A Stream Reasoning System for Maritime Monitoring

Georgios M. Santipantakis
University of Piraeus, Karaoli and Dimitriou 80, 18534 Piraeus, Greece
gsant@unipi.gr

Akrivi Vlachou
University of Piraeus, Karaoli and Dimitriou 80, 18534 Piraeus, Greece
avlachou@unipi.gr

Christos Doulkeridis
University of Piraeus, Karaoli and Dimitriou 80, 18534 Piraeus, Greece
cdoulk@unipi.gr

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

Ioannis Kontopoulos
Institute of Informatics & Telecommunications, NCSR "Demokritos", Ag. Paraskevi, Greece
ikon@iit.demokritos.gr

George A. Vouros
University of Piraeus, Karaoli and Dimitriou 80, 18534 Piraeus, Greece
georgev@unipi.gr

Abstract

We present a stream reasoning system for monitoring vessel activity in large geographical areas. The system ingests a compressed vessel position stream, and performs online spatio-temporal link discovery to calculate proximity relations between vessels, and topological relations between vessel and static areas. Capitalizing on the discovered relations, a complex activity recognition engine, based on the Event Calculus, performs continuous pattern matching to detect various types of dangerous, suspicious and potentially illegal vessel activity. We evaluate the performance of the system by means of real datasets including kinematic messages from vessels, and demonstrate the effects of the highly efficient spatio-temporal link discovery on performance.

2012 ACM Subject Classification Computing methodologies → Activity recognition and understanding

Keywords and phrases event pattern matching, Event Calculus

Digital Object Identifier 10.4230/LIPIcs.TIME.2018.20

Acknowledgements This work is supported by the datAcron project, which has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 687591 (`http://datacron-project.eu`).

1 Introduction

Nowadays, maritime surveillance systems that keep track of the daily operation of vessel fleets constitute an essential tool for large shipping companies, coast guards, as well as governmental agencies, due to their immense effect on economy and environment [10]. Advances in navigation technology enable the real-time provision of vessel positional information for the benefit of surveillance systems. The Automatic Identification System (AIS)[1], for example, is a tracking system for identifying and locating vessels at sea through data exchange. The acquisition of positional data is achieved either by AIS base stations along coastlines, or even by satellites when out of range of terrestrial networks. As this data is streamed in a maritime surveillance system, several operations need to be performed in real-time, including data integration and complex maritime activity recognition.

We present a monitoring system that exploits spatio-temporal relations between vessels, or vessels and areas of interest (e.g., protected areas), which are calculated online by a dedicated component for spatio-temporal link discovery (stLD). These relations are provided as input to a complex activity recognition component, which is based on the "Event Calculus for Run-Time reasoning" (RTEC) [3]. This is an Event Calculus [12] implementation with various optimization techniques for continuous narrative assimilation on data streams.

We address the problem of online spatio-temporal link discovery over streaming and archival data. This link discovery (LD) problem is challenging because of (a) the streaming nature of data, and (b) the complexity of evaluating spatio-temporal similarity functions to determine the links between spatio-temporal entities. Typically, LD tasks are solved by *filter-and-refine* algorithms that identify a set of candidate entities (for linking) in the *filtering step*, which need to be verified in the *refinement step*. The refinement step is the principal cost factor that determines the overall performance of LD, since it requires the evaluation of distance/similarity functions on complex geometrical entities. For the case of stream to static LD, we propose a filtering technique that improves the efficiency of the filtering step by eliminating entities that cannot be linked, thereby reducing the number of entities that have to be considered during refinement. For the case of streaming data only, we provide an efficient method for streaming LD, identifying proximity relations between moving vessels.

In earlier work, we presented a maritime monitoring system which employed RTEC for complex activity recognition [20]. In that work, RTEC performed spatial calculations to determine whether a vessel is close to a port, or within an area of interest. Furthermore, it approximated crudely the *nearby* relation between vessels by checking whether a pair of vessels is located within the same cell of a grid. In this work, we placed emphasis on the detection of spatial relations and developed a separate component for highly efficient spatio-temporal link discovery. Moreover, RTEC has at its disposal additional spatial relations – whether a vessel is *nearby* some area – and a much more accurate account of proximity between vessels.

To summarise, this paper makes the following contributions:

- We present a stream reasoning system integrating a component for spatio-temporal link discovery (stLD), and a component for recognizing complex activities by means of temporal pattern matching.
- We propose a technique for stream-based LD, which improves the efficiency of filtering, by eliminating entities that cannot be linked, thereby reducing the number of entities that have to be considered during refinement.
- We evaluate the performance of the system by means of real datasets including AIS kinematic messages from vessels.

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

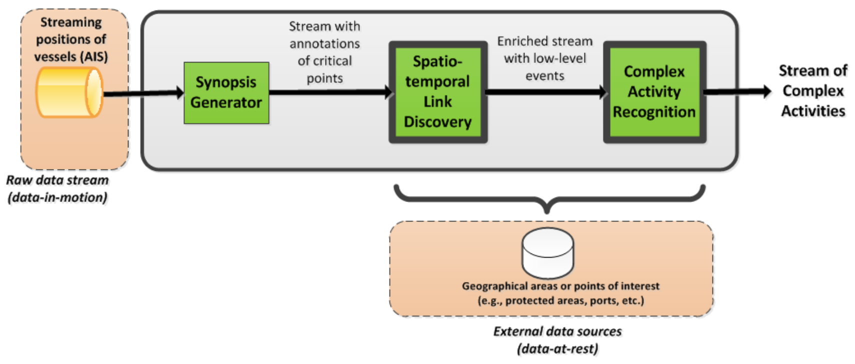

Streaming positions of vessels (AIS)
Raw data stream (data-in-motion)
Synopsis Generator
Stream with annotations of critical points
Spatio-temporal Link Discovery
Enriched stream with low-level events
Complex Activity Recognition
Stream of Complex Activities
Geographical areas or points of interest (e.g., protected areas, ports, etc.)
External data sources (data-at-rest)

Figure 1 The datAcron prototype architecture.

2 System Architecture

Our work is performed in the context of the datAcron research project[2], which aims at advancing the management of voluminous and heterogeneous data-at-rest (archival data) and data-in-motion (streaming data) sources, so as to promote the safety and effectiveness of critical operations of moving objects in large geographical areas. The architecture of the datAcron prototype for complex activity recognition is depicted in Figure 1.

