Extending the Range of Temporal Specifications of the Run-Time Event Calculus

Periklis Mantenoglou
National and Kapodistrian University of Athens, Greece
NCSR "Demokritos", Athens, Greece

Alexander Artikis
University of Piraeus, Greece
NCSR "Demokritos", Athens, Greece

Abstract

Composite event recognition (CER) frameworks reason over streams of low-level, symbolic events in order to detect instances of spatio-temporal patterns defining high-level, composite activities. The Event Calculus is a temporal, logical formalism that has been used to define composite activities in CER, while $\text{RTEC}_\circ$ is a formal CER framework that detects composite activities based on their Event Calculus definitions. $\text{RTEC}_\circ$, however, cannot handle every possible set of Event Calculus definitions for composite activities, limiting the range of CER applications supported by $\text{RTEC}_\circ$. We propose $\text{RTEC}_{\mathit{fl}}$, an extension of $\text{RTEC}_\circ$ that supports arbitrary composite activity specifications in the Event Calculus. We present the syntax, semantics, reasoning algorithms and time complexity of $\text{RTEC}_{\mathit{fl}}$. Our analysis demonstrates that $\text{RTEC}_{\mathit{fl}}$ extends the scope of $\text{RTEC}_\circ$, supporting every possible set of Event Calculus definitions for composite activities, while maintaining the high reasoning efficiency of $\text{RTEC}_\circ$.

2012 ACM Subject Classification Computing methodologies → Temporal reasoning

Keywords and phrases Event Calculus, temporal pattern matching, composite event recognition

Digital Object Identifier 10.4230/LIPIcs.TIME.2024.6

Supplementary Material *Software*: `https://github.com/aartikis/rtec`
archived at `swh:1:dir:71c0e451729b42a4ad25f6e6fad998fe5d35adba`

Funding This work was supported by the EU-funded CREXDATA project (No 101092749), and partly supported by the University of Piraeus Research Center.

1 Introduction

Composite event recognition (CER) involves the detection of composite activities by reasoning over streams of time-stamped, symbolic events [16, 20]. A CER framework employs an activity specification language, where it is possible to express the spatio-temporal combinations of input events that form each activity of interest in some application domain. In human activity recognition, e.g., we may specify the time periods during which two people are "gathering" using a pattern stating that at least one of the two people is walking towards the other one, while, at the same time, the distance between them is a few meters and they are facing each other. As another example, in the task of monitoring composite maritime activities, we may define "trawling", i.e., a type of fishing activity that involves several consecutive turns, as a sequence of "change in heading" events.

The literature contains numerous CER frameworks [1, 20], several of which are automata-based [32, 39, 21]. CORE, e.g., is a formal automata-based CER system that has proven to be more efficient than other contemporary automata-based engines [10]. CORE is restricted to unary relations, while the composite activities derived by CORE cannot be used as building blocks in other patterns. In other words, CORE does not support relational and hierarchical

31st International Symposium on Temporal Representation and Reasoning (TIME 2024).
Editors: Pietro Sala, Michael Sioutis, and Fusheng Wang; Article No. 6; pp. 6:1–6:14
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

composite activity specifications. There are also logic-based CER formalisms [17, 11, 8]. For instance, there are several frameworks supporting fragments of the LARS language [5] that are suitable for CER [6, 4, 18]. MeTeoR is a logic-based CER engine whose language extends DatalogMTL with windowing [37, 38]. The Chronicle Recognition System (CRS) represents composite activities as sets of events that are associated with time constraints [17]. The language of CRS includes several operators, such as sequencing, iteration and negation. These formalisms support relational composite activities, as well as compositional specifications, paving the way for hierarchical definitions. Moreover, logic-based formalisms typically exhibit a formal and declarative semantics, as opposed to automata-based approaches, which do not always come with a clear semantics, making them hard to evaluate and generalise [21].

The Event Calculus is a logic programming formalism for representing and reasoning about events and their effects over time [24]. The Event Calculus may be used as an activity specification language for CER, as it exhibits a formal, declarative semantics, while supporting relational and hierarchical activity specifications that may include background knowledge [27, 20]. Moreover, the Event Calculus includes a built-in representation of inertia, allowing for succinct composite activity patterns, and thus code maintenance. The Event Calculus has been employed in various settings, including mobility assistance [9], reactive and proactive health monitoring [13, 22] and simulations with cognitive agents [34]. The "Macro Event Calculus", e.g., uses "macro-events" to support composite event operators, such as sequence, disjunction, parallelism and iteration [12]. The "Interval-based Event Calculus" incorporates durative events and supports sequencing, concurrency and negation [28]. jREC is a reactive implementation of the Cached Event Calculus [14] which is optimised for CER [7, 19]. The Run-Time Event Calculus (RTEC) extends the Event Calculus with optimisation techniques for CER, such as windowing, indexing and caching [3]. In order to perform CER with minimal latency, RTEC processes hierarchies of composite activity definitions bottom-up, while caching and reusing the derived instances of composite activities, thus avoiding re-computations. RTEC has proven highly efficient in demanding CER applications, including city transport management [3], maritime situational awareness [30] and commercial fleet management [36], outperforming the state-of-the-art [26, 25, 36].

