the view. The re-derivation step thus introduces a form of redundancy. The counting algorithm avoids this inefficiency since the counter's value determines the presence of a fact in the view (Hu et al., 2018; Motik et al., 2019).

In approaches where duplicate semantics (i.e. storing a derived fact as many times as it is inferred) is not desired, as in the case of RTEC, the counting algorithm, in combination with stratum by stratum evaluation, is preferred over DRed (Motik et al., 2019). The counter of a predicate denotes how many times the fact is derived within a stratum. Predicates of higher strata that depend on it, do not need to be evaluated as many times as the value of the counter, but only if the counter has value 1 or 0, that is, if it exists or not in the view. In this manner duplicate elimination, which is an expensive operation, can be achieved at little or no extra cost (Gupta et al., 1993). Finally, counting and DRed can handle negation in the body literals of a rule (Motik et al., 2019). Negation in rule bodies is very important in CER since the absence of an event may lead to the firing of the rule. Additionally, during updates the delayed arrival of an event may lead to the deletion of a rule derivation.

Motik et al. (2015) presented the Backward/Forward (BF) algorithm, which in certain cases overcomes the redundancy introduced during the re-derivation step of DRed. If a fact is considered for deletion, BF searches for alternative derivations and only if one cannot be found it proceeds with the deletion. Hu et al. (2018) proposed two hybrid approaches, combining DRed and BF with counting. DRed with counting associates with each fact two counters, one for the non-recursive and one for the recursive rules. A fact is never over-deleted if its non-recursive counter is positive, while if this counter equals to zero the recursive counter must also equals to zero in order for the fact to be removed from the view. If the recursive counter differs from zero the fact remains in the view. Thus, the re-derivation step is replaced with a simple check at the recursive counter. In order to reduce the amount of time spent in discovering alternative derivations of a fact, BF with counting associates with each fact a non-recursive counter. Only if the non-recursive counter equals to zero it evaluates recursive rules to examine if the fact still holds. In the presence of a lot of recursive rules BF with counting is the preferred choice since it finds alternative derivations and the cost of over-deletion is reduced. However, when the number of recursive rules is small, over-deletion favors computation and thus, DRed with counting is the best choice. Notice that in the absence of recursive rules DRed with counting is identical to the original counting algorithm. An improvement of DRed with counting, regarding recursive rules, can be found in Hu et al. (2019).

All the approaches presented above refer to databases which are usually static and do not undergo great changes. Our work transfers these ideas to a streaming environment, which is dynamic in nature and the underlying facts, i.e. SDEs, are constantly changing. Query answering in streaming environments needs to be fast and make use of the available limited resources. Stream management systems achieve this by employing window mechanisms that restrict the analysis in a finite part of the stream. Our method, $RTEC_{inc}$, takes advantage of the window mechanism of RTEC, as well as its ability to handle temporal reasoning, and applies incremental maintenance techniques in order to deal with out-of-order events. Additionally, the logic programs used in our experimental evaluation contain many rules and thus $RTEC_{inc}$ supports multi-query answering. The ideas of DRed and rewriting of rules have been implemented in a stream reasoning framework by Barbieri et al. (2010). These authors treat insertions with the use of incremental maintenance rules. However, they do not support negation in their rules, they consider the stream temporally sorted and their logic program consists only of one rule. Another interesting stream reasoning approach that is based on Temporal Datalog is proposed by Ronca et al. (2018). In this case, a window construct is presented, that forgets old and accepts new input only if the stream is temporally sorted. Furthermore, negation is not supported.

DBToaster is a method that performs Higher-Order incremental view maintenance, meaning that it does not store only the result of the query but also stores the result of subqueries (Koch et al., 2014). This way it achieves considerable speed-ups in updating a materialisation compared to the methods outlined above but in the cost of increased memory consumption. In streaming environments where there are limited resources and near real-time analysis is required, this is prohibitive. Idris et al. (2019) developed a method that maintains an indexing structure that can produce through constant delay enumeration the database materialisation without the need to store the answer of the query and the results of its subqueries. Furthermore, depending on the type of the query it achieves fast updating times of the materialised view. This approach is very efficient but the experimental evaluation always concerns the processing of a single query. Therefore, it is not clear how it can address large (hierarchical) knowledge bases.

A work that is closely related to ours, in the sense that it concerns the Event Calculus and the minimization of redundant interval computations, is the Cached Event Calculus (CEC) (Chittaro & Montanari, 1996). A key difference between our approach and CEC is that we propagate changes stratum by stratum, as opposed to sequentially propagating the effects of individual facts. Therefore, $RTEC_{inc}$ could scale to the datasets presented in the previous section, while CEC could not.

The abundance of sensors that transmit data in high rates in the recent years has also led to the development of distributed event-based systems. Mutschler and Philippsen (2014) propose a hybrid method that combines buffering and speculative techniques to deal with out-of-order event arrival in distributed environments. This method concerns a middleware that is responsible for sending the input to the event detectors and is blind to the recognition algorithm used by event detectors. A distributed materialisation of Datalog rules is presented in (Ajileye et al., 2019) where dynamic data exchange reduces the network communication. This is an interesting approach that does not consider, however, negation and incremental materialisation.

7. Summary and Future Work

We presented $RTEC_{inc}$, an incremental version of the Event Calculus for Run-Time reasoning. $RTEC_{inc}$ is a formal computational framework for incremental event recognition which deals efficiently with the delayed arrival and retraction of SDEs. Using techniques reminiscent of the incremental maintenance of deductive databases, $RTEC_{inc}$ avoids unnecessary calculations and improves performance. Our empirical evaluation on big synthetic and real datasets from the fields of maritime situational awareness and fleet management demonstrated the effectiveness of the incremental procedure. Our intention is to compare experimentally the two methods in other application domains where the granularity of time is completely different. Furthermore, we intend to compare our incremental procedure with other state-of-the-art frameworks. Finally, besides devising optimisations that can boost performance, such as distributed incremental reasoning, we would like to extend our method in order to handle recursive definitions and events with delayed effects, which are necessary in various applications (Pitsikalis et al., 2019).

Acknowledgments

This work was funded partly by the EU H2020 Track & Know "Big Data for Mobility Tracking Knowledge Extraction in Urban Areas" and partly by the EU H2020 INFORE "Interactive Extreme-Scale Analytics and Forecasting" projects.

Appendix

We present the built-in axioms of RTEC ensuring that a fluent F may have at most one value at each time-point, the derivations of the incremental evaluation formulas concerning statically determined fluents, as well the complete tables expressing the comparison of RTEC and $RTEC_{inc}$ regarding the three interval manipulation constructs.

Built-in Axioms Concerning Fluent-Value Pairs

Consider the axioms below:

$$\begin{aligned}&\mathsf{broken}(F{=}V,\ T_s,\ T) \leftarrow\\ &\quad \mathsf{terminatedAt}(F{=}V,\ T_f), T_s < T_f \leq T.\end{aligned} \tag{a}$$

$$\begin{aligned}&\mathsf{broken}(F{=}V_1,\ T_s,\ T) \leftarrow\\ &\quad \mathsf{initiatedAt}(F{=}V_2,\ T_f), T_s < T_f \leq T,\ V_1 \neq V_2.\end{aligned} \tag{b}$$

$\mathsf{broken}(F{=}V,\ T_s,\ T)$ represents that a maximal interval starting at T_s for which $F{=}V$ holds continuously is terminated at some time T_f such that $T_s < T_f \leq T$. According to rule (a), $F{=}V$ is 'broken' if it is terminated. According to rule (b), if $F{=}V_2$ is initiated at T_f then effectively $F{=}V_1$ is terminated at time T_f, for all other possible values V_1 of F. Rule (b) ensures therefore that a fluent cannot have more than one value at any time. We do not insist that a fluent must have a value at every time-point. There is a difference between initiating a Boolean fluent $F{=}\mathsf{false}$ and terminating $F{=}\mathsf{true}$: the former implies, but is not implied by the latter.

Proofs of the Incremental Evaluation of Interval Manipulation Constructs

We present the derivations of the incremental evaluation formulas of $RTEC_{inc}$ concerning the interval manipulation constructs. Table 16 displays a list of properties (Bassiouni et al., 1993) concerning sets of intervals, that are used in the derivations (property numbers are displayed below the horizontal curly brackets in the derivations).

Table 16: Properties of temporal sets. Capital letters represent sets of time intervals.

♯	**Property**
1	$(a)\quad A \cup_T B = B \cup_T A$ $(b)\quad A \cap_T B = B \cap_T A$

2	$(a) \quad A \cup_T (B \cap_T C) = (A \cup_T B) \cap_T (A \cup_T C)$ $(b) \quad A \cap_T (B \cup_T C) = (A \cap_T B) \cup_T (A \cap_T C)$
3	$(a) \quad A \cup_T (A \cap_T B) = A$ $(b) \quad A \cap_T (A \cup_T B) = A$
4	$A \cap_T B = A \setminus_T (A \setminus_T B)$
5	$A \setminus_T (B \cup_T C) = (A \cup_T C) \setminus_T (B \cup_T C) =$ $(A \setminus_T B) \setminus_T C = (A \setminus_T C) \setminus_T B$
6	$(a) \quad (A \cup_T B) \setminus_T C = (A \setminus_T C) \cup_T (B \setminus_T C)$ $(b) \quad (A \cap_T B) \setminus_T C = (A \setminus_T C) \cap_T (B \setminus_T C)$
7	$A \setminus_T (B \setminus_T C) = (A \cap_T C) \cup_T (A \setminus_T B)$
8	$(A \setminus_T B) \cup_T C = (A \cup_T C) \setminus_T (B \setminus_T C)$
9	$(A \setminus_T B) \cap_T C = (A \cap_T C) \setminus_T B = (A \cap_T C) \setminus_T (B \cap_T C)$
10	$(a) \quad A \setminus_T (B \cap_T C) = (A \setminus_T B) \cup_T (A \setminus_T C)$ $(b) \quad A \setminus_T (B \cup_T C) = (A \setminus_T B) \cap_T (A \setminus_T C)$
11	$(a) \quad (A \setminus_T B) \cup_T A = A, \quad (b) \quad (A \setminus_T B) \cup_T B = A \cup_T B$ $(c) \quad (A \setminus_T B) \cap_T A = A \setminus_T B, \quad (d) \quad (A \setminus_T B) \cap_T B = \varnothing$