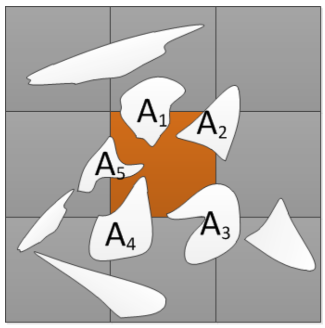
The main input is streaming positions of vessels, in the form of AIS messages. As this streaming data flows in the system, trajectory compression takes place, by annotating a subset of the original vessel positions as *critical points/events*. The *Synopses Generator* component (see Figure 1) provides algorithms for trajectory reconstruction and compression, by cleaning erroneous data and eliminating vessel positions that do not significantly affect the quality of trajectory representation. Then, critical points are linked with archival data online, namely marine areas or points of interest, such as protected areas (Natura2000) or ports (World Port Index). Link discovery is performed at the spatio-temporal level, thus identifying those areas (or points) that are related to a given position of a vessel. Relations may be topological (e.g., a vessel is located *within* an area) or proximity-based (e.g., a vessel is *nearby* a port). In addition, vessel positions are linked to each other, by processing the streaming data only; this results in discovering proximity relations between moving vessels (e.g., a vessel is *nearby* another vessel) in an online fashion. Subsequently, a complex activity recognition module consumes the stream of spatial relations and critical points to recognize various types of vessel activity. The *Complex Activity Recognition* component is based on the “Event Calculus for Run-Time reasoning” (RTEC) [3].

The innovative feature of the proposed architecture is the integration of an optimized component for online discovery of spatial relations with an activity recognition engine. In the following, we delve into the technical details of spatio-temporal link discovery (Section 3) and complex activity recognition (Section 4) for maritime surveillance.

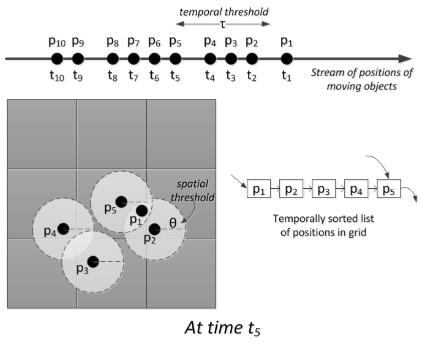
3 Spatio-temporal Link Discovery

We begin by providing the definitions and problem setting for spatio-temporal link discovery. Then, we focus on: (a) topological and proximity relations between the positions of moving entities and static geographical areas of interest (Section 3.1), and (b) proximity relations

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



(a) Illustration of the mask of a grid cell that overlaps five areas $\{A_1, \ldots, A_5\}$.


temporal threshold
τ
p10 p9 p8 p7 p6 p5 p4 p3 p2 p1
t10 t9 t8 t7 t6 t5 t4 t3 t2 t1
Stream of positions of moving objects
spatial threshold
θ
p5 p1 p2 p4 p3
p1 p2 p3 p4 p5
Temporally sorted list of positions in grid
At time t5

(b) Example of grid and bookkeeping structure for "nearby" LD between moving vessels.

Figure 2 Link discovery examples.

between the positions of moving entities (Section 3.2). The former is a case of streaming to static link discovery, while the latter is a case of streaming to streaming link discovery.

Let $p = (p.x, p.y, p.t)$ denote a spatio-temporal point corresponding to a vessel's V position $(p.x, p.y)$ at a given time $p.t$, and $\mathcal{A}$ a dataset that consists of geographical areas represented as polygons. A polygon $A \in \mathcal{A}$ is represented as a set of points a_i, i.e., $A = \{a_1, a_2, \ldots, a_n\}$, and we write $a_i \in A$ to denote that a_i is included in the representation of A.

Further, let $d(p, p')$ denote the distance between the spatial positions $(p.x, p.y)$ and $(p'.x, p'.y)$ of two vessels, whereas $t(p, p')$ stands for their temporal difference. Without loss of generality, in this paper, we employ the Haversine distance function to quantify the spatial distance of two points, whereas $t(p, p') = |p.t - p'.t|$. However, our approach is readily applicable to other domains, where other distance functions (e.g., Euclidean) are more appropriate. We abuse notation slightly by denoting $d(p, A)$ the distance between a point p and an area A (polygon). This is also known as MINDIST, and is defined as the distance of p to the closest point of A (one point of the perimeter), if p is outside of A. Otherwise, $d(p, A) = 0$. Below are the formal descriptions of the spatial relations discovered by the stLD component.

▶ **Definition 1.** $withinArea(V, A)$: Given a spatio-temporal position p of a vessel V and an area $A \in \mathcal{A}$, $withinArea(V, A)$ is true, if p is enclosed in A.

▶ **Definition 2.** $nearbyArea(V, A, \theta)$: Given a spatio-temporal point p of a vessel V, an area $A \in \mathcal{A}$, and a distance threshold θ, $nearbyArea(V, A, \theta)$ is true, if $d(p, A) \leq \theta$.

▶ **Definition 3.** $nearby(V_1, V_2, \theta, \tau)$: Given two spatio-temporal points p and p' of vessels V_1 and V_2 respectively, a distance threshold θ, and a temporal threshold τ, $nearby(V_1, V_2, \theta, \tau)$ is true, if $d(p, p') \leq \theta$ and $t(p, p') \leq \tau$.

3.1 Stream to Static LD

Discovery of Topological Relations: Within. Given a *target* dataset T, a *source* dataset S, and a relation r, the goal of link discovery is to detect the pairs $(\sigma, \tau) \subseteq S \times T$, where $\sigma \in S$ and $\tau \in T$, s.t. (σ, τ) satisfies r. Consider the case of relation "within" between a moving entity p, whose current spatio-temporal position is streamed into our system, and a set of static geographical areas of interest $\mathcal{A} = \{A_1, \ldots, A_n\}$. The aim of link discovery is to identify all areas $A_i \in \mathcal{A}$ which enclose the spatial position $(p.x, p.y)$ of p.

Algorithm 1 Spatio-temporal LD algorithm for relation *"within"* using mask.

```
1: Input: Grid cells $C = \{c_1, \ldots, c_m\}$, Areas $\mathcal{A} = \{A_1, \ldots, A_n\}$, position $p$ $(p.x, p.y)$
2: Output: Subset of areas $\mathcal{A}^w \subseteq \mathcal{A}$ that enclose the position $(p.x, p.y)$ of $p$
3: Requires: Grid has been constructed and areas have been assigned to overlapping cells
4: $\mathcal{A}^w \leftarrow \emptyset$
5: locate cell $c_i$ that encloses $p$
6: if within($p$, $mask(c_i)$) then
7:    return $\mathcal{A}^w$
8: else
9:    for each $A_j \in c_i$ do
10:       if within($p$, $A_j$) then
11:          $\mathcal{A}^w \leftarrow \mathcal{A}^w \cup A_j$
12: return $\mathcal{A}^w$
```

A brute force link discovery algorithm would have to perform the geometrical test between p and all areas in $\mathcal{A}$, thereby performing $O(n)$ comparisons, where $n = |\mathcal{A}|$. To avoid this prohibitively expensive cost, the state-of-the-art LD methods (e.g., [16, 22]) employ a *space tiling* approach, which essentially partitions the space in cells and assigns each area to its overlapping cells. Then, to compute the relation "within", the position of a vessel is compared only against those (say c) areas overlapping the cell, thus resulting in $O(c)$ comparisons, and typically $c \ll n$. Practically, this method belongs to the filter-and-refine paradigm, where in the filtering step only a small set of c (out of n) candidate areas are identified, and in the refinement step the candidates are examined one-by-one to discover the subset of areas actually enclosing the vessel. Since the refinement step is the most costly processing part, any efficient link discovery algorithm should minimize the number of candidates.