RTEC does not support every possible composite activity definition that may be expressed in the Event Calculus. In human activity recognition, e.g., there is a need to model composite activities defined in terms of the concept "$movement(P_1, P_2)$", expressing the relative movement between persons P_1 and P_2. For instance, "$movement(P_1, P_2) = gathering$" expresses that P_1 and P_2 are moving towards one another in order to have a meeting, and "$movement(P_1, P_2) = abrupt\_gestures$" denotes that, while P_1 and P_2 are talking to each other, one of them is moving his arms abruptly. Furthermore, it may be desirable to express that P_1 and P_2 may be making abrupt gestures to each other only after they have gathered close to one another, i.e., $movement(P_1, P_2) = abrupt\_gestures$ depends on $movement(P_1, P_2) = gathering$. RTEC does not support Event Calculus definitions where composite activities characterised by the same underlying concept, such as $movement(P_1, P_2)$, depend on each other. To address this issue, we propose an extension of RTEC that supports an arbitrary set of Event Calculus definitions.

Our starting point is RTEC$_\circ$, an extension of RTEC that supports Event Calculus definitions with cyclic dependencies, which are often required for CER [26], and propose RTEC$_{fl}$, an extension of RTEC$_\circ$ that supports every possible set of composite activity definitions in the Event Calculus. Our contributions are the following. First, we present the semantics of RTEC$_{fl}$. Second, we present a compiler for RTEC$_{fl}$, identifying the reasoning algorithm that needs to be used at run-time in order to resolve each condition of a composite activity definition. Third, we outline the time complexity of RTEC$_{\mathit{fl}}$, demonstrating that its cost is the same as RTEC$_\circ$, while supporting a wider range of temporal specifications. RTEC$_{\mathit{fl}}$ and its compiler are publicly available[1].

2 Background

Our starting point is RTEC$_\circ$, i.e., a recent extension of the Run-Time Event Calculus (RTEC) that supports efficient reasoning over temporal specifications with cyclic dependencies [26] (the other extensions of RTEC are orthogonal to this work). We present the syntax, semantics and reasoning algorithms of RTEC$_\circ$. In Section 3, we outline the limitations of RTEC$_\circ$, and, in Section 4, we present an extension of RTEC$_\circ$ that supports every set of Event Calculus definitions.

2.1 Syntax & Semantics

The language of RTEC$_\circ$ follows the Event Calculus, which is many-sorted, including sorts for representing time, instantaneous events and "fluents", i.e., properties that may have different values at different points in time. The time model comprises a linear time-line with non-negative integer time-points. $\mathsf{happensAt}(E, T)$ signifies that event E occurs at time-point T. $\mathsf{initiatedAt}(F=V, T)$ (resp. $\mathsf{terminatedAt}(F=V, T)$) expresses that a time period during which a fluent F has the value V continuously is initiated (terminated) at time-point T. $\mathsf{holdsAt}(F=V, T)$ states that F has value V at T, while $\mathsf{holdsFor}(F=V, I)$ expresses that the "fluent-value pair" (FVP) $F{=}V$ holds continuously in the maximal intervals included in list I.

In CER, happensAt is used to express the input events of the stream, while FVPs express composite activities. A formalisation of the activity specification of a domain in the Event Calculus is called *event description*.

▶ **Definition 1** (Event Description). *An event description $\mathcal{E}$ is a set of:*

- *ground $\mathsf{happensAt}(E, T)$ facts, expressing a stream of event instances, and*
- *rules with head $\mathsf{initiatedAt}(F=V, T)$ or $\mathsf{terminatedAt}(F=V, T)$, expressing the effects of events on FVP $F{=}V$.*

▶ **Definition 2** (Syntax of the Rules in the Event Description). *$\mathsf{initiatedAt}(F=V, T)$ rules have the following syntax:*

$$\begin{aligned}&\mathsf{initiatedAt}(F=V,\ T) \leftarrow\\ &\quad \mathsf{happensAt}(E_1,\ T)[[, [\mathsf{not}]\ \mathsf{happensAt}(E_2,\ T),\ \dots, [\mathsf{not}]\ \mathsf{happensAt}(E_n,\ T),\\ &\quad [\mathsf{not}]\ \mathsf{holdsAt}(F_1=V_1,\ T),\ \dots, [\mathsf{not}]\ \mathsf{holdsAt}(F_k=V_k,\ T)]].\end{aligned} \tag{1}$$

The first body literal of an initiatedAt rule is a positive happensAt predicate; this is followed by a possibly empty set, denoted by "[[ ]]", of positive/negative happensAt and holdsAt predicates. "not" expresses negation-by-failure [15], while "[not]" denotes that "not" is optional. All (head and body) predicates are evaluated on the same time-point T. The bodies of $\mathsf{terminatedAt}(F=V, T)$ rules have the same form.