Derivation of Eq. (15) expressing the incremental evaluation of union_all

$$I_A^{\mathsf{Q_i}} \cup_T I_B^{\mathsf{Q_i}} =$$

$$\underbrace{\left[(I_A^{\mathsf{Q_{i-1}}} \setminus_T I_A^-) \cup_T I_A^+\right]}_{\texttt{Eq. (12)}} \cup_T \underbrace{\left[(I_B^{\mathsf{Q_{i-1}}} \setminus_T I_B^-) \cup_T I_B^+\right]}_{\texttt{Eq. (12)}} =$$

$$\underbrace{(I_B^{\mathsf{Q_{i-1}}} \setminus_T I_B^-) \cup_T (I_A^{\mathsf{Q_{i-1}}} \setminus_T I_A^-) \cup_T (I_A^+ \cup_T I_B^+)}_{\texttt{Pr.1a}} =$$

$$(I_B^{\mathsf{Q_{i-1}}} \setminus_T I_B^-) \cup_T \left[\underbrace{\left[I_A^{\mathsf{Q_{i-1}}} \cap_T (I_A^{\mathsf{Q_{i-1}}} \cup_T I_B^{\mathsf{Q_{i-1}}})\right]}_{\texttt{Pr.3b}} \setminus_T I_A^-\right] \cup_T (I_A^+ \cup_T I_B^+) =$$

$$(I_B^{\mathsf{Q_{i-1}}} \setminus_T I_B^-) \cup_T \underbrace{\left[I_A^{\mathsf{Q_{i-1}}} \cap_T \left[(I_A^{\mathsf{Q_{i-1}}} \cup_T I_B^{\mathsf{Q_{i-1}}}) \setminus_T I_A^-\right]\right]}_{\texttt{Pr.9}} \cup_T (I_A^+ \cup_T I_B^+) =$$

$$\underbrace{\Bigg[\Big[I_A^{\mathsf{Q}_{\mathsf{i}-1}} \cup_T (I_B^{\mathsf{Q}_{\mathsf{i}-1}} \setminus_T I_B^-)\Big] \cap_T \Big[(I_B^{\mathsf{Q}_{\mathsf{i}-1}} \setminus_T I_B^-) \cup_T \Big[(I_A^{\mathsf{Q}_{\mathsf{i}-1}} \cup_T I_B^{\mathsf{Q}_{\mathsf{i}-1}}) \setminus_T I_A^-\Big]\Big]\Bigg]}_{\texttt{Pr.2a}} \cup_T (I_A^+ \cup_T I_B^+) =$$

$$\Bigg[\Big[\underbrace{I_A^{\mathsf{Q}_{\mathsf{i}-1}} \cup_T (I_A^{\mathsf{Q}_{\mathsf{i}-1}} \setminus_T I_B^-)}_{\texttt{Pr.11a}} \cup_T (I_B^{\mathsf{Q}_{\mathsf{i}-1}} \setminus_T I_B^-)\Big] \cap_T \Big[\Big[\underbrace{[(I_A^{\mathsf{Q}_{\mathsf{i}-1}} \cup_T I_B^{\mathsf{Q}_{\mathsf{i}-1}}) \cap_T I_B^{\mathsf{Q}_{\mathsf{i}-1}}]}_{\texttt{Pr.3b}} \setminus_T I_B^-\Big] \cup_T \Big[(I_A^{\mathsf{Q}_{\mathsf{i}-1}} \cup_T I_B^{\mathsf{Q}_{\mathsf{i}-1}}) \setminus_T I_A^-\Big]\Big]\Bigg] \cup_T (I_A^+ \cup_T I_B^+) =$$

$$\Bigg[\Big[I_A^{\mathsf{Q}_{\mathsf{i}-1}} \cup_T \underbrace{\Big[(I_A^{\mathsf{Q}_{\mathsf{i}-1}} \cup_T I_B^{\mathsf{Q}_{\mathsf{i}-1}}) \setminus_T I_B^-\Big]}_{\texttt{Pr.6a}}\Big] \cap_T \Big[\underbrace{\Big[(I_A^{\mathsf{Q}_{\mathsf{i}-1}} \cup_T I_B^{\mathsf{Q}_{\mathsf{i}-1}}) \cap_T (I_B^{\mathsf{Q}_{\mathsf{i}-1}} \setminus_T I_B^-)\Big]}_{\texttt{Pr.9}} \cup_T \Big[(I_A^{\mathsf{Q}_{\mathsf{i}-1}} \cup_T I_B^{\mathsf{Q}_{\mathsf{i}-1}}) \setminus_T I_A^-\Big]\Big]\Bigg] \cup_T (I_A^+ \cup_T I_B^+) =$$

$$\Bigg[\Big[\underbrace{\Big[(I_A^{\mathsf{Q}_{\mathsf{i}-1}} \cup_T I_B^{\mathsf{Q}_{\mathsf{i}-1}}) \cap_T I_A^{\mathsf{Q}_{\mathsf{i}-1}}\Big]}_{\texttt{Pr.3b}} \cup_T \Big[(I_A^{\mathsf{Q}_{\mathsf{i}-1}} \cup_T I_B^{\mathsf{Q}_{\mathsf{i}-1}}) \setminus_T I_B^-\Big]\Big] \cap_T \underbrace{\Big[(I_A^{\mathsf{Q}_{\mathsf{i}-1}} \cup_T I_B^{\mathsf{Q}_{\mathsf{i}-1}}) \setminus_T \Big[I_A^- \setminus_T (I_B^{\mathsf{Q}_{\mathsf{i}-1}} \setminus_T I_B^-)\Big]\Big]}_{\texttt{Pr.7}}\Bigg] \cup_T (I_A^+ \cup_T I_B^+) =$$

$$\Bigg[\underbrace{\Big[(I_A^{\mathsf{Q}_{\mathsf{i}-1}} \cup_T I_B^{\mathsf{Q}_{\mathsf{i}-1}}) \setminus_T (I_B^- \setminus_T I_A^{\mathsf{Q}_{\mathsf{i}-1}})\Big]}_{\texttt{Pr.7}} \cap_T \Big[(I_A^{\mathsf{Q}_{\mathsf{i}-1}} \cup_T I_B^{\mathsf{Q}_{\mathsf{i}-1}}) \setminus_T \Big[I_A^- \setminus_T (I_B^{\mathsf{Q}_{\mathsf{i}-1}} \setminus_T I_B^-)\Big]\Big]\Bigg] \cup_T (I_A^+ \cup_T I_B^+) =$$

$$\underbrace{\Bigg[(I_A^{\mathsf{Q}_{\mathsf{i}-1}} \cup_T I_B^{\mathsf{Q}_{\mathsf{i}-1}}) \setminus_T \Big[(I_B^- \setminus_T I_A^{\mathsf{Q}_{\mathsf{i}-1}}) \cup_T \Big[I_A^- \setminus_T (I_B^{\mathsf{Q}_{\mathsf{i}-1}} \setminus_T I_B^-)\Big]\Big]\Bigg]}_{\texttt{Pr.10b}} \cup_T (I_A^+ \cup_T I_B^+) =$$

$$\Bigg[(I_A^{\mathsf{Q}_{\mathsf{i}-1}} \cup_T I_B^{\mathsf{Q}_{\mathsf{i}-1}}) \setminus_T \Big[(I_B^- \setminus_T I_A^{\mathsf{Q}_{\mathsf{i}-1}}) \cup_T \Big[\underbrace{[I_A^- \cap_T (I_A^- \cup_T I_B^-)]}_{\texttt{Pr.3b}} \setminus_T (I_B^{\mathsf{Q}_{\mathsf{i}-1}} \setminus_T I_B^-)\Big]\Big]\Bigg] \cup_T (I_A^+ \cup_T I_B^+) =$$

$$\Bigg[(I_A^{\mathsf{Q}_{\mathsf{i}-1}} \cup_T I_B^{\mathsf{Q}_{\mathsf{i}-1}}) \setminus_T \Big[(I_B^- \setminus_T I_A^{\mathsf{Q}_{\mathsf{i}-1}}) \cup_T \underbrace{\Big[I_A^- \cap_T [(I_A^- \cup_T I_B^-) \setminus_T (I_B^{\mathsf{Q}_{\mathsf{i}-1}} \setminus_T I_B^-)]\Big]}_{\texttt{Pr.9}}\Big]\Bigg] \cup_T (I_A^+ \cup_T I_B^+) =$$

$$\Bigg[(I_A^{\mathsf{Q}_{\mathsf{i}-1}} \cup_T I_B^{\mathsf{Q}_{\mathsf{i}-1}}) \setminus_T \Big[(I_B^- \setminus_T I_A^{\mathsf{Q}_{\mathsf{i}-1}}) \cup_T \underbrace{\Big[(I_A^- \setminus_T I_B^{\mathsf{Q}_{\mathsf{i}-1}}) \cup_T I_A^-\Big]}_{\texttt{Pr.11a}} \cap_T \underbrace{\Big[(I_A^- \setminus_T I_B^{\mathsf{Q}_{\mathsf{i}-1}}) \cup_T I_B^-\Big]}_{\texttt{Pr.8}}\Big]\Bigg] \cup_T (I_A^+ \cup_T I_B^+) =$$

$$\Bigg[(I_A^{\mathsf{Q}_{\mathsf{i}-1}} \cup_T I_B^{\mathsf{Q}_{\mathsf{i}-1}}) \setminus_T \Big[(I_B^- \setminus_T I_A^{\mathsf{Q}_{\mathsf{i}-1}}) \cup_T \underbrace{(I_A^- \setminus_T I_B^{\mathsf{Q}_{\mathsf{i}-1}}) \cup_T (I_A^- \cap_T I_B^-)}_{\texttt{Pr.2a}}\Big]\Bigg] \cup_T (I_A^+ \cup_T I_B^+) =$$