In several applications of spatial link discovery, such as the maritime domain, grid cells contain a significant amount of "empty space", namely the space of the cell that overlaps with no areas. Our observation is that positions of moving vessels located in the empty space of a cell induce high processing cost, as they must be compared to all areas in the cell in vain, since no "within" links can be produced. Motivated by this observation, we propose a technique to explicitly represent the empty space within cells as yet another area. Thus, for each grid cell, we construct an artificial area called *mask*, which is defined as the difference between the cell and the union of areas overlapping with the cell, i.e. $mask = c - (c \cap \bigcup(A)_i)$. Notice that the intersection with c before computing the set difference is only used to cover the case where areas are larger the cells. Figure 2a shows an example of the mask of a cell; the middle cell overlaps with areas $\{A_1, \ldots, A_5\}$, and the mask of the cell is the area represented in orange color.

Having the mask of a cell as yet another area, we can devise an efficient algorithm for link discovery that eagerly avoids comparisons to areas for positions located in the empty space. In practice, after we identify the enclosing cell of a position of a vessel, we first compare it to the mask of the cell, to check if it is enclosed in the empty space. If this single comparison returns true, we stop processing this position, thereby saving c comparisons (c denotes the average number of areas in a cell). For the typical case where a cell contains several areas, this technique can save significant computational cost, as will be demonstrated in the empirical analysis presented in Section 5.

Algorithm 1 presents the pseudo-code for discovering a "within" link between the position p of a vessel and the areas that enclose it. As a prerequisite, the grid has already been constructed and the static areas in the dataset R have been assigned to cells. This is a pre-processing step. In the first step, the cell c_i that encloses p is determined (line 5). This operation is performed in constant time $O(1)$ in the case of equi-grids. Then, we check if p is contained in the mask $mask(c_i)$ of cell c_i (line 6). If it is contained, no further processing is required, and the algorithm terminates returning the empty set. If it is not contained, then we check for containment against all areas A_j in cell c_i (line 9). For those areas A_j that contain p, we add them to the result set $\mathcal{A}^w$ (line 11) and eventually return them as result.

The lines 9–11 of Algorithm 1 are processed in parallel, i.e., each iteration in the for-loop is carried out by a different thread/"worker". The number of concurrent workers is usually a predefined constant to allow uninterruptible system operation (in our experiments we employ 8 workers). We have enabled multi-thread processing using a *pool of tasks*, populated with the refinement tasks of $within(p, A_j)$. As soon as a worker is available and the pool contains tasks, the next task is selected and assigned to the worker for processing. Also, the algorithm is amenable to parallelization beyond the scope of a single machine, by simply partitioning the stream of positions of vessels to the available processing nodes, each of which runs an instance of Algorithm 1 on an off-line constructed grid.

Discovery of Proximity Relations: Nearby. The above technique is applicable also for link discovery of a proximity relation, such as the "nearby" relation, between vessel position p and a set of static areas $\mathcal{A}$. The "nearby" relation is defined using a spatial threshold θ, and retrieves the subset of areas in $\mathcal{A}$ that are located at most at distance θ from p.

The technique of using the mask needs to be slightly adjusted to work in the case of relation "nearby". The main adjustment concerns the way the mask of each cell is computed. We expand each area A_i by θ and then the cell's mask is computed as previously, only using the expanded areas (A_i^θ) instead of the actual areas A_i. To differentiate this mask of a cell c_i from the one used in the previous algorithm, we denote it by $mask^\theta(c_i)$.

The LD algorithm for relation "nearby" operates as follows. The grid is constructed exactly as before, using the original areas $\{A_i\}$. First, the cell c_i that encloses p is located. If $mask^\theta(c_i)$ contains p, then we can safely stop processing, since no area is nearby p. This is because $mask^\theta(c_i)$ has been constructed based on expanded areas. If this pruning is not successful, we need to examine all cells $c_i \in C'$ that overlap with a circle centered at p with radius θ, since they may contain results. We examine each area A_j in c_i, and if the distance of p to A_j is lower or equal to the threshold θ, then A_j is added to the result $\mathcal{A}^n$. To avoid computing the distance of p to an area A_j multiple times (due to A_j assignment to multiple cells), we maintain the already examined areas in a set ($\mathcal{P}$). In summary, the algorithm avoids processing points that would safely produce no results, due to the use of the mask technique.

3.2 Stream to Stream LD

Streaming link discovery of "nearby" relations between positions of moving vessels requires meticulous use of the available memory, since it is not feasible to store the complete data stream in memory. Our solution relies on the use of a grid data structure on the spatial domain, similarly to the case of Section 3.1. The grid is used to maintain the positions arriving in the stream. When a new vessel position p arrives in the stream, in the filtering step, we examine the cells that intersect with a circle centered at $(p.x, p.y)$ with radius θ, and retrieve all vessel positions p'_i that satisfy the spatial constraint, i.e., $d(p, p'_i) \le \theta$. Then, in the refinement step, we identify the subset of vessel positions $\{p'_i\}$ that in addition satisfy the temporal constraint, i.e., $t(p, p'_i) \le \tau$. This solution avoids the exhaustive comparison of p to all other positions that have arrived before p. Algorithm 2 is invoked for each incoming p and describes this solution.