[1] https://github.com/aartikis/RTEC

▶ **Example 3** (Event Description for Human Activity Recognition). In human activity recognition, we apply rules on streams containing symbolic representations of video feeds [2]. In general, such rules are constructed in collaboration with domain experts or learned from data [23]. We use the fluent $interaction(P_1, P_2)$ to express that people P_1 and P_2 are interacting, while the value of $interaction(P_1, P_2)$ denotes the stage of the interaction. The "*greeting*" stage of $interaction(P_1, P_2)$ denotes that P_1 and P_2 are greeting each other at a distance. Below, we outline a set of rules included in the specification of FVP $interaction(P_1, P_2) = greeting$:

$$\begin{aligned}&\mathsf{initiatedAt}(interaction(P_1, P_2) = greeting,\ T) \leftarrow\\ &\quad \mathsf{happensAt}(active(P_1),\ T),\ \mathsf{happensAt}(active(P_2),\ T),\\ &\quad \mathsf{holdsAt}(distance(P_1, P_2) = mid,\ T),\ \mathsf{holdsAt}(orientation(P_1, P_2) = facing,\ T).\end{aligned} \tag{2}$$

$$\begin{aligned}&\mathsf{terminatedAt}(interaction(P_1, P_2) = greeting,\ T) \leftarrow\\ &\quad \mathsf{happensAt}(walking(P_1),\ T),\\ &\quad \mathsf{not}\ \mathsf{holdsAt}(orientation(P_1, P_2) = facing,\ T).\end{aligned} \tag{3}$$

$$\begin{aligned}&\mathsf{terminatedAt}(interaction(P_1, P_2) = greeting,\ T) \leftarrow\\ &\quad \mathsf{happensAt}(walking(P_2),\ T),\\ &\quad \mathsf{not}\ \mathsf{holdsAt}(orientation(P_1, P_2) = facing,\ T).\end{aligned} \tag{4}$$

According to rule (2), P_1 and P_2 start greeting when both of them are "active", i.e., moving their arms while in the same position, the distance between them is a few meters, denoted by the value "mid", and they are facing towards one another. Rules (3)–(4) express that P_1 and P_2 stop greeting when one of them starts walking, while they are not facing each other. The FVPs $distance(P_1, P_2) = mid$ and $orientation(P_1, P_2) = facing$ are defined based on the coordinates and the orientation of the tracked people, which are provided in the input stream.

Moreover, we may use the fluent $movement(P_1, P_2)$ to express the relative movement between people P_1 and P_2 and the value "*gathering*" of $movement(P_1, P_2)$ to denote that P_1 and P_2 are approaching one another. The specification of FVP $movement(P_1, P_2) = gathering$ includes the following rules:

$$\begin{aligned}&\mathsf{initiatedAt}(movement(P_1, P_2) = gathering,\ T) \leftarrow\\ &\quad \mathsf{happensAt}(walking(P_1),\ T),\\ &\quad \mathsf{holdsAt}(distance(P_1, P_2) = mid,\ T),\ \mathsf{holdsAt}(orientation(P_1, P_2) = facing,\ T).\end{aligned} \tag{5}$$

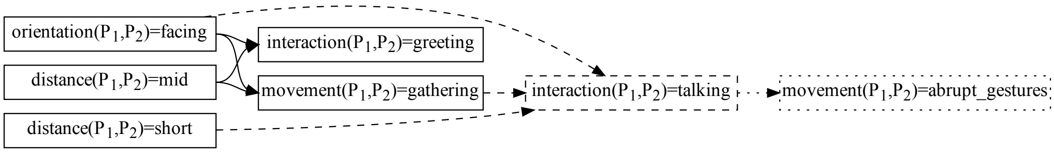
$$\begin{aligned}&\mathsf{initiatedAt}(movement(P_1, P_2) = gathering,\ T) \leftarrow\\ &\quad \mathsf{happensAt}(walking(P_2),\ T),\\ &\quad \mathsf{holdsAt}(distance(P_1, P_2) = mid,\ T),\ \mathsf{holdsAt}(orientation(P_1, P_2) = facing,\ T).\end{aligned} \tag{6}$$

$$\begin{aligned}&\mathsf{terminatedAt}(movement(P_1, P_2) = gathering,\ T) \leftarrow\\ &\quad \mathsf{happensAt}(active(P_1),\ T),\ \mathsf{not}\ \mathsf{happensAt}(walking(P_2),\ T).\end{aligned} \tag{7}$$

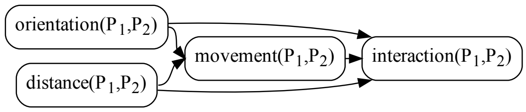
$$\begin{aligned}&\mathsf{terminatedAt}(movement(P_1, P_2) = gathering,\ T) \leftarrow\\ &\quad \mathsf{happensAt}(active(P_2),\ T),\ \mathsf{not}\ \mathsf{happensAt}(walking(P_1),\ T).\end{aligned} \tag{8}$$

Rules (5)–(6) state that P_1 and P_2 start gathering when one of them is walking towards the other person, while their distance is a few meters and they are facing each other. Rules (7)–(8) express that P_1 and P_2 stop gathering when one of them is being active, while the other person is not walking.

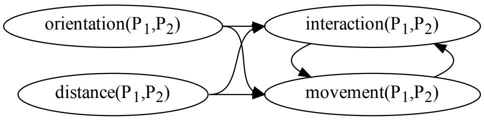
The dependencies among the FVPs in an event description can be expressed in the form of a *dependency graph*.