$$\Bigg[(I_A^{\mathsf{Q}_{\mathsf{i}-1}} \cup_T I_B^{\mathsf{Q}_{\mathsf{i}-1}}) \setminus_T \Big[\underbrace{\Big[(I_A^- \setminus_T I_A^{\mathsf{Q}_{\mathsf{i}-1}}) \cup_T (I_B^- \setminus_T I_A^{\mathsf{Q}_{\mathsf{i}-1}}) \cup_T (I_A^- \setminus_T I_B^{\mathsf{Q}_{\mathsf{i}-1}}) \cup_T (I_B^- \setminus_T I_B^{\mathsf{Q}_{\mathsf{i}-1}})\Big]}_{I_F^{\mathsf{Q}_{\mathsf{i}-1}} \cap_T I_F^- = I_F^- \Leftrightarrow I_F^- \setminus_T I_F^{\mathsf{Q}_{\mathsf{i}-1}} = \varnothing} \cup_T (I_A^- \cap_T I_B^-)\Big]\Bigg] \cup_T (I_A^+ \cup_T I_B^+) =$$

$$\Bigg[(I_A^{\mathsf{Q}_{\mathsf{i}-1}} \cup_T I_B^{\mathsf{Q}_{\mathsf{i}-1}}) \setminus_T \Big[\underbrace{\Big[[(I_A^- \cup_T I_B^-) \setminus_T I_A^{\mathsf{Q}_{\mathsf{i}-1}}] \cup_T [(I_A^- \cup_T I_B^-) \setminus_T I_B^{\mathsf{Q}_{\mathsf{i}-1}}]\Big]}_{\texttt{Pr.6a}} \cup_T (I_A^- \cap_T I_B^-)\Big]\Bigg] \cup_T (I_A^+ \cup_T I_B^+) =$$

$$\Bigg[(I_A^{\mathsf{Q}_{\mathsf{i}-1}} \cup_T I_B^{\mathsf{Q}_{\mathsf{i}-1}}) \setminus_T \Big[\underbrace{\Big[(I_A^- \cup_T I_B^-) \setminus_T (I_A^{\mathsf{Q}_{\mathsf{i}-1}} \cap_T I_B^{\mathsf{Q}_{\mathsf{i}-1}})\Big]}_{\texttt{Pr.10a}} \cup_T (I_A^- \cap_T I_B^-)\Big]\Bigg] \cup_T (I_A^+ \cup_T I_B^+)$$

Derivation of Eq. (19) expressing the incremental evaluation of intersect_all

$$I_A^{\mathsf{Q}_\mathsf{i}} \cap_T I_B^{\mathsf{Q}_\mathsf{i}} =$$

$$\underbrace{\Big[(I_A^{\mathsf{Q}_\mathsf{i}-1} \setminus_T I_A^-) \cup_T I_A^+\Big]}_{\texttt{Eq. (12)}} \cap_T \underbrace{\Big[(I_B^{\mathsf{Q}_\mathsf{i}-1} \setminus_T I_B^-) \cup_T I_B^+\Big]}_{\texttt{Eq. (12)}} =$$

$$\underbrace{\Big[(I_A^{\mathsf{Q}_\mathsf{i}-1} \setminus_T I_A^-) \cup_T I_A^+\Big] \cap_T \Big[(I_B^{\mathsf{Q}_\mathsf{i}-1} \setminus_T I_B^-) \cup_T (I_A^{\mathsf{Q}_\mathsf{i}-1} \setminus_T I_A^-) \cup_T I_A^+\Big]}_{\texttt{Pr.3b}} \cap_T \underbrace{\Big[(I_B^{\mathsf{Q}_\mathsf{i}-1} \setminus_T I_B^-) \cup_T I_B^+)\Big] \cap_T \Big[(I_B^{\mathsf{Q}_\mathsf{i}-1} \setminus_T I_B^-) \cup_T (I_A^{\mathsf{Q}_\mathsf{i}-1} \setminus_T I_A^-) \cup_T I_B^+)\Big]}_{\texttt{Pr.3b}} =$$

$$\underbrace{\Big[(I_A^{\mathsf{Q}_\mathsf{i}-1} \setminus_T I_A^-) \cup_T I_A^+\Big] \cap_T \Big[(I_B^{\mathsf{Q}_\mathsf{i}-1} \setminus_T I_B^-) \cup_T I_B^+)\Big] \cap_T \Big[(I_B^{\mathsf{Q}_\mathsf{i}-1} \setminus_T I_B^-) \cup_T (I_A^{\mathsf{Q}_\mathsf{i}-1} \setminus_T I_A^-) \cup_T I_A^+\Big] \cap_T \Big[(I_B^{\mathsf{Q}_\mathsf{i}-1} \setminus_T I_B^-) \cup_T (I_A^{\mathsf{Q}_\mathsf{i}-1} \setminus_T I_A^-) \cup_T I_B^+)\Big]}_{\texttt{Pr.1b}} =$$

$$\Big[\Big[(I_A^{\mathsf{Q}_\mathsf{i}-1} \setminus_T I_A^-) \cup_T \underbrace{I_A^+ \cup_T (I_A^+ \cap_T I_B^+)}_{\texttt{Pr.3a}}\Big] \cap_T \Big[(I_B^{\mathsf{Q}_\mathsf{i}-1} \setminus_T I_B^-) \cup_T \underbrace{I_B^+ \cup_T (I_A^+ \cap_T I_B^+)}_{\texttt{Pr.3a}}\Big] \cap_T \underbrace{\Big[(I_B^{\mathsf{Q}_\mathsf{i}-1} \setminus_T I_B^-) \cup_T (I_A^{\mathsf{Q}_\mathsf{i}-1} \setminus_T I_A^-) \cup_T (I_A^+ \cap_T I_B^+)\Big]}_{\texttt{Pr.2a}}\Big] =$$

$$\underbrace{\Big[\Big[(I_A^{\mathsf{Q}_\mathsf{i}-1} \setminus_T I_A^-) \cup_T I_A^+\Big] \cap_T \Big[(I_B^{\mathsf{Q}_\mathsf{i}-1} \setminus_T I_B^-) \cup_T I_B^+\Big] \cap_T \Big[(I_B^{\mathsf{Q}_\mathsf{i}-1} \setminus_T I_B^-) \cup_T (I_A^{\mathsf{Q}_\mathsf{i}-1} \setminus_T I_A^-)\Big]\Big] \cup_T (I_A^+ \cap_T I_B^+)}_{\texttt{Pr.2a}} =$$

$$\Big[\Big[\underbrace{(I_A^{\mathsf{Q}_\mathsf{i}-1} \setminus_T I_A^-) \cup_T \Big[I_B^+ \cap_T (I_A^{\mathsf{Q}_\mathsf{i}-1} \setminus_T I_A^-)\Big]}_{\texttt{Pr.3a}} \cup_T I_A^+\Big] \cap_T \Big[(I_B^{\mathsf{Q}_\mathsf{i}-1} \setminus_T I_B^-) \cup_T I_B^+\Big] \cap_T \Big[(I_B^{\mathsf{Q}_\mathsf{i}-1} \setminus_T I_B^-) \cup_T (I_A^{\mathsf{Q}_\mathsf{i}-1} \setminus_T I_A^-)\Big]\Big] \cup_T (I_A^+ \cap_T I_B^+) =$$

$$\Big[\underbrace{\Big[(I_A^{\mathsf{Q}_\mathsf{i}-1} \setminus_T I_A^-) \cup_T I_A^+ \cup_T \Big[I_B^+ \cap_T (I_A^{\mathsf{Q}_\mathsf{i}-1} \setminus_T I_A^-)\Big]\Big]}_{\texttt{Pr.1a}} \cap_T \underbrace{\Big[(I_B^{\mathsf{Q}_\mathsf{i}-1} \setminus_T I_B^-) \cup_T \Big[I_B^+ \cap_T (I_A^{\mathsf{Q}_\mathsf{i}-1} \setminus_T I_A^-)\Big]\Big]}_{\texttt{Pr.2a}}\Big] \cup_T (I_A^+ \cap_T I_B^+) =$$

$$\underbrace{\Big[\Big[(I_A^{\mathsf{Q}_\mathsf{i}-1} \setminus_T I_A^-) \cup_T I_A^+\Big] \cap_T (I_B^{\mathsf{Q}_\mathsf{i}-1} \setminus_T I_B^-)\Big] \cup_T \Big[I_B^+ \cap_T (I_A^{\mathsf{Q}_\mathsf{i}-1} \setminus_T I_A^-)\Big]}_{\texttt{Pr.2a}} \cup_T (I_A^+ \cap_T I_B^+) =$$

$$\underbrace{\Big[(I_A^{\mathsf{Q}_\mathsf{i}-1} \setminus_T I_A^-) \cap_T (I_B^{\mathsf{Q}_\mathsf{i}-1} \setminus_T I_B^-)\Big] \cup_T \Big[I_A^+ \cap_T (I_B^{\mathsf{Q}_\mathsf{i}-1} \setminus_T I_B^-)\Big]}_{\texttt{Pr.2b}} \cup_T \Big[I_B^+ \cap_T (I_A^{\mathsf{Q}_\mathsf{i}-1} \setminus_T I_A^-)\Big] \cup_T (I_A^+ \cap_T I_B^+) =$$

$$\Big[(I_A^{\mathsf{Q}_\mathsf{i}-1} \setminus_T I_A^-) \cap_T (I_B^{\mathsf{Q}_\mathsf{i}-1} \setminus_T I_B^-)\Big] \cup_T \Big[I_A^+ \cap_T (I_B^{\mathsf{Q}_\mathsf{i}-1} \setminus_T I_B^-)\Big] \cup_T \Big[I_B^+ \cap_T (I_A^{\mathsf{Q}_\mathsf{i}-1} \setminus_T I_A^-)\Big] \cup_T \underbrace{(I_A^+ \cap_T I_B^+) \cup_T (I_A^+ \cap_T I_B^+)}_{I_F \cup_T I_F = I_F} =$$

$$\Big[(I_A^{\mathsf{Q}_\mathsf{i}-1} \setminus_T I_A^-) \cap_T (I_B^{\mathsf{Q}_\mathsf{i}-1} \setminus_T I_B^-)\Big] \cup_T \underbrace{\Big[\Big[I_B^+ \cap_T (I_A^{\mathsf{Q}_\mathsf{i}-1} \setminus_T I_A^-)\Big] \cup_T (I_A^+ \cap_T I_B^+)\Big] \cup_T \Big[\Big[I_A^+ \cap_T (I_B^{\mathsf{Q}_\mathsf{i}-1} \setminus_T I_B^-)\Big] \cup_T (I_A^+ \cap_T I_B^+)\Big]}_{\texttt{Pr.1a}} =$$