Algorithm 2 Spatio-temporal LD algorithm for relation *"nearby"* between vessels.

```
1: Input: Grid cells $C = \{c_1, \dots, c_m\}$, vessel position $p$ $(p.x, p.y)$, thresholds $\theta$, $\tau$
2: Output: Set $R$ of pairs of vessel positions $(p, p')$ satisfying Definition 3
3: $R \leftarrow \emptyset$
4: locate cells $C$ that overlap with circle at $p$ with radius $\theta$
5: for each $c_j \in C$ do
6:    for each $p'_i \in c_j$ do
7:       if $\mathrm{d}(p, p'_i) \leq \theta$ then
8:          if $\mathrm{t}(p, p'_i) \leq \tau$ then
9:             $R \leftarrow R \cup (p, p'_i)$
10: add $p$ to grid $C$
11: return $R$
```

However, in order to manage the available memory effectively, we need to find a suitable way to clean up the grid, since vessel positions that have arrived before more than τ time units will never produce a "nearby" link to a new vessel position. A second reason to perform this cleaning operation is efficiency. If no cleaning were performed, then too many (old) vessel positions would be retrieved that satisfy the spatial constraint, but would be eliminated due to the temporal constraint, leading to wasteful processing.

A naive approach for cleaning would be to scan all grid cells (set at time t_{now}) and delete vessel positions p whose timestamp $p.t$ is more than τ units ago, i.e., $t_{now} - p.t > \tau$. However, this operation has linear complexity wrt. the number of positions in the grid, and incurs non-negligible overhead, as it must be performed during stream processing. To address this problem, we introduce an auxiliary, bookkeeping data structure that efficiently detects the vessel positions that need to be deleted. For this purpose, we maintain a list of pointers to the vessel positions in the grid. The list is in temporal order, since vessel positions are inserted in the list in the order in which they arrive in the stream. Then, cleaning can be performed efficiently, by traversing the list and deleting vessel positions (from the grid and list) until a position p is found with timestamp $t_{now} - p.t \leq \tau$. The maintenance cost of this list is small; insertions have $O(1)$ cost, whereas deletions have linear cost to the number of vessel positions that need to be deleted.

Figure 2b shows an example of the bookkeeping structure. Assuming that $t_{now} = t_5$, then the grid and the list contain the five vessel positions $p_1, \dots, p_5$, as depicted. In case no cleaning is performed, then p_1 would be identified as candidate for "nearby" to p_5, which would then be rejected due to the temporal threshold, since $p_5.t - p_1.t > \tau$. In case cleaning has already been performed, p_1 would have already been deleted from the grid, thus we would have avoided the cost of examining p_1.

An interesting issue that deserves further discussion is the frequency of performing the cleaning operation. An *eager* strategy could invoke the cleaning operation prior to processing every new vessel position that arrives in the stream. This would guarantee that all vessel positions satisfying the spatial distance threshold would be results, without the need to check the temporal threshold. However, the eager strategy would result in paying the overhead of grid cleanup for every new position. Another approach is to follow a *lazy* strategy, where the cleaning operation is invoked periodically, thus limiting the induced overhead at the expense of retrieving some candidate vessel positions that may be rejected by the temporal threshold. We evaluate the frequency of performing the grid cleanup in Section 5 presenting the empirical evaluation.

Table 1 Complex Activity Recognition: Input events are presented above the two horizontal lines, while the output stream is presented below these lines. The input events above the single horizontal line are detected by stLD, while the remaining input events are emitted by the trajectory compression component. All input events, except *nearBy*(V_1, V_2), are instantaneous, while all output activities are durative.

Event/Activity	Description
withinArea(V, A)	A vessel V is within some area A of interest
nearbyArea(V, A)	A vessel V is close to some area A of interest
nearPorts(V)	A vessel V is close to one or more ports
nearby(V_1, V_2)	Two vessels V_1 and V_2 are close to each other
gapStart(V)	A vessel V stopped sending position signals
gapEnd(V)	A vessel V resumed sending position signals
slowMotionStart(V)	A vessel started moving at a low speed
slowMotionEnd(V)	A vessel stopped moving at a low speed
stopStart(V)	A vessel started being idle
stopEnd(V)	A vessel stopped being idle
changeInSpeedStart(V)	A vessel started changing its speed
changeInSpeedEnd(V)	A vessel stopped changing its speed
changeInHeading(V)	A vessel changed its heading
gap(V)	A vessel V has a communication gap in the open sea
stopped(V)	A vessel V is stopped in the open sea
slowMotion(V)	A vessel V moves slowly in the open sea
illegalFishing(V)	A vessel V is potentially engaged in illegal fishing
loitering(V)	A vessel V is suspiciously idle
rendezVous(V_1, V_2)	Two vessels V_1 and V_2 are having a rendez-vous
highSpeedIn(V, *AreaType*)	A vessel V exceeds the speed limit of an area of *AreaType*

4 Complex Activity Recognition

The complex activity recognition (CAR) component of the datAcron prototype consumes the stream of spatial relations detected by stLD, as well as the critical points expressing compressed trajectories, to detect various types of vessel activity (see Section 2). The upper part (above the two horizontal lines) of Table 1 presents the input to the CAR component. The output of this component, i.e. the list of recognized activities, specified in collaboration with the domain experts of datAcron, is presented in the lower part of Table 1.

4.1 Run-Time Event Calculus

The CAR component of datAcron is based on the "Event Calculus for Run-Time reasoning" (RTEC) [3]. RTEC is a Prolog implementation of the Event Calculus [12], designed to compute continuous narrative assimilation queries for pattern matching on data streams. RTEC has a formal, declarative semantics – complex (vessel) activity formalisations are (locally) stratified logic programs [21]. Moreover, RTEC includes various optimisation techniques for efficient pattern matching, such as "windowing", whereby all input events that took place prior to the current window are discarded/"forgotten". Details about the reasoning algorithms of RTEC, including a complexity analysis, may be found in [3]. In what follows, we illustrate the use of RTEC for specifying maritime activities, focusing on the effects of stLD on CAR.

Table 2 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 the maximal intervals for which $F = V$ holds continuously
initiatedAt$(F = V, T)$	At time T a period of time for which $F = V$ is initiated
terminatedAt$(F = V, T)$	At time T a period of time for which $F = V$ is terminated
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
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

The time model in RTEC is linear and includes integer time-points. Variables start with an upper-case letter, while predicates and constants start with a lower-case letter. Where 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. An *event description* in RTEC includes rules that define the event instances with the use of the happensAt predicate, the effects of events on fluents with the use of the initiatedAt and terminatedAt predicates, and the values of the fluents with the use of the holdsAt and holdsFor predicates. Table 2 summarizes the main predicates of RTEC. Complex activities are typically durative (see the lower part of Table 2), thus in CAR the task generally is to compute the maximal intervals for which a fluent expressing a complex activity has a particular value continuously. Below, we discuss the representation of fluents/complex maritime activities, and briefly present the way we compute their maximal intervals.