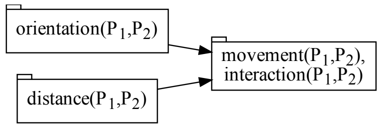
(a) The dependency graph $G_{\mathcal{E}_1}$ of event description $\mathcal{E}_1$ (continuous lines), the dependency graph $G_{\mathcal{E}_2}$ of event description $\mathcal{E}_2$ (continuous and dashed lines), and the dependency graph $G_{\mathcal{E}_3}$ of event description $\mathcal{E}_3$ (all lines). For simplicity, a vertex v_j is displayed as j.



(b) The fluent dependency graph $G^{fl}_{\mathcal{E}_2}$ of $\mathcal{E}_2$. The contracted fluent dependency graph $G^{cdfl}_{\mathcal{E}_2}$ of $\mathcal{E}_2$ is the same as $G^{fl}_{\mathcal{E}_2}$.



(c) The fluent dependency graph $G^{fl}_{\mathcal{E}_3}$ of $\mathcal{E}_3$.



(d) The contracted fluent dependency graph $G^{cdfl}_{\mathcal{E}_3}$ of $\mathcal{E}_3$. We display a vertex v_{S_i} of a contracted fluent dependency graph, where S_i is a SCC of the corresponding fluent dependency graph, as the set of fluents whose vertices are in S_i.

Figure 1 Dependency graphs, fluent dependency graphs and contracted fluent dependency graphs. We use distinct shapes for the vertices of each type of graph to aid the presentation.

▶ **Definition 4** (Dependency Graph). *The dependency graph of an event description is a directed graph $G = (\mathcal{V}, \mathcal{E})$, where:*

1. *$\mathcal{V}$ contains one vertex $v_{F=V}$ for each FVP $F = V$.*
2. *$\mathcal{E}$ contains an edge $(v_{F_j = V_j}, v_{F_i = V_i})$ iff there is an* initiatedAt *or* terminatedAt *rule for $F_i = V_i$ having* holdsAt$(F_j = V_j, T)$ *as one of its conditions.*

The vertices and edges of Figure 1a that are drawn with continuous lines, e.g., comprise the dependency graph $G_{\mathcal{E}_1}$ of event description $\mathcal{E}_1$, which contains rules (2)–(8) of Example 3.

Based on the dependency graph of an event description, it is possible to define a function *level* that maps the FVPs of the event description to the positive integers. Towards defining an FVP level function, we define the *level of a vertex* in a directed acyclic graph as follows:

▶ **Definition 5** (Vertex Level). *Given a directed acyclic graph, the level of a vertex v is equal to:*

1. *1, if v has no incoming edges.*
2. *n, where $n > 1$, if v has at least one incoming edge from a vertex of level $n-1$, and zero or more incoming edges from vertices of levels lower than $n-1$.*

A dependency graph may or may not be acyclic. Given an acyclic dependency graph, the level of an FVP $F = V$ is defined as the level of vertex $v_{F=V}$ in the dependency graph. In the acyclic dependency graph of Figure 1a, e.g., $v_{interaction(P_1,P_2)=greeting}$ has level 2, and thus FVP $interaction(P_1, P_2) = greeting$ has level 2. In order to handle cyclic dependency graphs, we employ the *contracted dependency graph* of an event description, which is, by definition, acyclic. Then, we define the *level of an FVP* based on the level of the corresponding vertex in the contracted dependency graph.

A directed graph becomes acyclic by contracting its strongly connected components (SCC)s into single vertices.

▶ **Definition 6** (SCC Contracted Graph). *Given a directed graph $G=(\mathcal{V},\mathcal{E})$ and the SCCs $S_1, S_2, \ldots, S_n$ of G, the SCC contracted graph $G^{cd}=(\mathcal{V}^{cd},\mathcal{E}^{cd})$ of G is defined as follows:*

1. $\mathcal{V}^{cd}=\bigcup_{1\leq i\leq n}\{v_{S_i}\}$.
2. $(v_{S_i},\ v_{S_j})\in\mathcal{E}^{cd}$ *iff* $\exists v_i, v_j \in \mathcal{V}$, *such that* $v_i \in S_i$, $v_j \in S_j$, $S_i \neq S_j$ *and* $(v_i,\ v_j)\in\mathcal{E}$.

▶ **Definition 7** (Contracted Dependency Graph). *Consider an event description with dependency graph G. The contracted dependency graph of the event description is the SCC contracted graph of G.*

The dependency graph $G_{\mathcal{E}_1}$ in Figure 1a is acyclic, i.e., every SCCs of $G_{\mathcal{E}_1}$ contains one vertex. As a result, the contracted dependency graph $G^{cd}_{\mathcal{E}_1}$ of $G_{\mathcal{E}_1}$ is the same as $G_{\mathcal{E}_1}$.

▶ **Definition 8** (FVP Level in RTEC$_\circ$). *Consider an event description with dependency graph G and contracted dependency graph G^{cd}. The level of an FVP $F=V$, such that vertex $v_{F=V}$ is included in SCC S_i of G, is equal to the level of vertex v_{S_i} in G^{cd}.*

RTEC$_\circ$ supports event descriptions where FVPs with the same fluent have the same FVP level. For such an event description, a local stratification may be constructed as follows. The first stratum contains all groundings of happensAt. The remaining strata are formed by following, in a bottom-up fashion, the levels of FVPs. For each FVP level l without cyclic dependencies, we have one stratum containing the ground predicates for FVPs with level l. For each FVP level l with cyclic dependencies, the ground predicates for FVPs with level l have to be stratified further in terms of their time-stamp. We introduce an additional stratum for each time-point of the window, i.e., the finite portion of the stream currently being processing by RTEC$_\circ$.