$$\Big[(I_A^{\mathsf{Q}_\mathsf{i}-1} \setminus_T I_A^-) \cap_T (I_B^{\mathsf{Q}_\mathsf{i}-1} \setminus_T I_B^-)\Big] \cup_T \Big[\Big[\underbrace{I_A^{\mathsf{Q}_\mathsf{i}-1} \cap_T (I_B^+ \setminus_T I_A^-)}_{\texttt{Pr.9}}\Big] \cup_T (I_A^+ \cap_T I_B^+)\Big] \cup_T \Big[\Big[\underbrace{I_B^{\mathsf{Q}_\mathsf{i}-1} \cap_T (I_A^+ \setminus_T I_B^-)}_{\texttt{Pr.9}}\Big] \cup_T (I_A^+ \cap_T I_B^+)\Big] =$$

$$\Big[(I_A^{\mathsf{Q}_\mathsf{i}-1} \setminus_T I_A^-) \cap_T (I_B^{\mathsf{Q}_\mathsf{i}-1} \setminus_T I_B^-)\Big] \cup_T \Big[\Big[I_A^{\mathsf{Q}_\mathsf{i}-1} \cap_T \underbrace{I_B^+ \cap_T (I_B^+ \setminus_T I_A^-)}_{\texttt{Pr.11c}}\Big] \cup_T (I_A^+ \cap_T I_B^+)\Big] \cup_T \Big[\Big[I_B^{\mathsf{Q}_\mathsf{i}-1} \cap_T \underbrace{I_A^+ \cap_T (I_A^+ \setminus_T I_B^-)}_{\texttt{Pr.11c}}\Big] \cup_T (I_A^+ \cap_T I_B^+)\Big] =$$

$$\Big[(I_A^{\mathsf{Q}_\mathsf{i}-1} \setminus_T I_A^-) \cap_T (I_B^{\mathsf{Q}_\mathsf{i}-1} \setminus_T I_B^-)\Big] \cup_T \Big[\Big[I_A^{\mathsf{Q}_\mathsf{i}-1} \cap_T \underbrace{(I_B^+ \setminus_T I_B^{\mathsf{Q}_\mathsf{i}-1})}_{I_F^{\mathsf{Q}_\mathsf{i}-1} \cap_T I_F^+ = \varnothing} \cap_T (I_B^+ \setminus_T I_A^-)\Big] \cup_T (I_A^+ \cap_T I_B^+)\Big] \cup_T \Big[\Big[I_B^{\mathsf{Q}_\mathsf{i}-1} \cap_T \underbrace{(I_A^+ \setminus_T I_A^{\mathsf{Q}_\mathsf{i}-1})}_{I_F^{\mathsf{Q}_\mathsf{i}-1} \cap_T I_F^+ = \varnothing} \cap_T (I_A^+ \setminus_T I_B^-)\Big] \cup_T (I_A^+ \cap_T I_B^+)\Big] =$$

$$\Big[(I_A^{\mathsf{Q_{i-1}}} \setminus_T I_A^-) \cap_T (I_B^{\mathsf{Q_{i-1}}} \setminus_T I_B^-)\Big] \cup_T \Big[\Big[I_A^{\mathsf{Q_{i-1}}} \cap_T \underbrace{[I_B^+ \setminus_T (I_B^{\mathsf{Q_{i-1}}} \cup_T I_A^-)]}_{\texttt{Pr.10b}}\Big] \cup_T (I_A^+ \cap_T I_B^+)\Big] \cup_T \Big[\Big[I_B^{\mathsf{Q_{i-1}}} \cap_T \underbrace{[I_A^+ \setminus_T (I_A^{\mathsf{Q_{i-1}}} \cup_T I_B^-)]}_{\texttt{Pr.10b}}\Big] \cup_T (I_A^+ \cap_T I_B^+)\Big] =$$

$$\Big[\underbrace{(I_A^{\mathsf{Q_{i-1}}} \setminus_T I_A^-) \cap_T I_B^{\mathsf{Q_{i-1}}} \cap_T (I_B^{\mathsf{Q_{i-1}}} \setminus_T I_B^-) \cap_T I_A^{\mathsf{Q_{i-1}}}\Big]}_{\texttt{Pr.11c}}\Big] \cup_T \Big[\underbrace{\Big[I_B^+ \cap_T [I_A^{\mathsf{Q_{i-1}}} \setminus_T (I_B^{\mathsf{Q_{i-1}}} \cup_T I_A^-)]}_{\texttt{Pr.9}}\Big] \cup_T (I_A^+ \cap_T I_B^+)\Big] \cup_T \Big[\underbrace{\Big[I_A^+ \cap_T [I_B^{\mathsf{Q_{i-1}}} \setminus_T (I_A^{\mathsf{Q_{i-1}}} \cup_T I_B^-)]}_{\texttt{Pr.9}}\Big] \cup_T (I_A^+ \cap_T I_B^+)\Big] =$$

$$\Big[\underbrace{\Big[(I_A^{\mathsf{Q_{i-1}}} \cap_T I_B^{\mathsf{Q_{i-1}}}) \setminus_T I_A^-\Big]}_{\texttt{Pr.9}} \cap_T \underbrace{\Big[(I_A^{\mathsf{Q_{i-1}}} \cap_T I_B^{\mathsf{Q_{i-1}}}) \setminus_T I_B^-\Big]}_{\texttt{Pr.9}}\Big] \cup_T \underbrace{\Big[I_B^+ \cap_T \Big[[I_A^{\mathsf{Q_{i-1}}} \setminus_T (I_B^{\mathsf{Q_{i-1}}} \cup_T I_A^-)] \cup_T I_A^+\Big]\Big]}_{\texttt{Pr.2b}} \cup_T \underbrace{\Big[I_A^+ \cap_T \Big[[I_B^{\mathsf{Q_{i-1}}} \setminus_T (I_A^{\mathsf{Q_{i-1}}} \cup_T I_B^-)] \cup_T I_B^+\Big]\Big]}_{\texttt{Pr.2b}} =$$

$$\Big[\underbrace{(I_A^{\mathsf{Q_{i-1}}} \cap_T I_B^{\mathsf{Q_{i-1}}}) \setminus_T (I_A^- \cup_T I_B^-)}_{\texttt{Pr.10b}}\Big] \cup_T \Big[I_B^+ \cap_T \Big[\underbrace{[(I_A^{\mathsf{Q_{i-1}}} \setminus_T I_B^{\mathsf{Q_{i-1}}}) \setminus_T I_A^-]}_{\texttt{Pr.5}} \cup_T I_A^+\Big]\Big] \cup_T \Big[I_A^+ \cap_T \Big[\underbrace{[(I_B^{\mathsf{Q_{i-1}}} \setminus_T I_A^{\mathsf{Q_{i-1}}}) \setminus_T I_B^-]}_{\texttt{Pr.5}} \cup_T I_B^+\Big]\Big] =$$

$$\Big[(I_A^{\mathsf{Q_{i-1}}} \cap_T I_B^{\mathsf{Q_{i-1}}}) \setminus_T (I_A^- \cup_T I_B^-)\Big] \cup_T \Big[I_B^+ \cap_T \Big[(\underbrace{I_{A \setminus_T B}^{\mathsf{Q_{i-1}}}}_{I_{A\setminus_T B}^{\mathsf{Q_{i-1}}} = (I_A^{\mathsf{Q_{i-1}}} \setminus_T I_B^{\mathsf{Q_{i-1}}})} \setminus_T I_A^-) \cup_T I_A^+\Big]\Big] \cup_T \Big[I_A^+ \cap_T \Big[(\underbrace{I_{B \setminus_T A}^{\mathsf{Q_{i-1}}}}_{I_{B\setminus_T A}^{\mathsf{Q_{i-1}}} = (I_B^{\mathsf{Q_{i-1}}} \setminus_T I_A^{\mathsf{Q_{i-1}}})} \setminus_T I_B^-) \cup_T I_B^+\Big]\Big]$$

Derivation of Eq. (23) expressing the incremental evaluation of relative_complement_all

$$I_A^{\mathsf{Q_i}} \setminus_T I_B^{\mathsf{Q_i}} =$$

$$\underbrace{\Big[(I_A^{\mathsf{Q_{i-1}}} \setminus_T I_A^-) \cup_T I_A^+\Big]}_{\texttt{Eq. (12)}} \setminus_T \underbrace{\Big[(I_B^{\mathsf{Q_{i-1}}} \setminus_T I_B^-) \cup_T I_B^+\Big]}_{\texttt{Eq. (12)}} =$$

$$\underbrace{\Big[\Big[(I_A^{\mathsf{Q_{i-1}}} \setminus_T I_A^-) \cup I_A^+\Big] \cap_T \Big[(I_A^{\mathsf{Q_{i-1}}} \setminus_T I_A^-) \cup_T I_A^+ \cup_T I_B^-\Big]\Big]}_{\texttt{3b}} \setminus_T \Big[(I_B^{\mathsf{Q_{i-1}}} \setminus_T I_B^-) \cup_T I_B^+\Big] =$$

$$\underbrace{\Big[(I_A^{\mathsf{Q_{i-1}}} \setminus_T I_A^-) \cup I_A^+\Big] \cap_T \Big[\Big[(I_A^{\mathsf{Q_{i-1}}} \setminus_T I_A^-) \cup_T I_A^+ \cup_T I_B^-\Big] \setminus_T \Big[(I_B^{\mathsf{Q_{i-1}}} \setminus_T I_B^-) \cup_T I_B^+\Big]\Big]}_{\texttt{Pr.9}} =$$

$$\Big[(I_A^{\mathsf{Q_{i-1}}} \setminus_T I_A^-) \cup \underbrace{I_A^+ \cup_T \Big[I_A^+ \setminus_T [(I_B^{\mathsf{Q_{i-1}}} \setminus_T I_B^-) \cup_T I_B^+]\Big]}_{\texttt{Pr.11a}}\Big] \cap_T \underbrace{\Big[\Big[[(I_A^{\mathsf{Q_{i-1}}} \setminus_T I_A^-) \cup_T I_B^-] \setminus_T [(I_B^{\mathsf{Q_{i-1}}} \setminus_T I_B^-) \cup_T I_B^+]\Big] \cup_T \Big[I_A^+ \setminus_T [(I_B^{\mathsf{Q_{i-1}}} \setminus_T I_B^-) \cup_T I_B^+]\Big]\Big]}_{\texttt{Pr.6a}} =$$