4.2 Maritime Pattern Representation

For a fluent F, $F = V$ holds at a particular time-point T if $F = V$ has been initiated by an event at some time-point earlier than T, and has not been terminated at some other time-point in the meantime. This is an implementation of the law of *inertia*. The time-points at which $F = V$ is initiated (terminated) are computed with the use of initiatedAt (terminatedAt) rules. Consider the following example:

$$\begin{aligned}
&\mathsf{initiatedAt}(\mathit{gap}(V) = \mathsf{true},\ T) \leftarrow \\
&\qquad \mathsf{happensAt}(\mathit{gapStart}(V),\ T),\ \ \mathsf{not\ happensAt}(\mathit{nearPorts}(V),\ T) \\
&\mathsf{terminatedAt}(\mathit{gap}(V) = \mathsf{true},\ T) \leftarrow \\
&\qquad \mathsf{happensAt}(\mathit{gapEnd}(V),\ T)
\end{aligned} \tag{1}$$

gap(V) is a Boolean fluent denoting a communication gap for some vessel V, i.e. V stops transmitting AIS messages in the open sea. In some cases, the absence of AIS messages is suspicious and thus we need to record it. *gapStart*(V) and *gapEnd*(V) are instantaneous critical events indicating, respectively, the time-points in which a vessel V stops and resumes sending AIS messages (see Table 1). "not" is negation by failure. *nearPorts*(V) is an event emitted by stLD when vessel V is close to a port. Thus, rule-set (1) states that *gap*$(V) =$ true is initiated if the trajectory compression component reports a *gapStart* event for V, and stLD does not report that V is close to a port. Furthermore, $gap(V) = \mathsf{true}$ is terminated when V resumes communications. Given rule-set (1), RTEC computes the maximal intervals I for which $gap(V) = \mathsf{true}$ holds continuously, i.e. $\mathsf{holdsFor}(gap(V) = \mathsf{true},\ I)$, by finding all time-points T_s at which $gap(V) = \mathsf{true}$ is initiated, and then, for each T_s, computing the first time-point T_e after T_s at which $gap(V) = \mathsf{true}$ is terminated.

Most of the maritime patterns (defined in collaboration with domain experts) concern vessel activity close to, or within, some area of interest, such a Natura area. To simplify the structure of such patterns, we defined the auxiliary fluent *inArea* as follows:

$$\begin{aligned}
&\mathsf{initiatedAt}(inArea(V, protected) = \mathsf{true},\ T) \leftarrow \\
&\qquad \mathsf{happensAt}(withinArea(V, A),\ T),\ \ protected(A) \\
&\mathsf{terminatedAt}(inArea(V, protected) = \mathsf{true},\ T) \leftarrow \\
&\qquad \mathsf{happensAt}(nearbyArea(V, A),\ T),\ \ protected(A) \\
&\mathsf{terminatedAt}(inArea(V, protected) = \mathsf{true},\ T) \leftarrow \\
&\qquad \mathsf{happensAt}(\mathsf{start}(gap(V) = \mathsf{true}),\ T)
\end{aligned} \tag{2}$$

$inArea(V, AreaType)$ records the maximal intervals in which a vessel V is in an area of some type. $withinArea(V, A)$ and $nearbyArea(V, A)$ are spatial events emitted by stLD, stating, respectively, that V is within, or close to area A. $protected(A)$ is an atemporal predicate that stores protected areas. $\mathsf{start}(F = V)$ (respectively $\mathsf{end}(F = V)$) is a built-in RTEC event taking place at each starting (ending) point of each maximal interval for which $F = V$ holds continuously. According to rule-set (2), $inArea(V, protected) = \mathsf{true}$ is initiated when stLD detects a $withinArea(V, A)$ event for vessel V and protected area A. Furthermore, $inArea(V, protected) = \mathsf{true}$ is terminated when there is information that V has exited the area, i.e. when a $nearbyArea(V, A)$ event is emitted for the protected area A, or when V stops transmitting position signals (in this case we do not know the location of V).

Similar rule-sets define *inArea* for other types of area. Moreover, *inArea* is represented as a Boolean fluent since areas of different type may overlap, and thus a vessel may be within various types of area at the same time.

In previous work, RTEC performed both temporal and atemporal reasoning for CAR [20]. For example, RTEC performed spatial calculations to determine whether a vessel is within some area of interest or close to a port. In this work, all spatial relations necessary for CAR are computed by stLD. Thus, the evaluation of the initiatedAt rule of rule-set (2), for example, is reduced to checking for (*withinArea*) facts (in the input stream). This way, RTEC is used only for temporal reasoning for which it is optimized.

The absence of the *nearbyArea* relation in earlier work [20] did not allow us to compute the *intervals* during which a vessel is in some area. These intervals allow us to develop more accurate patterns of maritime activity – consider e.g. illegal fishing:

$$\begin{aligned}
&\mathsf{holdsFor}(illegalFishing(V) = \mathsf{true},\ I) \leftarrow \\
&\qquad fishing(V),\ \ \mathsf{holdsFor}(slowMotion(V) = \mathsf{true},\ I_1), \\
&\qquad \mathsf{holdsFor}(inArea(V, protected) = \mathsf{true},\ I_2),\ \ \mathsf{intersect\_all}([I_1, I_2],\ I)
\end{aligned} \tag{3}$$

In addition to the domain-independent definition of holdsFor, an event description may include domain-specific holdsFor rules, such as rule (3), used to define the values of a fluent F, e.g. *illegalFishing*, in terms of a Boolean function on the values of other fluents. Domain-specific holdsFor rules make use of interval manipulation constructs: union_all, intersect_all, and relative_complement_all. These are presented in Table 2. *fishing* is an atemporal predicate recording fishing vessels. According to rule (3), the list I of maximal intervals during which a fishing vessel V is said to be engaged in illegal fishing is computed by determining the list I_1 of maximal intervals during which V is moving in slow motion in the open sea (these intervals are computed given the instantaneous *slowMotionStart* and *slowMotionEnd* critical events), the list I_2 of maximal intervals during which V is in a protected area, and then calculating the list I representing the intersections of the maximal intervals in I_1 and I_2.

The interval manipulation constructs of RTEC support the following type of definition: for all time-points T, $F = V$ holds at T if and only if some Boolean combination of fluent-value pairs holds at T. For a wide range of fluents, this is a much more concise definition than the traditional style of Event Calculus representation, i.e. identifying the various conditions under which the fluent is initiated and terminated so that maximal intervals can then be computed using the domain-independent holdsFor. For instance, the representation of *illegalFishing* by means of initiatedAt and terminatedAt predicates requires four rules.