▶ **Proposition 9** (Semantics of RTEC$_\circ$). *Consider an event description $\mathcal{E}$ where the FVPs with the same fluent have the same FVP level (see Definition 8). $\mathcal{E}$ is a locally stratified logic program [33].*

2.2 Reasoning & Complexity

The key reasoning task of RTEC$_\circ$ is the computation of holdsFor$(F=V, I)$, i.e., the list of maximal intervals I during which each FVP $F=V$ of the event description holds continuously. Recall that, in CER, FVPs express the composite activities that we are interested in detecting. RTEC$_\circ$ computes list I in holdsFor$(F=V, I)$ as follows. First, it computes the initiations of $F=V$ based on the rules of the event description with head initiatedAt$(F=V, T)$. Second, if there is at least one initiation of $F=V$, then RTEC$_\circ$ computes the terminations of $F=V$ based on the rules with head terminatedAt$(F=V, T)$, as well as the rules with head initiatedAt$(F=V', T)$, where $V'\neq V$. Third, RTEC$_\circ$ computes the maximal intervals of $F=V$ by matching each initiation T_s of $F=V$ with the first termination T_e of $F=V$ after T_s, ignoring every intermediate initiation between T_s and T_e. holdsAt$(F=V, T)$ may then be evaluated by checking whether T belongs to one of the maximal intervals of FVP $F=V$.

RTEC$_\circ$ processes FVPs in a bottom-up manner, computing and caching their intervals level-by-level. In order to derive the initiations and the terminations of an FVP $F=V$, we evaluate the initiatedAt and terminatedAt rules defining $F=V$. The body of such a rule may include a $\mathsf{holdsAt}(F' = V', T)$ condition (see rule schema (1)), leading to an edge $(v_{F'=V'}, v_{F=V})$ in the dependency graph (see Definition 4). We distinguish two cases for the evaluation of $\mathsf{holdsAt}(F' = V', T)$:

1. Vertices $v_{F'=V'}$ and $v_{F=V}$ are not part of a cycle in the dependency graph. In this case, $v_{F'=V'}$ and $v_{F=V}$ are in different SCCs of the dependency graph and, based on edge $(v_{F'=V'}, v_{F=V})$, $F' = V'$ has a lower level than $F{=}V$ (see Definition 8). Since RTEC$_\circ$ processes FVPs in ascending FVP level order, at the time of processing $F{=}V$, the intervals of $F' = V'$ that are required to compute $\mathsf{holdsAt}(F' = V', T)$ have been derived and cached at a previous step. As a result, $\mathsf{holdsAt}(F' = V', T)$ is resolved by fetching the intervals of $F' = V'$ from the cache and checking whether T belongs to one of those intervals, without the need for re-computation.
2. Vertices $v_{F'=V'}$ and $v_{F=V}$ are part of a cycle in the dependency graph. In this case, $v_{F'=V'}$ and $v_{F=V}$ are in the same SCC of the dependency graph, and thus $F' = V'$ and $F{=}V$ have the same level (see Definition 8). As a result, RTEC$_\circ$ may process $F{=}V$ before $F' = V'$, in which case the intervals of $F' = V'$ are not be present in the cache at the time of processing $F{=}V$. To address this issue, RTEC$_\circ$ computes $\mathsf{holdsAt}(F' = V', T)$ using the incremental caching techniques presented in [26].

3 Problem Statement

Towards a more accurate domain specification for human activity recognition, we may extend event description $\mathcal{E}_1$ of Example 3 with a definition for an FVP expressing that two people are talking.

▶ **Example 10** (Representing $\mathit{interaction}(P_1, P_2){=}\mathit{talking}$ (Example 3 cont'd))**.** After having approached one another, persons P_1 and P_2 may start talking, in which case the value of the $\mathit{interaction}(P_1, P_2)$ fluent should change from "*greeting*" to "*talking*". The specification of FVP $\mathit{interaction}(P_1, P_2){=}\mathit{talking}$ includes the following rules:

$$\begin{aligned}&\mathsf{initiatedAt}(\mathit{interaction}(P_1, P_2){=}\mathit{talking},\ T) \leftarrow\\&\quad \mathsf{happensAt}(\mathit{active}(P_1),\ T),\\&\quad \mathsf{holdsAt}(\mathit{distance}(P_1, P_2){=}\mathit{short},\ T),\ \mathsf{holdsAt}(\mathit{orientation}(P_1, P_2){=}\mathit{facing},\ T),\\&\quad \mathsf{not}\ \mathsf{holdsAt}(\mathit{movement}(P_1, P_2){=}\mathit{gathering},\ T).\end{aligned} \tag{9}$$