$$\underbrace{\Big[\Big[(I_A^{\mathsf{Q_{i-1}}} \setminus_T I_A^-) \cup I_A^+\Big] \cap_T \Big[[(I_A^{\mathsf{Q_{i-1}}} \setminus_T I_A^-) \cup_T I_B^-] \setminus_T [(I_B^{\mathsf{Q_{i-1}}} \setminus_T I_B^-) \cup_T I_B^+]\Big]\Big] \cup_T \Big[I_A^+ \setminus_T [(I_B^{\mathsf{Q_{i-1}}} \setminus_T I_B^-) \cup_T I_B^+]\Big]}_{\texttt{Pr.2a}} =$$

$$\Big[\Big[(I_A^{\mathsf{Q_{i-1}}} \setminus_T I_A^-) \cup I_A^+\Big] \cap_T \Big[[(I_A^{\mathsf{Q_{i-1}}} \setminus_T I_A^-) \cup_T I_B^-] \setminus_T [(I_B^{\mathsf{Q_{i-1}}} \setminus_T I_B^-) \cup_T \underbrace{(I_B^+ \setminus_T I_B^-)}_{I_F^- \cap_T I_F^+ = \varnothing}]\Big]\Big] \cup_T \Big[I_A^+ \setminus_T [(I_B^{\mathsf{Q_{i-1}}} \setminus_T I_B^-) \cup_T I_B^+]\Big] =$$

$$\Big[\Big[(I_A^{\mathsf{Q_{i-1}}} \setminus_T I_A^-) \cup I_A^+\Big] \cap_T \Big[[(I_A^{\mathsf{Q_{i-1}}} \setminus_T I_A^-) \cup_T I_B^-] \setminus_T \underbrace{[(I_B^{\mathsf{Q_{i-1}}} \cup_T I_B^+) \setminus_T I_B^-]}_{\texttt{Pr.6a}}\Big]\Big] \cup_T \Big[I_A^+ \setminus_T [(I_B^{\mathsf{Q_{i-1}}} \setminus_T I_B^-) \cup_T I_B^+]\Big] =$$

$$\Big[\underbrace{\Big[[(I_A^{\mathsf{Q_{i-1}}} \setminus_T I_A^-) \setminus_T (I_B^{\mathsf{Q_{i-1}}} \cup_T I_B^+)] \cup_T (I_A^{\mathsf{Q_{i-1}}} \setminus_T I_A^-)}_{\texttt{Pr.11a}} \cup_T I_A^+\Big] \cap_T \underbrace{\Big[[(I_A^{\mathsf{Q_{i-1}}} \setminus_T I_A^-) \setminus_T (I_B^{\mathsf{Q_{i-1}}} \cup_T I_B^+)] \cup_T I_B^-\Big]}_{\texttt{Pr.8}}\Big] \cup_T \Big[I_A^+ \setminus_T [(I_B^{\mathsf{Q_{i-1}}} \setminus_T I_B^-) \cup_T I_B^+]\Big] =$$

$$\underbrace{\Big[(I_A^{\mathsf{Q_i}-1} \setminus_T I_A^-) \setminus_T (I_B^{\mathsf{Q_i}-1} \cup_T I_B^+)\Big] \cup_T \Big[\Big[(I_A^{\mathsf{Q_i}-1} \setminus_T I_A^-) \cup_T I_A^+\Big] \cap_T I_B^-\Big]}_{\texttt{Pr.2a}} \cup_T \Big[I_A^+ \setminus_T \Big[(I_B^{\mathsf{Q_i}-1} \setminus_T I_B^-) \cup_T I_B^+\Big]\Big] =$$

$$\underbrace{\Big[(I_A^{\mathsf{Q_i}-1} \setminus_T I_A^-) \setminus_T (I_B^{\mathsf{Q_i}-1} \cup_T I_B^+)\Big] \cup_T \Big[I_A^+ \setminus_T \Big[(I_B^{\mathsf{Q_i}-1} \setminus_T I_B^-) \cup_T I_B^+\Big]\Big] \cup_T \Big[\Big[(I_A^{\mathsf{Q_i}-1} \setminus_T I_A^-) \cup_T I_A^+\Big] \cap_T I_B^-\Big]}_{\texttt{Pr.1a}} =$$

$$\Big[(I_A^{\mathsf{Q_i}-1} \setminus_T I_A^-) \setminus_T (I_B^{\mathsf{Q_i}-1} \cup_T I_B^+)\Big] \cup_T \Big[\Big[\underbrace{(I_A^+ \cap_T I_A^{\mathsf{Q_i}-1})}_{I_F^{\mathsf{Q_i}-1} \cap_T I_F^+ = \varnothing} \setminus_T I_B^+\Big] \cup_T \Big[I_A^+ \setminus_T \big[(I_B^{\mathsf{Q_i}-1} \setminus_T I_B^-) \cup_T I_B^+\big]\Big]\Big] \cup_T \Big[\Big[(I_A^{\mathsf{Q_i}-1} \setminus_T I_A^-) \cup_T I_A^+\Big] \cap_T I_B^-\Big] =$$

$$\Big[(I_A^{\mathsf{Q_i}-1} \setminus_T I_A^-) \setminus_T (I_B^{\mathsf{Q_i}-1} \cup_T I_B^+)\Big] \cup_T \Big[\underbrace{\Big[I_A^+ \cap_T (I_A^{\mathsf{Q_i}-1} \setminus_T I_B^+)\Big]}_{\texttt{Pr.9}} \cup_T \Big[I_A^+ \setminus_T \big[(I_B^{\mathsf{Q_i}-1} \setminus_T I_B^-) \cup_T I_B^+\big]\Big]\Big] \cup_T \underbrace{\Big[\Big[(I_A^{\mathsf{Q_i}-1} \setminus_T I_A^-) \cap_T I_B^-\Big] \cup_T \Big[I_A^+ \cap_T I_B^-\Big]\Big]}_{\texttt{Pr.2b}} =$$

$$\Big[(I_A^{\mathsf{Q_i}-1} \setminus_T I_A^-) \setminus_T (I_B^{\mathsf{Q_i}-1} \cup_T I_B^+)\Big] \cup_T \underbrace{\Big[I_A^+ \setminus_T \Big[\big[(I_B^{\mathsf{Q_i}-1} \setminus_T I_B^-) \cup_T I_B^+\big] \setminus_T \big[I_A^{\mathsf{Q_i}-1} \setminus_T I_B^+\big]\Big]\Big]}_{\texttt{Pr.7}} \cup_T \Big[\Big[(I_A^{\mathsf{Q_i}-1} \setminus_T I_A^-) \cap_T \underbrace{I_B^{\mathsf{Q_i}-1} \cap_T I_B^-}_{I_F^{\mathsf{Q_i}-1} \cap_T I_F^- = I_F^-}\Big] \cup_T \Big[I_A^+ \cap_T I_B^-\Big]\Big] =$$

$$\Big[(I_A^{\mathsf{Q_i}-1} \setminus_T I_A^-) \setminus_T (I_B^{\mathsf{Q_i}-1} \cup_T I_B^+)\Big] \cup_T \Big[I_A^+ \setminus_T \underbrace{\Big[\big[(I_B^{\mathsf{Q_i}-1} \setminus_T I_B^-) \setminus_T I_A^{\mathsf{Q_i}-1}\big] \cup_T I_B^+\Big]}_{\texttt{Pr.8}}\Big] \cup_T \underbrace{\Big[I_B^- \cap_T \Big[\big[(I_A^{\mathsf{Q_i}-1} \setminus_T I_A^-) \cap_T I_B^{\mathsf{Q_i}-1}\big] \cup_T I_A^+\Big]\Big]}_{\texttt{Pr.2b}} =$$

$$\underbrace{\Big[(I_A^{\mathsf{Q_i}-1} \setminus_T I_B^{\mathsf{Q_i}-1}) \setminus_T (I_A^- \cup_T I_B^+)\Big]}_{\texttt{Pr.5}} \cup_T \Big[I_A^+ \setminus_T \Big[\underbrace{\big[(I_B^{\mathsf{Q_i}-1} \setminus_T I_A^{\mathsf{Q_i}-1}) \setminus_T I_B^-\big]}_{\texttt{Pr.5}} \cup_T I_B^+\Big]\Big] \cup_T \Big[I_B^- \cap_T \Big[\underbrace{\big[(I_A^{\mathsf{Q_i}-1} \cap_T I_B^{\mathsf{Q_i}-1}) \setminus_T I_A^-\big]}_{\texttt{Pr.9}} \cup_T I_A^+\Big]\Big] =$$

$$\Big[(I_A^{\mathsf{Q_i}-1} \setminus_T I_B^{\mathsf{Q_i}-1}) \setminus_T (I_A^- \cup_T I_B^+)\Big] \cup_T \Big[I_A^+ \setminus_T \Big[(\underbrace{I_{B \setminus_T A}^{\mathsf{Q_i}-1}}_{I_{B \setminus_T A}^{\mathsf{Q_i}-1} = I_B^{\mathsf{Q_i}-1} \setminus_T I_A^{\mathsf{Q_i}-1}} \setminus_T I_B^-) \cup_T I_B^+\Big]\Big] \cup_T \Big[I_B^- \cap_T \Big[(\underbrace{I_{A \cap_T B}^{\mathsf{Q_i}-1}}_{I_{A \cap_T B}^{\mathsf{Q_i}-1} = I_A^{\mathsf{Q_i}-1} \cap_T I_B^{\mathsf{Q_i}-1}} \setminus_T I_A^-) \cup_T I_A^+\Big]\Big]$$

Complete Comparison for Statically Determined Fluents.

Table 17: The 16 different cases of union_all according to the deletion and insertion sets of the two body fluents. Bold lines highlight the cases in which $RTEC_{inc}$ outperforms RTEC.