In addition to patterns for individual vessels, the knowledge base of the CAR component of datAcron includes formalizations of activities for pairs of vessels. Consider *rendezVous*:

$$\begin{aligned}
&\mathsf{holdsFor}(\mathit{rendezVous}(V_1, V_2) = \mathsf{true},\ I) \leftarrow \\
&\quad \mathsf{holdsFor}(\mathit{nearby}(V_1, V_2) = \mathsf{true},\ I_1),\ \ \mathsf{holdsFor}(\mathit{slowMotion}(V_1) = \mathsf{true},\ I_2), \\
&\quad \mathsf{holdsFor}(\mathit{stopped}(V_1) = \mathsf{true},\ I_3),\ \ \mathsf{union\_all}([I_2, I_3],\ I_4), \\
&\quad \mathsf{holdsFor}(\mathit{slowMotion}(V_2) = \mathsf{true},\ I_5),\ \ \mathsf{holdsFor}(\mathit{stopped}(V_2) = \mathsf{true},\ I_6), \\
&\quad \mathsf{union\_all}([I_5, I_6],\ I_7),\ \ \mathsf{intersect\_all}([I_1, I_4, I_7],\ I)
\end{aligned} \tag{4}$$

$\mathit{nearby}(V_1, V_2) = \mathsf{true}$ is a fluent denoting whether two vessels V_1 and V_2 are close to each other. This relation is computed by stLD. $\mathit{stopped}(V) = \mathsf{true}$ expresses that vessel V has stopped in the open sea. The intervals of this fluent-value pair are computed with the use of the instantaneous *stopStart* and *stopEnd* critical events (see Table 1). According to rule (4), vessels V_1 and V_2 are said to have a rendez-vous when they are close to each other, and each of them is stopped or moving in slow motion in the open sea. In previous work, we approximated very crudely the $\mathit{nearby}(V_1, V_2)$ relation by checking whether V_1 and V_2 are located within the same cell of a grid. This approximation produced several false negatives (vessels located close to each other but in different cells) as well as false positives (vessels located within the same cell but not close to each other). In this work, we can avoid these issues, due to the efficient spatial relation detection of stLD.

5 Experimental Evaluation

We present the results of our experimental study using real maritime datasets.

5.1 Experimental Setup

Platform. All experiments presented below were performed on a computer with 8 cores (Intel(R) Core(TM) i7-7700 CPU @ 3.6GHz) and 16 GB of RAM, running Ubuntu 16.04 LTS 64-bit with Linux Kernel 4.8.0-53-generic, Java 8, and SWI Prolog 7.2.3 for RTEC.

Datasets. Our main dataset contains AIS kinematic messages from vessels sailing in the Atlantic Ocean around Brest, France, spanning between 1 October 2015 to 31 March 2016 (`https://zenodo.org/record/1167595#.WwlWjp99LJ9`). AIS messages from 5,050 vessels are compressed by the trajectory synopsis module (see Figure 1), keeping only critical points (see Table 1). Each critical point is accompanied by mobility information such as speed and heading. The dataset contains 4,765,647 critical points. These are consumed by stLD which produces 3,204,206 spatial events that are additionally provided to CAR as input. Spatial events determine whether a vessel is in or near an area of interest, and if two vessels are close to each other in space and time.

Table 3 Distribution of areas from target dataset for each grid configuration.

Granularity	Average Number of Areas in Cell	Cells Constructed
0.5°	5.74	2,852
1°	13.54	858
2°	36.09	272
3°	64.14	147

We also employ a set of spatial datasets describing areas and points of interest, as follows:

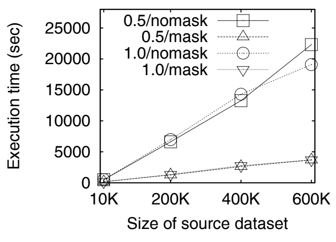
- Natura2000 regions: Natura 2000 is an index of protected regions for biodiversity in the European Union. It is set up to ensure the survival of Europe's most valuable species and habitats, indexing 3,522 polygons.
- Fishing areas: This dataset includes 5,076 polygons generated from raster images depicting the fishing intensity in European waters (as reported by European Union).
- The World Port Index (WPI) including a listing of 3,685 ports throughout the world, describing their location, characteristics, known facilities, and available services. Ports in this dataset are represented as points, and WPI is used for discovering proximity relations between vessel positions and ports.

Metrics. Our main metric is the achieved throughput of the stream reasoning system, assessed by delving into the individual performance of its two constituent components, stLD and CER. Other auxiliary metrics are additionally presented, in order to explain performance, such as processing time and recognition time for stLD and complex activity recognition respectively. We also measure the number of *unrewarding comparisons* of each LD configuration, i.e., the number of comparisons that did not result in a relation.

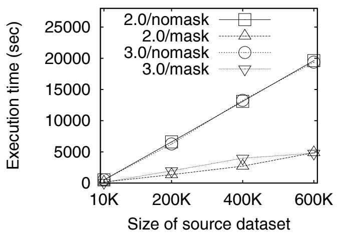
Parameters. In the experiments evaluating the performance of link discovery, we vary the input size and granularity of the grid (i.e., cell size). In the case of stream to static LD (i.e., "within" and "nearby" relations between moving objects and regions), we setup multiple equi-grid configurations varying in granularity {0.5°, 1°, 2°, 3°} (degrees in latitude/longitude), using the datasets of Natura2000 and fishing regions, totaling 8,599 regions. An equi-grid of 0.5° granularity means that the size of a cell is $0.5° \times 0.5°$. Notice that we construct the grid over the complete static datasets, as typically done in link discovery tasks. We evaluate the performance and scalability of stLD, using subsets of critical points datasets, i.e., {10K, 200K, 400K, 600K} entries. Also, we evaluate the gain obtained by using the mask technique (for the stream to static link discovery method of Section 3.1), and using the bookkeeping technique (for the stream to stream link discovery method of Section 3.2). All topological relations (such as within, but also those required for computing the mask) are provided by the Java Topology Suite (JTS) v.1.13. Finally, we evaluate the effect of varying the number of cores on CAR, as well as the effect of increasing the window size.

5.2 Link Discovery

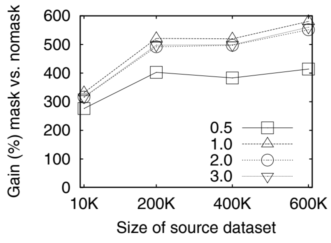
Stream to Static LD. We refer to the areas organized in a grid as *target* dataset, whereas *source* dataset refers to the dataset of streaming vessel positions. We use as *target* dataset the regions of Natura2000 and fishing areas, totaling 8,599 areas. This static data is organized off-line in main-memory using grid configurations of varying granularity for the complete geographical space covered by the areas. For the streaming *source* dataset, we employ subsets



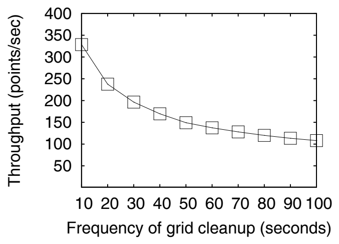
(a) Total execution time.