$$\begin{aligned}&\mathsf{initiatedAt}(\mathit{interaction}(P_1, P_2){=}\mathit{talking},\ T) \leftarrow\\&\quad \mathsf{happensAt}(\mathit{active}(P_2),\ T),\\&\quad \mathsf{holdsAt}(\mathit{distance}(P_1, P_2){=}\mathit{short},\ T),\ \mathsf{holdsAt}(\mathit{orientation}(P_1, P_2){=}\mathit{facing},\ T),\\&\quad \mathsf{not}\ \mathsf{holdsAt}(\mathit{movement}(P_1, P_2){=}\mathit{gathering},\ T).\end{aligned} \tag{10}$$

$$\begin{aligned}&\mathsf{terminatedAt}(\mathit{interaction}(P_1, P_2){=}\mathit{talking},\ T) \leftarrow\\&\quad \mathsf{happensAt}(\mathit{inactive}(P_1),\ T),\ \mathsf{happensAt}(\mathit{inactive}(P_2),\ T).\end{aligned} \tag{11}$$

According to rules (9)–(10), P_1 and P_2 start talking when one of them is being active, while their distance is about one meter, denoted by "short", they are facing one another and their relative movement is not "gathering", i.e., P_1 and P_2 are not moving towards one another. Rule (11) denotes that P_1 and P_2 stop talking when neither of them is being active.

A fluent cannot have more than one value at any time; an initiation of an FVP $F = V_1$ implies a termination of FVP $F = V_2$, where $V_1 \neq V_2$. As a result, there are implicit dependencies among FVPs with the same fluent. For instance, in the event description of Example 10, FVPs $\mathit{interaction}(P_1, P_2){=}\mathit{greeting}$ and $\mathit{interaction}(P_1, P_2){=}\mathit{talking}$ implicitly depend on each other.

The vertices and edges of Figure 1a that are drawn with continuous or dashed lines comprise the dependency graph $G_{\mathcal{E}_2}$ of event description $\mathcal{E}_2$, i.e., the extension of event description $\mathcal{E}_1$ with rules (9)–(11) of Example 10. FVPs $interaction(P_1, P_2) = greeting$ and $movement(P_1, P_2) = gathering$ have level 2, while FVP $interaction(P_1, P_2) = talking$ has level 3 (see Definition 8).

Event description $\mathcal{E}_2$ contains FVPs with the same fluent and different levels, which is common in CER specifications. In city transport management, e.g., fluent "$punctuality(Vh)$" may be used to monitor the punctuality level of a vehicle Vh over time [3]. $punctuality(Vh) = low$ may be initiated when Vh leaves a stop earlier than scheduled while $punctuality(Vh) = mid$ holds. As another example, in maritime activity monitoring, we may employ the fluent "$fishing\_trip(Vl)$" to survey a fishing trip of a vessel Vl [30]. FVP $fishing\_trip(Vl) = ended$ may depend on FVP $fishing\_trip(Vl) = returning$, which expresses the previous stage of the trip. In these cases, FVP $punctuality(Vh) = low$ has a higher level than FVP $punctuality = mid$, and FVP $fishing\_trip(Vl) = ended$ has a higher level than FVP $fishing\_trip(Vl) = returning$ (see Definition 8).

RTEC$_\circ$ does not support event descriptions, such as $\mathcal{E}_2$, where FVPs with the same fluent have different levels. Suppose that FVP $F = V_1$ has level n and FVP $F = V_2$ has level m, where $n < m$, and that RTEC$_\circ$ is currently processing the FVPs with level n. When processing $F = V_1$, RTEC$_\circ$ needs to evaluate the rules with head $\mathsf{initiatedAt}(F = V_2, T)$, as the initiation of $F = V_2$ constitute terminations of $F = V_1$. Such a rule may include a body condition referring to an FVP $F' = V'$ with level n', where $n \leq n' < m$. Since $F' = V'$ has a lower level than $F = V_2$, RTEC$_\circ$ attempts to evaluate $\mathsf{holdsAt}(F' = V', T)$ by retrieving the intervals of $F' = V'$ from the cache, in order to check whether T belongs to one of them. However, the cache of RTEC$_\circ$ may not contain the intervals of $F' = V'$ at this time, because $F' = V'$ has level n' and RTEC$_\circ$ is currently processing the FVPs with level n, where $n \leq n'$, compromising correctness.

In the case of event description $\mathcal{E}_2$, when processing $interaction(P_1, P_2) = greeting$, RTEC$_\circ$ evaluates the initiations of $interaction(P_1, P_2) = talking$, as they are terminations of $interaction(P_1, P_2) = greeting$. According to rules (9)–(10) of event description $\mathcal{E}_2$, the initiations of $interaction(P_1, P_2) = talking$ depend on FVP $movement(P_1, P_2) = gathering$, whose intervals may not present in the cache at the time of processing $interaction(P_1, P_2) = greeting$. For this reason, RTEC$_\circ$ does not support event description $\mathcal{E}_2$.