$\sharp$	$\|I_A^-\|$	$\|I_A^+\|$	$\|I_B^-\|$	$\|I_B^+\|$	**RTEC vs $RTEC_{inc}$**
1	$\neq 0$	$\neq 0$	$\neq 0$	$\neq 0$	$\|I_{A\cup_T B}^{\mathsf{Q_{i-1}}}\| \nless \|I_{AB}^{\mathsf{Q_{i-1}}}\| - \frac{3}{2}\|I_{A\cap_T B}^{\mathsf{Q_{i-1}}}\| - 5\|I_{AB}^{-+}\|$
2	0	$\neq 0$	$\neq 0$	$\neq 0$	$\|I_{A\cup_T B}^{\mathsf{Q_{i-1}}}\| \nless \|I_{AB}^{\mathsf{Q_{i-1}}}\| - \frac{3}{2}\|I_{A\cap_T B}^{\mathsf{Q_{i-1}}}\| - 2\|I_{AB}^{-+}\|$
3	$\neq 0$	0	$\neq 0$	$\neq 0$	$\|I_{A\cup_T B}^{\mathsf{Q_{i-1}}}\| \nless \|I_{AB}^{\mathsf{Q_{i-1}}}\| - \frac{3}{2}\|I_{A\cap_T B}^{\mathsf{Q_{i-1}}}\| - 4\|I_{AB}^{-+}\|$
4	0	0	$\neq 0$	$\neq 0$	$\|I_{A\cup_T B}^{\mathsf{Q_{i-1}}}\| \nless \|I_{AB}^{\mathsf{Q_{i-1}}}\| - \frac{3}{2}\|I_{A\cap_T B}^{\mathsf{Q_{i-1}}}\| - \|I_{AB}^{-+}\|$
5	$\neq 0$	$\neq 0$	0	$\neq 0$	$\|I_{A\cup_T B}^{\mathsf{Q_{i-1}}}\| \nless \|I_{AB}^{\mathsf{Q_{i-1}}}\| - \frac{3}{2}\|I_{A\cap_T B}^{\mathsf{Q_{i-1}}}\| - 2\|I_{AB}^{-+}\|$
6	**0**	$\boldsymbol{\neq 0}$	**0**	$\boldsymbol{\neq 0}$	$\boldsymbol{\|I_{A\cup_T B}^{\mathsf{Q_{i-1}}}\| < 2\|I_{AB}^{\mathsf{Q_{i-1}}}\| - 2\|I_{AB}^{-+}\|}$
7	$\neq 0$	0	0	$\neq 0$	$\|I_{A\cup_T B}^{\mathsf{Q_{i-1}}}\| \nless \|I_{AB}^{\mathsf{Q_{i-1}}}\| - \frac{3}{2}\|I_{A\cap_T B}^{\mathsf{Q_{i-1}}}\| - \|I_{AB}^{-+}\|$
8	**0**	**0**	**0**	$\boldsymbol{\neq 0}$	$\boldsymbol{\|I_{A\cup_T B}^{\mathsf{Q_{i-1}}}\| < 2\|I_{AB}^{\mathsf{Q_{i-1}}}\|}$
9	$\neq 0$	$\neq 0$	$\neq 0$	0	$\|I_{A\cup_T B}^{\mathsf{Q_{i-1}}}\| \nless \|I_{AB}^{\mathsf{Q_{i-1}}}\| - \frac{3}{2}\|I_{A\cap_T B}^{\mathsf{Q_{i-1}}}\| - 4\|I_{AB}^{-+}\|$
10	0	$\neq 0$	$\neq 0$	0	$\|I_{A\cup_T B}^{\mathsf{Q_{i-1}}}\| \nless \|I_{AB}^{\mathsf{Q_{i-1}}}\| - \frac{3}{2}\|I_{A\cap_T B}^{\mathsf{Q_{i-1}}}\| - \|I_{AB}^{-+}\|$
11	$\neq 0$	0	$\neq 0$	0	$\|I_{A\cup_T B}^{\mathsf{Q_{i-1}}}\| \nless 2\|I_{AB}^{\mathsf{Q_{i-1}}}\| - 2\|I_{A\cap_T B}^{\mathsf{Q_{i-1}}}\| - 3\|I_{AB}^{-+}\|$
12	0	0	$\neq 0$	0	$\|I_{A\cup_T B}^{\mathsf{Q_{i-1}}}\| \nless 2\|I_{AB}^{\mathsf{Q_{i-1}}}\| - 2\|I_{A\cap_T B}^{\mathsf{Q_{i-1}}}\| - \frac{\|I_{AB}^{-+}\|}{2}$
13	$\neq 0$	$\neq 0$	0	0	$\|I_{A\cup_T B}^{\mathsf{Q_{i-1}}}\| \nless \|I_{AB}^{\mathsf{Q_{i-1}}}\| - \frac{3}{2}\|I_{A\cap_T B}^{\mathsf{Q_{i-1}}}\| - \|I_{AB}^{-+}\|$
14	**0**	$\boldsymbol{\neq 0}$	**0**	**0**	$\boldsymbol{\|I_{A\cup_T B}^{\mathsf{Q_{i-1}}}\| < 2\|I_{AB}^{\mathsf{Q_{i-1}}}\|}$
15	$\neq 0$	0	0	0	$\|I_{A\cup_T B}^{\mathsf{Q_{i-1}}}\| \nless 2\|I_{AB}^{\mathsf{Q_{i-1}}}\| - 2\|I_{A\cap_T B}^{\mathsf{Q_{i-1}}}\| - \|I_{AB}^{-+}\|$
16	**0**	**0**	**0**	**0**	$\top$

Table 18: The 16 different cases of $\mathsf{intersect\_all}$ according to the deletion and insertion sets of the two body fluents. Bold lines highlight the cases in which $RTEC_{inc}$ outperforms RTEC.

♯	$\lvert I^{-}_{A}\rvert$	$\lvert I^{+}_{A}\rvert$	$\lvert I^{-}_{B}\rvert$	$\lvert I^{+}_{B}\rvert$	**RTEC vs** $RTEC_{inc}$
1	$\neq 0$	$\neq 0$	$\neq 0$	$\neq 0$	$\lvert I^{\mathsf{Q_{i-1}}}_{A\cap_T B}\rvert \not< \lvert I^{\mathsf{Q_{i-1}}}_{AB}\rvert - 5\lvert I^{\mathsf{Q_{i-1}}}_{A\setminus_T B}\rvert - \frac{19}{2}\lvert I^{-+}_{AB}\rvert$
2	0	$\neq 0$	$\neq 0$	$\neq 0$	$\lvert I^{\mathsf{Q_{i-1}}}_{A\cap_T B}\rvert \not< \lvert I^{\mathsf{Q_{i-1}}}_{AB}\rvert - \frac{9}{2}\lvert I^{\mathsf{Q_{i-1}}}_{A\setminus_T B}\rvert - 6\lvert I^{-+}_{AB}\rvert$
3	$\neq 0$	**0**	$\neq 0$	$\neq 0$	$\lvert I^{\mathsf{Q_{i-1}}}_{A\cap_T B}\rvert < \lvert I^{\mathsf{Q_{i-1}}}_{AB}\rvert - \frac{3}{2}\lvert I^{\mathsf{Q_{i-1}}}_{A\setminus_T B}\rvert - 2\lvert I^{-+}_{AB}\rvert$
4	**0**	**0**	$\neq 0$	$\neq 0$	$\lvert I^{\mathsf{Q_{i-1}}}_{A\cap_T B}\rvert < \lvert I^{\mathsf{Q_{i-1}}}_{AB}\rvert - \lvert I^{\mathsf{Q_{i-1}}}_{A\setminus_T B}\rvert$
5	$\neq 0$	$\neq 0$	0	$\neq 0$	$\lvert I^{\mathsf{Q_{i-1}}}_{A\cap_T B}\rvert \not< \lvert I^{\mathsf{Q_{i-1}}}_{AB}\rvert - \frac{5}{2}\lvert I^{\mathsf{Q_{i-1}}}_{A\setminus_T B}\rvert - 6\lvert I^{-+}_{AB}\rvert$
6	**0**	$\neq 0$	**0**	$\neq 0$	$\lvert I^{\mathsf{Q_{i-1}}}_{A\cap_T B}\rvert < 2\lvert I^{\mathsf{Q_{i-1}}}_{AB}\rvert - 8\lvert I^{\mathsf{Q_{i-1}}}_{A\setminus_T B}\rvert - \frac{7}{2}\lvert I^{-+}_{AB}\rvert$
7	$\neq 0$	**0**	**0**	$\neq 0$	$\lvert I^{\mathsf{Q_{i-1}}}_{A\cap_T B}\rvert < \lvert I^{\mathsf{Q_{i-1}}}_{AB}\rvert - \frac{3}{2}\lvert I^{\mathsf{Q_{i-1}}}_{A\setminus_T B}\rvert - \frac{3}{2}\lvert I^{-+}_{AB}\rvert$
8	**0**	**0**	**0**	$\neq 0$	$\lvert I^{\mathsf{Q_{i-1}}}_{A\cap_T B}\rvert < 2\lvert I^{\mathsf{Q_{i-1}}}_{AB}\rvert - 2\lvert I^{\mathsf{Q_{i-1}}}_{A\setminus_T B}\rvert$
9	$\neq 0$	$\neq 0$	$\neq 0$	**0**	$\lvert I^{\mathsf{Q_{i-1}}}_{A\cap_T B}\rvert < \lvert I^{\mathsf{Q_{i-1}}}_{AB}\rvert - \frac{3}{2}\lvert I^{\mathsf{Q_{i-1}}}_{A\setminus_T B}\rvert - 2\lvert I^{-+}_{AB}\rvert$
10	**0**	$\neq 0$	$\neq 0$	**0**	$\lvert I^{\mathsf{Q_{i-1}}}_{A\cap_T B}\rvert < \lvert I^{\mathsf{Q_{i-1}}}_{AB}\rvert - \frac{3}{2}\lvert I^{\mathsf{Q_{i-1}}}_{A\setminus_T B}\rvert - \frac{3}{2}\lvert I^{-+}_{AB}\rvert$
11	$\neq 0$	**0**	$\neq 0$	**0**	$\lvert I^{\mathsf{Q_{i-1}}}_{A\cap_T B}\rvert < 2\lvert I^{\mathsf{Q_{i-1}}}_{AB}\rvert - 2\lvert I^{-+}_{AB}\rvert$
12	**0**	**0**	$\neq 0$	**0**	$\lvert I^{\mathsf{Q_{i-1}}}_{A\cap_T B}\rvert < 2\lvert I^{\mathsf{Q_{i-1}}}_{AB}\rvert$
13	$\neq 0$	$\neq 0$	**0**	**0**	$\lvert I^{\mathsf{Q_{i-1}}}_{A\cap_T B}\rvert < \lvert I^{\mathsf{Q_{i-1}}}_{AB}\rvert - \lvert I^{\mathsf{Q_{i-1}}}_{A\setminus_T B}\rvert$
14	**0**	$\neq 0$	**0**	**0**	$\lvert I^{\mathsf{Q_{i-1}}}_{A\cap_T B}\rvert < 2\lvert I^{\mathsf{Q_{i-1}}}_{AB}\rvert - 2\lvert I^{\mathsf{Q_{i-1}}}_{A\setminus_T B}\rvert$
15	$\neq 0$	**0**	**0**	**0**	$\lvert I^{\mathsf{Q_{i-1}}}_{A\cap_T B}\rvert < 2\lvert I^{\mathsf{Q_{i-1}}}_{AB}\rvert$
16	**0**	**0**	**0**	**0**	$\top$

Table 19: The 16 different cases of relative_complement_all according to the deletion and insertion sets of the two body fluents. Bold lines highlight the cases in which *RTEC*$RTEC_{inc}$ outperforms RTEC.