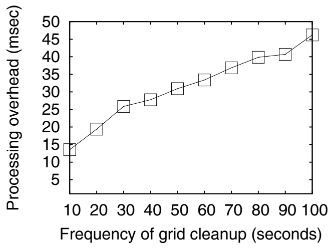
(b) Total execution time.



(c) Gain (%) of using mask.



(d) Throughput.



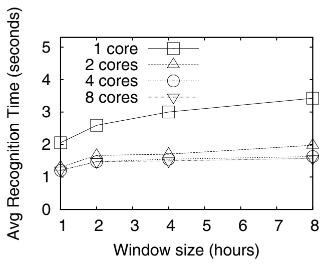
(e) Overhead of grid cleanup.

Figure 3 Spatio-temporal LD. Stream to static: **(a)**–**(c)**; Stream to stream: **(d)** and **(e)**.

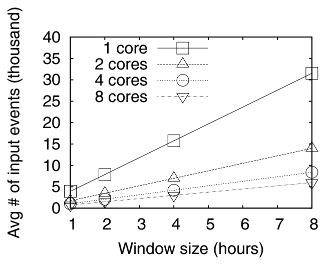
of the critical points dataset of different sizes, in order to evaluate its effect. The distribution of areas for each grid configuration is shown in Table 3. The second column shows the average number of areas in a cell, while the third column indicates the number of non-empty cells that were constructed (i.e., hold at least one of the given areas).

Figures 3a–3c show the benefits of using the proposed technique with mask. Figure 3a depicts the total execution time required for processing a subset of critical points, whose size is indicated in the x-axis, using two grids with granularity 0.5° and 1.0° respectively. We observe that the mask technique achieves better execution time for all grid configurations and input sizes. Figure 3b shows both grid configurations of granularity 2.0° and 3.0°. In terms of throughput, the use of the mask achieves to increase the throughput up to a factor of 5, compared to not using the mask. Figure 3c quantifies this gain, as the ratio of unrewarding comparisons, i.e., the number of unrewarding comparisons without mask, divided by the number of unrewarding comparisons with mask. For instance, the ratio of unrewarding comparisons for input size 200K and granularity 0.5° is 402.7%. This result clearly demonstrates the advantage of using the mask technique.

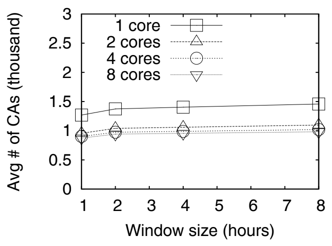
Stream to Stream LD. We evaluate the effect of the grid cleaning technique on throughput, using a temporal threshold of 30 seconds. Figure 3d shows how throughput is affected when varying the frequency of grid cleanup from 10–100 sec. The chart shows that the more frequent cleanup is performed, the higher the achieved throughput. When the cleanup is delayed, the processing time spent on comparisons is increased, affecting the total execution time, and decreasing the average throughput. The next question to be answered concerns the overhead imposed by frequent grid cleanup. Figure 3e shows the average time spent for grid cleanup in milliseconds. We observe that the overhead is very small, and increases when the cleaning is not performed regularly. Recall that the bookkeeping structure is a list of spatio-temporal positions of vessels ordered by their temporal values, and the disposal of part of the list on each call allows us to avoid searching every cell of the grid for "expired" entries. In summary, the cleanup operation has minimal overhead, which makes it applicable with high frequency, thereby increasing the achieved throughput.



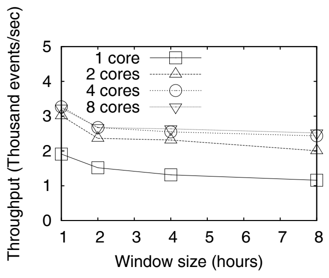
(a) Average recognition time.



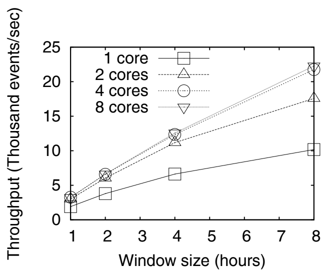
(b) Average number of input events.



(c) Average number of recognised activities.



(d) Throughput for slide step $\beta = 1$ hour.



(e) Throughput for slide step $\beta = \omega$.

Figure 4 Complex activity recognition.

5.3 Complex Activity Recognition

RTEC performs continuous query processing, computing the maximal intervals of fluents representing complex maritime activities. At each query time Q_i, the input events that fall within a specified sliding window ω are taken into consideration. All input events that took place before or at $Q_i{-}\omega$ are discarded/"forgotten". This way, the cost of reasoning is independent of the data stream size. At Q_i, the complex activity intervals computed by RTEC are those that can be derived from input events that occurred in the interval $(Q_i{-}\omega, Q_i]$, as recorded at time Q_i. When the range ω is longer than the slide step β, it is possible that an input event occurs in the interval $(Q_i{-}\omega, Q_{i-1}]$ but arrives at RTEC only after Q_{i-1}; its effects are taken into account at query time Q_i. Window parameters for ω and slide step β are defined by the domain experts along with the maritime patterns, since they reflect the time horizon over which interesting phenomena may be detected. Usually, ω spans many minutes or even hours for capturing meaningful events across a vessel's route.

Figure 4 presents the experimental results for increasing window sizes ω: 1 hour to 8 hours. Figure 4a presents the average recognition time per window ω, while Figures 4b and 4c show, respectively, the average number of input events and recognized activities per window. Recall that Table 1 lists the types of input event and recognized activity. The slide step β is set to 1 hour. The empirical analysis shows that RTEC is capable of real-time maritime activity recognition – e.g. RTEC recognizes ($\approx$ 1,500) maritime activities given a window of 8 hours ($\approx$ 31,000 input events) in less than 3.5 sec, even when operating on a single processing core of the desktop computer. (The use of multiple cores will be discussed shortly.) Figure 4d presents the throughput. As the window size ω increases, throughput decreases since the average recognition time per window increases (see Figure 4a). Figure 4e presents the throughput when the slide step is the same as the window size ($\beta = \omega$), i.e. windows are non-overlapping. In this case, the average recognition time per window increases, but only very slightly (due to the higher cost of “forgetting” more past events), and thus not presented here. (The average numbers of input events and recognized activities per window are also similar to the case of $\beta = 1$ hour). Figure 4e shows that, as the window size ω increases, throughput increases. When ω increases, the number of windows/query-times decreases (we have significantly less windows than the case of $\beta = 1$ hour), while the average recognition time per window increases only slightly.