One way to address this issue is to assign to FVP $interaction(P_1, P_2) = greeting$ a higher level than the level of FVP $movement(P_1, P_2) = gathering$. According to dependency graph $G_{\mathcal{E}_2}$ (see Figure 1a), since there is no FVP that depends on FVP $interaction(P_1, P_2) = greeting$, we may increase the level of $interaction(P_1, P_2) = greeting$ to 3 without producing an FVP level assignment that compromises the correctness of the bottom-up processing of RTEC$_\circ$. In this way, $movement(P_1, P_2) = gathering$ is processed before $interaction(P_1, P_2) = greeting$, and thus, at the time of processing $interaction(P_1, P_2) = greeting$, the maximal intervals of $movement(P_1, P_2) = gathering$ are present in the cache of RTEC$_\circ$, avoiding the aforementioned error.

However, it is not always possible to circumvent the issues introduced by FVPs with the same fluent and different levels by increasing the level of an FVP. Consider the following example, where we extend event description $\mathcal{E}_2$ with a definition for an FVP expressing that two people are making abrupt movements while talking.

▶ **Example 11** (Representing $movement(P_1, P_2) = abrupt\_gestures$ (Example 10 cont'd)). While people P_1 and P_2 are talking, they may start moving their arms abruptly, possibly indicating that a fight between P_1 and P_2 is about to start. The specification of FVP $movement(P_1, P_2) = abrupt\_gestures$ includes the following rules:

$$\begin{aligned}&\mathsf{initiatedAt}(movement(P_1, P_2) = abrupt\_gestures,\ T) \leftarrow \\ &\quad \mathsf{happensAt}(abrupt(P_1),\ T), \\ &\quad \mathsf{holdsAt}(interaction(P_1, P_2) = talking,\ T).\end{aligned} \tag{12}$$

$$\begin{aligned}&\mathsf{initiatedAt}(movement(P_1, P_2) = abrupt\_gestures,\ T) \leftarrow \\ &\quad \mathsf{happensAt}(abrupt(P_2),\ T), \\ &\quad \mathsf{holdsAt}(interaction(P_1, P_2) = talking,\ T).\end{aligned} \tag{13}$$

$$\begin{aligned}&\mathsf{terminatedAt}(movement(P_1, P_2) = abrupt\_gestures,\ T) \leftarrow \\ &\quad \mathsf{happensAt}(active(P_1),\ T),\ \mathsf{not}\ \mathsf{happensAt}(abrupt(P_2),\ T).\end{aligned} \tag{14}$$

$$\begin{aligned}&\mathsf{terminatedAt}(movement(P_1, P_2) = abrupt\_gestures,\ T) \leftarrow \\ &\quad \mathsf{happensAt}(active(P_2),\ T),\ \mathsf{not}\ \mathsf{happensAt}(abrupt(P_1),\ T).\end{aligned} \tag{15}$$

Rules (12)–(13) denote that $movement(P_1, P_2) = abrupt\_gestures$ is initiated when one of the people P_1 and P_2 starts moving abruptly while the two of them are talking. Rules (14)–(15) express that we have a termination of $movement(P_1, P_2) = abrupt\_gestures$ when one of the two people starts being active while the other one is not moving abruptly.

All the vertices and edges in Figure 1a compose dependency graph $G_{\mathcal{E}_3}$ of event description $\mathcal{E}_3$, i.e., the extension of event description $\mathcal{E}_2$ with rules (12)–(15). According to dependency graph $G_{\mathcal{E}_3}$, FVP $movement(P_1, P_2) = abrupt\_gestures$ has level 4.

Event description $\mathcal{E}_3$ contains FVPs with the same fluent and different levels. The FVPs $interaction(P_1, P_2) = greeting$ and $interaction(P_1, P_2) = talking$ have level 2 and 3, respectively, while FVPs $movement(P_1, P_2) = gathering$ and $movement(P_1, P_2) = abrupt\_gestures$ have level 2 and 4. As a result, RTEC$_\circ$ does not support event description $\mathcal{E}_3$. When processing FVP $movement(P_1, P_2) = gathering$, RTEC$_\circ$ may need to evaluate its terminations, which include the initiations of FVP $movement(P_1, P_2) = abrupt\_gestures$. According to rules (12)–(13), the initiations of $movement(P_1, P_2) = abrupt\_gestures$ depend on $interaction(P_1, P_2) = talking$, whose intervals are not present in the cache at this time.

In this case, it is not possible to set the level of $movement(P_1, P_2) = gathering$ to 4, with the goal of processing $interaction(P_1, P_2) = talking$ before $movement(P_1, P_2) = gathering$, because there is an edge $(v_{movement(P_1,P_2)=gathering}, v_{interaction(P_1,P_2)=talking})$ in $G_{\mathcal{E}_3}$, implying that we cannot process $interaction(P_1, P_2) = talking$ before $movement(P_1, P_2) = gathering$. These FVPs should have the same level. Moreover, $interaction(P_1, P_2) = greeting$ depends on $interaction(P_1, P_2) = talking$, and vice versa, which means that these FVPs should also have the same level. Therefore, $movement(P_1, P_2) = gathering$, $interaction(P_1, P_2) = greeting$, $interaction(P_1, P_2) = talking$ and $movement(P_1, P_2) = abrupt\_gestures$ should have the same level, i.e., 2, implying that these FVPs must be processed with incremental caching (see the second case presented in Section 2.2).