$\sharp$	$\lvert I_A^- \rvert$	$\lvert I_A^+ \rvert$	$\lvert I_B^- \rvert$	$\lvert I_B^+ \rvert$	**RTEC vs $\boldsymbol{RTEC_{inc}}$**
1	$\neq 0$	$\neq 0$	$\neq 0$	$\neq 0$	$\lvert I_{A\setminus_T B}^{\mathsf{Q_{i-1}}} \rvert \not< \lvert I_{AB}^{\mathsf{Q_{i-1}}} \rvert - \frac{5}{2}(\lvert I_{B\setminus_T A}^{\mathsf{Q_{i-1}}} \rvert + \lvert I_{A\cap_T B}^{\mathsf{Q_{i-1}}} \rvert) - 8\lvert I_{AB}^{-+} \rvert$
2	0	$\neq 0$	$\neq 0$	$\neq 0$	$\lvert I_{A\setminus_T B}^{\mathsf{Q_{i-1}}} \rvert \not< \lvert I_{AB}^{\mathsf{Q_{i-1}}} \rvert - \frac{5}{2}\lvert I_{B\setminus_T A}^{\mathsf{Q_{i-1}}} \rvert - 2\lvert I_{A\cap_T B}^{\mathsf{Q_{i-1}}} \rvert - \frac{9}{2}\lvert I_{AB}^{-+} \rvert$
3	$\boldsymbol{\neq 0}$	**0**	$\boldsymbol{\neq 0}$	$\boldsymbol{\neq 0}$	$\boldsymbol{\lvert I_{A\setminus_T B}^{\mathsf{Q_{i-1}}} \rvert < \lvert I_{AB}^{\mathsf{Q_{i-1}}} \rvert - \frac{3}{2}\lvert I_{A\cap_T B}^{\mathsf{Q_{i-1}}} \rvert - \frac{5}{2}\lvert I_{AB}^{-+} \rvert}$
4	**0**	**0**	$\boldsymbol{\neq 0}$	$\boldsymbol{\neq 0}$	$\boldsymbol{\lvert I_{A\setminus_T B}^{\mathsf{Q_{i-1}}} \rvert < \lvert I_{AB}^{\mathsf{Q_{i-1}}} \rvert - \lvert I_{A\cap_T B}^{\mathsf{Q_{i-1}}} \rvert}$
5	$\boldsymbol{\neq 0}$	$\boldsymbol{\neq 0}$	**0**	$\boldsymbol{\neq 0}$	$\boldsymbol{\lvert I_{A\setminus_T B}^{\mathsf{Q_{i-1}}} \rvert < \lvert I_{AB}^{\mathsf{Q_{i-1}}} \rvert - \frac{3}{2}\lvert I_{B\setminus_T A}^{\mathsf{Q_{i-1}}} \rvert - 3\lvert I_{AB}^{-+} \rvert}$
6	**0**	$\boldsymbol{\neq 0}$	**0**	$\boldsymbol{\neq 0}$	$\boldsymbol{\lvert I_{A\setminus_T B}^{\mathsf{Q_{i-1}}} \rvert < \lvert I_{AB}^{\mathsf{Q_{i-1}}} \rvert - \frac{3}{2}\lvert I_{B\setminus_T A}^{\mathsf{Q_{i-1}}} \rvert - 2\lvert I_{AB}^{-+} \rvert}$
7	$\boldsymbol{\neq 0}$	**0**	**0**	$\boldsymbol{\neq 0}$	$\boldsymbol{\lvert I_{A\setminus_T B}^{\mathsf{Q_{i-1}}} \rvert < 2\lvert I_{AB}^{\mathsf{Q_{i-1}}} \rvert - 2\lvert I_{AB}^{-+} \rvert}$
8	**0**	**0**	**0**	$\boldsymbol{\neq 0}$	$\boldsymbol{\lvert I_{A\setminus_T B}^{\mathsf{Q_{i-1}}} \rvert < 2\lvert I_{AB}^{\mathsf{Q_{i-1}}} \rvert}$
9	$\neq 0$	$\neq 0$	$\neq 0$	0	$\lvert I_{A\setminus_T B}^{\mathsf{Q_{i-1}}} \rvert \not< \lvert I_{AB}^{\mathsf{Q_{i-1}}} \rvert - \frac{5}{2}\lvert I_{A\cap_T B}^{\mathsf{Q_{i-1}}} \rvert - 2\lvert I_{B\setminus_T A}^{\mathsf{Q_{i-1}}} \rvert - 5\lvert I_{AB}^{-+} \rvert$
10	0	$\neq 0$	$\neq 0$	0	$\lvert I_{A\setminus_T B}^{\mathsf{Q_{i-1}}} \rvert \not< 2\lvert I_{AB}^{\mathsf{Q_{i-1}}} \rvert - 4(\lvert I_{B\setminus_T A}^{\mathsf{Q_{i-1}}} \rvert + \lvert I_{A\cap_T B}^{\mathsf{Q_{i-1}}} \rvert) - \frac{5}{2}\lvert I_{AB}^{-+} \rvert$
11	$\boldsymbol{\neq 0}$	**0**	$\boldsymbol{\neq 0}$	**0**	$\boldsymbol{\lvert I_{A\setminus_T B}^{\mathsf{Q_{i-1}}} \rvert < \lvert I_{AB}^{\mathsf{Q_{i-1}}} \rvert - \frac{3}{2}\lvert I_{A\cap_T B}^{\mathsf{Q_{i-1}}} \rvert - \frac{3}{2}\lvert I_{AB}^{-+} \rvert}$
12	**0**	**0**	$\boldsymbol{\neq 0}$	**0**	$\boldsymbol{\lvert I_{A\setminus_T B}^{\mathsf{Q_{i-1}}} \rvert < 2\lvert I_{AB}^{\mathsf{Q_{i-1}}} \rvert - 2\lvert I_{A\cap_T B}^{\mathsf{Q_{i-1}}} \rvert}$
13	$\boldsymbol{\neq 0}$	$\boldsymbol{\neq 0}$	**0**	**0**	$\boldsymbol{\lvert I_{A\setminus_T B}^{\mathsf{Q_{i-1}}} \rvert < \lvert I_{AB}^{\mathsf{Q_{i-1}}} \rvert - \lvert I_{B\setminus_T A}^{\mathsf{Q_{i-1}}} \rvert - \frac{\lvert I_{AB}^{-+} \rvert}{2}}$
14	**0**	$\boldsymbol{\neq 0}$	**0**	**0**	$\boldsymbol{\lvert I_{A\setminus_T B}^{\mathsf{Q_{i-1}}} \rvert < 2\lvert I_{AB}^{\mathsf{Q_{i-1}}} \rvert - 2\lvert I_{B\setminus_T B}^{\mathsf{Q_{i-1}}} \rvert - \frac{\lvert I_{AB}^{-+} \rvert}{2}}$
15	$\boldsymbol{\neq 0}$	**0**	**0**	**0**	$\boldsymbol{\lvert I_{A\setminus_T B}^{\mathsf{Q_{i-1}}} \rvert < 2\lvert I_{AB}^{\mathsf{Q_{i-1}}} \rvert}$
16	**0**	**0**	**0**	**0**	$\top$

References

Ajileye, T., Motik, B., & Horrocks, I. (2019). Datalog materialisation in distributed RDF stores with dynamic data exchange. In *The Semantic Web - ISWC 2019 - 18th International Semantic Web Conference, Auckland, New Zealand, October 26-30, 2019, Proceedings, Part I*, pp. 21–37.

Alevizos, E., Skarlatidis, A., Artikis, A., & Paliouras, G. (2017). Probabilistic complex event recognition: A survey. *ACM Comput. Surv.*, *50*(5), 71:1–71:31.

Anicic, D., Rudolph, S., Fodor, P., & Stojanovic, N. (2012). Real-time complex event recognition and reasoning - a logic programming approach. *Applied Artificial Intelligence*, *26*(1-2), 6–57.

Arasu, A., Babu, S., & Widom, J. (2006). The cql continuous query language: Semantic foundations and query execution. *The VLDB Journal*, *15*(2), 121–142.

Artikis, A., Sergot, M. J., & Paliouras, G. (2015). An event calculus for event recognition. *IEEE Trans. Knowl. Data Eng.*, *27*(4), 895–908.

Barbieri, D. F., Braga, D., Ceri, S., Della Valle, E., & Grossniklaus, M. (2010). Incremental reasoning on streams and rich background knowledge. In Aroyo, L., Antoniou, G., Hyvönen, E., ten Teije, A., Stuckenschmidt, H., Cabral, L., & Tudorache, T. (Eds.), *The Semantic Web: Research and Applications*, pp. 1–15, Berlin, Heidelberg. Springer Berlin Heidelberg.

Bassiouni, M., Llewellyn, M., & Mukherjee, A. (1993). Time-based operators for relational algebra query languages. *Computer Languages*, *19*(4), 261 – 276.

Beck, H., Dao-Tran, M., & Eiter, T. (2018). LARS: A logic-based framework for analytic reasoning over streams. *Artif. Intell.*, *261*, 16–70.

Brandt, S., Kalayci, E. G., Ryzhikov, V., Xiao, G., & Zakharyaschev, M. (2018). Querying log data with metric temporal logic. *Journal of Artificial Intelligence Research*, *62*(1), 829–877.