We also run RTEC in parallel, by launching different instances of the engine, each operating on a different processing core. Each RTEC instance was responsible for activity recognition in a separate sub-area of the dataset, receiving input events only from vessels in that sub-area. To avoid false negatives on the borders of the sub-areas, we allowed for some overlap. Figure 4 presents the results when 2, 4 and 8 processing cores are used. Figure 4a presents the average recognition of the worst-performing RTEC instance, while Figures 4b and 4c show the average input events and recognised activities for that instance. Figures 4d and 4e present the throughput. As shown in Figure 4, the benefits of parallelization can be significant. A more refined segmentation of the dataset into sub-areas, optimizing load allocation, would have had more profound effects on performance.

6 Discussion

We presented a stream reasoning system for maritime monitoring, which supports complex activity recognition assisted by online spatio-temporal discovery of relations between vessels and areas of interest. Compared to earlier work, the proposed system optimizes the computation of spatial relations, leading to improved system performance. Our experimental evaluation on real datasets demonstrated the efficiency of the proposed prototype.

Even though the topic of link discovery has attracted much interest lately (see [15] for a recent survey), there is not much work on spatio-temporal link discovery nor on link discovery over streaming data. Our work tackles explicitly these topics. Generic LD frameworks include LIMES [17] and SILK [11]. LIMES [17] is an LD framework for metric spaces that uses the triangular inequality in order to avoid processing all pairs of objects. LIMES employs the concept of exemplars, which are used to represent areas in the multidimensional space, and prunes entire areas (and the respective enclosed entities) during link discovery. SILK [11] is an LD framework proposing a novel blocking method called MultiBlock, which uses a multidimensional index. The spatial LD methods [16, 22] apply grid partitioning (a.k.a. space tiling) on sources A and B, in order to perform efficiently the *filtering step*. Then, in the *refinement step*, different optimizations are employed in order to minimize the number of computations necessary to produce the correct result set. However, all these works focus on topological relations, and it is not straightforward to extend them for proximity relations.

Various languages and systems have been proposed in the literature for complex event/activity recognition [7, 1]. RTEC has a formal, declarative semantics – activity patterns in RTEC are (locally) stratified logic programs. In contrast, most complex event processing languages, event query languages, data stream processing languages and commercial production rule systems, rely on an informal and/or procedural semantics [8]. Furthermore, RTEC explicitly represents complex activity intervals (unlike e.g. [9, 8, 4]) and thus avoids the related logical problems [18]. Maritime activities form hierarchies, in the sense that the formulation of one activity is used to define other, higher-level activities. We defined e.g. potentially illegal fishing in terms of slow motion (recall rule (3)). In contrast to state-of-the-art recognition systems, such as the Esper engine (`http://www.espertech.com/esper/`) and SASE (`http://sase.cs.umass.edu/`), RTEC can naturally express hierarchical knowledge by means of well-structured specifications, and consequently employ caching techniques to avoid unnecessary re-computations [3]. Concerning the Event Calculus literature, a key feature of RTEC is that it includes a windowing technique. No other Event Calculus system (including [6, 5, 13, 19, 2, 14]) "forgets" or represents concisely the data stream history.

References

1 E. Alevizos, A. Skarlatidis, A. Artikis, and G. Paliouras. Probabilistic complex event recognition: A survey. *ACM Comput. Surv.*, 50(5):71:1–71:31, 2017. doi:10.1145/3117809.

2 Alexander Artikis and Marek J. Sergot. Executable specification of open multi-agent systems. *Logic Journal of the IGPL*, 18(1):31–65, 2010.

3 Alexander Artikis, Marek J. Sergot, and Georgios Paliouras. An event calculus for event recognition. *IEEE Trans. Knowl. Data Eng.*, 27(4):895–908, 2015.

4 H. Beck, M. Dao-Tran, and T. Eiter. LARS: A Logic-Based Framework for Analytic Reasoning over Streams. Technical Report INFSYS RR-1843-17-03, Institute of Information Systems, TU Vienna, 2017.

5 Iliano Cervesato and Angelo Montanari. A calculus of macro-events: Progress report. In *Proceedings of TIME*, pages 47–58, 2000.

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

7 G. Cugola and A. Margara. Processing flows of information: From data stream to complex event processing. *ACM Computing Surveys*, 44(3):15, 2012.

8 Gianpaolo Cugola and Alessandro Margara. TESLA: a formally defined event specification language. In *Proceedings of DEBS*, pages 50–61, 2010.

9 C. Dousson and P. Le Maigat. Chronicle recognition improvement using temporal focusing and hierarchisation. In *Proceedings of IJCAI*, pages 324–329, 2007.

10 Bilal Idiri and Aldo Napoli. The automatic identification system of maritime accident risk using rule-based reasoning. In *Proceedings of SoSE*, pages 125–130, 2012.

11 Robert Isele, Anja Jentzsch, and Christian Bizer. Efficient multidimensional blocking for link discovery without losing recall. In *Proceedings of WebDB*, 2011.

12 Robert A. Kowalski and Marek J. Sergot. A logic-based calculus of events. *New Generation Comput.*, 4(1):67–95, 1986.

13 R. Miller and M. Shanahan. Some alternative formulations of the event calculus. In *Computational Logic: Logic Programming and Beyond*, LNAI 2408, pages 452–490. Springer, 2002.

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

15 Markus Nentwig, Michael Hartung, Axel-Cyrille Ngonga Ngomo, and Erhard Rahm. A survey of current link discovery frameworks. *Semantic Web*, 8(3):419–436, 2017.

16 Axel-Cyrille Ngonga Ngomo. ORCHID - reduction-ratio-optimal computation of geo-spatial distances for link discovery. In *Proceedings of ISWC*, pages 395–410, 2013.

17 Axel-Cyrille Ngonga Ngomo and Sören Auer. LIMES - A time-efficient approach for large-scale link discovery on the web of data. In *Proceedings of IJCAI*, pages 2312–2317, 2011.

18 A. Paschke. ECA-RuleML: An approach combining ECA rules with temporal interval-based KR event/action logics and transactional update logics. Technical Report 11, Technische Universität München, 2005.

19 Adrian Paschke and Martin Bichler. Knowledge representation concepts for automated SLA management. *Decision Support Systems*, 46(1), 2008.

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

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

22 Mohamed Ahmed Sherif, Kevin Dreßler, Panayiotis Smeros, and Axel-Cyrille Ngonga Ngomo. Radon - rapid discovery of topological relations. In *Proceedings of AAAI*, pages 175–181, 2017.