Camossi, E., Jousselme, A.-L., Ray, C., Hadzagic, M., Dreo, R., & Claramunt, C. (2017). Maritime experiments specification, h2020 datacron project deliverable d5.3.. http://datacron-project.eu/.

Ceri, S., & Widom, J. (1991). Deriving production rules for incremental view maintenance. In *Proceedings of the 17th International Conference on Very Large Data Bases*, VLDB '91, pp. 577–589, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Cervesato, I., & Montanari, A. (2000). A calculus of macro-events: Progress report. In *Seventh International Workshop on Temporal Representation and Reasoning, TIME 2000, Nova Scotia, Canada, July 7-9, 2000*, pp. 47–58.

Chittaro, L., & Montanari, A. (1996). Efficient temporal reasoning in the cached event calculus. *Computational Intelligence*, *12*(3), 359–382.

Clark, K. L. (1977). Negation as failure. In *Logic and Data Bases, Symposium on Logic and Data Bases, Centre d'études et de recherches de Toulouse, France, 1977*, pp. 293–322.

Cugola, G., & Margara, A. (2010). TESLA: a formally defined event specification language. In *Proceedings of the Fourth ACM International Conference on Distributed Event-Based Systems, DEBS 2010, Cambridge, United Kingdom, July 12-15, 2010*, pp. 50–61.

Cugola, G., & Margara, A. (2012). Complex event processing with t-rex. *Journal of Systems and Software*, *85*(8), 1709 – 1728.

Demers, A. J., Gehrke, J., Panda, B., Riedewald, M., Sharma, V., & White, W. M. (2007). Cayuga: A general purpose event monitoring system. In *CIDR 2007, Third Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, January 7-10, 2007, Online Proceedings*, pp. 412–422.

Dindar, N., Fischer, P. M., Soner, M., & Tatbul, N. (2011). Efficiently correlating complex events over live and archived data streams. In *Proceedings of the 5th ACM International Conference on Distributed Event-based System*, DEBS '11, pp. 243–254, New York, NY, USA. ACM.

Dong, G., & Su, J. (1994). First-order incremental evaluation of datalog queries. In Beeri, C., Ohori, A., & Shasha, D. E. (Eds.), *Database Programming Languages (DBPL-4)*, pp. 295–308, London. Springer London.

Dong, G., Su, J., & Topor, R. (1995). Nonrecursive incremental evaluation of datalog queries. *Annals of Mathematics and Artificial Intelligence*, *14*(2), 187–223.

Dong, G., & Topor, R. (1992). Incremental evaluation of datalog queries. In Biskup, J., & Hull, R. (Eds.), *Database Theory — ICDT '92*, pp. 282–296, Berlin, Heidelberg. Springer Berlin Heidelberg.

Dong, G., & Su, J. (1995). Incremental and decremental evaluation of transitive closure by first-order queries. *Information and Computation*, *120*(1), 101 – 106.

Dousson, C., & Maigat, P. L. (2007). Chronicle recognition improvement using temporal focusing and hierarchization. In *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007*, pp. 324–329.

Fern, A., Givan, R., & Siskind, J. M. (2002). Specific-to-general learning for temporal events with application to learning event definitions from video. *Journal of Artificial Intelligence Research*, *17*(1), 379–449.

Giatrakos, N., Alevizos, E., Artikis, A., Deligiannakis, A., & Garofalakis, M. N. (2020). Complex event recognition in the big data era: a survey. *VLDB J.*, *29*(1), 313–352.

Grez, A., Riveros, C., & Ugarte, M. (2019). A formal framework for complex event processing. In *22nd International Conference on Database Theory, ICDT 2019, March 26-28, 2019, Lisbon, Portugal*, pp. 5:1–5:18.

Grez, A., Riveros, C., Ugarte, M., & Vansummeren, S. (2020). On the Expressiveness of Languages for Complex Event Recognition. In Lutz, C., & Jung, J. C. (Eds.), *23rd International Conference on Database Theory (ICDT 2020)*, Vol. 155 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pp. 15:1–15:17, Dagstuhl, Germany. Schloss Dagstuhl–Leibniz-Zentrum für Informatik.

Gupta, A., Mumick, I. S., & Subrahmanian, V. S. (1993). Maintaining views incrementally. *SIGMOD Rec.*, *22*(2), 157–166.

Gyllstrom, D., Wu, E., Chae, H., Diao, Y., Stahlberg, P., & Anderson, G. (2006). SASE: complex event processing over streams. *CoRR*, *abs/cs/0612128*.

Hu, P., Motik, B., & Horrocks, I. (2018). Optimised maintenance of datalog materialisations. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pp. 1871–1879.

Hu, P., Motik, B., & Horrocks, I. (2019). Modular materialisation of datalog programs. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pp. 2859–2866.

Idris, M., Ugarte, M., Vansummeren, S., Voigt, H., & Lehner, W. (2019). Efficient query processing for dynamically changing datasets. *SIGMOD Rec.*, *48*(1), 33–40.

Koch, C., Ahmad, Y., Kennedy, O., Nikolic, M., Nötzli, A., Lupei, D., & Shaikhha, A. (2014). Dbtoaster: higher-order delta processing for dynamic, frequently fresh views. *The VLDB Journal*, *23*(2), 253–278.

Kowalski, R. A., & Sergot, M. J. (1986). A logic-based calculus of events. *New Generation Comput.*, *4*(1), 67–95.

Lee, J., & Palla, R. (2012). Reformulating the situation calculus and the event calculus in the general theory of stable models and in answer set programming. *Journal of Artificial Intelligence Research*, *43*, 571–620.

Li, M., Mani, M., Rundensteiner, E. A., & Lin, T. (2011). Complex event pattern detection over streams with interval-based temporal semantics. In *Proceedings of the 5th ACM International Conference on Distributed Event-based System*, DEBS '11, pp. 291–302, New York, NY, USA. ACM.

Liu, M., Rundensteiner, E. A., Greenfield, K., Gupta, C., Wang, S., Ari, I., & Mehta, A. (2011). E-cube: multi-dimensional event sequence analysis using hierarchical pattern query sharing. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2011, Athens, Greece, June 12-16, 2011*, pp. 889–900.

Mei, Y., & Madden, S. (2009). Zstream: a cost-based query processor for adaptively detecting composite events. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2009, Providence, Rhode Island, USA, June 29 - July 2, 2009*, pp. 193–206.

Miller, R., & Shanahan, M. (2002). Some alternative formulations of the event calculus. In *Computational Logic: Logic Programming and Beyond, Essays in Honour of Robert A. Kowalski, Part II*, pp. 452–490.

Montali, M., Maggi, F. M., Chesani, F., Mello, P., & van der Aalst, W. M. P. (2013). Monitoring business constraints with the event calculus. *ACM TIST*, *5*(1), 17:1–17:30.

Motik, B., Nenov, Y., Piro, R., & Horrocks, I. (2015). Incremental update of datalog materialisation: The backward/forward algorithm. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, pp. 1560–1568. AAAI Press.

Motik, B., Nenov, Y., Piro, R., & Horrocks, I. (2019). Maintenance of datalog materialisations revisited. *Artif. Intell.*, *269*, 76–136.

Mutschler, C., & Philippsen, M. (2014). Adaptive speculative processing of out-of-order event streams. *ACM Trans. Internet Techn.*, *14*(1), 4:1–4:24.

Paschke, A. (2006). Eca-ruleml: An approach combining ECA rules with temporal interval-based KR event/action logics and transactional update logics. *CoRR*, *abs/cs/0610167*.

Paschke, A., & Bichler, M. (2008). Knowledge representation concepts for automated SLA management. *Decision Support Systems*, *46*(1), 187–205.

Patroumpas, K., Alevizos, E., Artikis, A., Vodas, M., Pelekis, N., & Theodoridis, Y. (2017). Online event recognition from moving vessel trajectories. *GeoInformatica*, *21*(2), 389–427.

Pitsikalis, M., Artikis, A., Dreo, R., Ray, C., Camossi, E., & Jousselme, A.-L. (2019). Composite event recognition for maritime monitoring. In *Proceedings of the 13th ACM International Conference on Distributed and Event-Based Systems*, DEBS '19, p. 163–174, New York, NY, USA. Association for Computing Machinery.

Przymusinski, T. (1987). On the declarate semantics of stratified deductive databases and logic programs. In *Foundations of Deductive Databases and Logic Programming*. Morgan.

Ray, C., Dréo, R., Camossi, E., Jousselme, A.-L., & Iphar, C. (2019). Heterogeneous integrated dataset for maritime intelligence, surveillance, and reconnaissance. https://zenodo.org/record/1167595.

Ray, C., Camossi, E., Jousselme, A.-L., Hadzagic, M., Claramunt, C., & Batty, E. (2016). Maritime data preparation and curation, h2020 datacron project deliverable d5.2.. http://datacron-project.eu/.

Ronca, A., Kaminski, M., Grau, B. C., Motik, B., & Horrocks, I. (2018). Stream reasoning in temporal datalog. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pp. 1941–1948.

Santipantakis, G. M., Vlachou, A., Doulkeridis, C., Artikis, A., Kontopoulos, I., & Vouros, G. A. (2018). A stream reasoning system for maritime monitoring. In *25th International Symposium on Temporal Representation and Reasoning, TIME 2018, Warsaw, Poland, October 15-17, 2018*, pp. 20:1–20:17.

Schultz-Møller, N. P., Migliavacca, M., & Pietzuch, P. R. (2009). Distributed complex event processing with query rewriting. In *Proceedings of the Third ACM International Conference on Distributed Event-Based Systems, DEBS 2009, Nashville, Tennessee, USA, July 6-9, 2009.*

Tsilionis, E., Artikis, A., & Paliouras, G. (2019a). Incremental event calculus for run-time reasoning. In *Proceedings of the 13th ACM International Conference on Distributed and Event-based Systems*, DEBS '19, pp. 79–90, New York, NY, USA. ACM.

Tsilionis, E., Koutroumanis, N., Nikitopoulos, P., Doulkeridis, C., & Artikis, A. (2019b). Online event recognition from moving vehicles: Application paper. *Theory Pract. Log. Program.*, *19*(5-6), 841–856.

Zhang, H., Diao, Y., & Immerman, N. (2014). On complexity and optimization of expensive queries in complex event processing. In *International Conference on Management of Data, SIGMOD 2014, Snowbird, UT, USA, June 22-27, 2014*, pp. 217–228.