4 Proposed Solution

We propose RTEC$_{fl}$, an extension of RTEC$_\circ$ that supports event descriptions where the vertices of FVPs with the same fluent may have different levels, such as event descriptions $\mathcal{E}_2$ and $\mathcal{E}_3$. To achieve this, RTEC$_{fl}$ incorporates a new definition for FVP level that takes into account the implicit dependencies between FVPs with the same fluent. We demonstrate that, based on the definition of FVP level in RTEC$_{fl}$, we may construct a local stratification for every possible event description. Afterwards, we propose a compiler for RTEC$_{fl}$, identifying the $\mathsf{holdsAt}(F = V, T)$ conditions that need to be resolved with the incremental caching technique proposed in [26], because the intervals of $F{=}V$ may not be present in the cache at the time of evaluating holdsAt($F = V$, T). We outline the cost of RTEC$_{fl}$, showing that it is the same as the cost of RTEC$_{\circ}$. Therefore, RTEC$_{fl}$ extends the range of temporal specifications supported by RTEC$_{\circ}$, while maintaining its high reasoning efficiency.

4.1 Syntax & Semantics

In RTEC$_{fl}$, all FVPs with the same fluent have the same level. This is achieved by determining FVP level based on the *fluent dependency graph* of the event description, which is defined as follows:

▶ **Definition 12** (Fluent Dependency Graph). *Consider an event description with dependency graph $G = (\mathcal{V}, \mathcal{E})$. The fluent dependency graph of the event description is a directed graph $G^{fl} = (\mathcal{V}^{fl}, \mathcal{E}^{fl})$, where:*

1. *$\mathcal{V}^{fl}$ contains one vertex v_F for each fluent F.*
2. *$\mathcal{E}^{fl}$ contains an edge (v_{F_1}, v_{F_2}), where $F_1 \neq F_2$, iff there is an edge $(v_{F_1 = V_1}, v_{F_2 = V_2})$ in $\mathcal{E}$, where V_1 and V_2 are values of fluents F_1 and F_2, respectively.*

Figure 1b, e.g., depicts the fluent dependency graph $G^{fl}_{\mathcal{E}_2}$ of event description $\mathcal{E}_2$ of Example 10. Vertex $v_{interaction(P_1,P_2)}$ of $G^{fl}_{\mathcal{E}_2}$ corresponds to vertices $v_{interaction(P_1,P_2) = greeting}$ and $v_{interaction(P_1,P_2) = talking}$ of $G_{\mathcal{E}_2}$, inheriting their incoming edges.

The fluent dependency graph $G^{fl}_{\mathcal{E}_2}$ is acyclic. Therefore, we may assign to each FVP $F = V$ of event description $\mathcal{E}_2$ the level of vertex v_F in the fluent dependency graph $G^{fl}_{\mathcal{E}_2}$, which is derived by following Definition 5. It could be the case, however, that the fluent dependency graph of an event description contains cycles. Figure 1c, e.g., depicts the fluent dependency graph $G^{fl}_{\mathcal{E}_3}$ of event description $\mathcal{E}_3$. $G^{fl}_{\mathcal{E}_3}$ includes a cycle, while, according to Definition 5, the level of a vertex is defined only on acyclic graphs. To address this issue, we contract the vertices of the fluent dependency graph that are in the same strongly connected component (SCC), leading to an acyclic graph. We define the *contracted fluent dependency graph* as follows:

▶ **Definition 13** (Contracted Fluent Dependency Graph). *Consider an event description with fluent dependency graph G^{fl}. The contracted fluent dependency graph G^{cdfl} of the event description is the SCC contracted graph of G^{fl}.*

Consider, e.g., the fluent dependency graph $G^{fl}_{\mathcal{E}_2}$ of Figure 1b. $G^{fl}_{\mathcal{E}_2}$ is acyclic, and thus every SCC of $G^{fl}_{\mathcal{E}_2}$ contains one vertex. As a result, the contracted fluent dependency graph $G^{cdfl}_{\mathcal{E}_2}$ of $G^{fl}_{\mathcal{E}_2}$ is the same as $G^{fl}_{\mathcal{E}_2}$. As another example, Figure 1d presents the contracted fluent dependency graph $G^{cdfl}_{\mathcal{E}_3}$ corresponding to the fluent dependency graph $G^{fl}_{\mathcal{E}_3}$ in Figure 1c, which is produced by contracting vertices $v_{movement(P_1,P_2)}$ and $v_{interaction(P_1,P_2)}$ of $G^{fl}_{\mathcal{E}_3}$, as these vertices are in the same SCC of $G^{fl}_{\mathcal{E}_3}$. Due to this contraction of vertices, $G^{cdfl}_{\mathcal{E}_3}$ is acyclic.

We may assign a level to each vertex in a contracted fluent dependency graph by following Definition 5. We define the *level of an FVP* in RTEC$_{fl}$ as follows:

▶ **Definition 14** (FVP Level in RTEC$_{fl}$). *Consider an event description with fluent dependency graph G^{fl} and contracted fluent dependency graph G^{cdfl}. The level of an FVP $F = V$, such that vertex v_F is included in SCC S_i of G^{fl}, is equal to the level of vertex v_{S_i} of G^{cdfl}.*

Based on Definition 14, FVPs with the same fluent have the same level. In the case of event description $\mathcal{E}_2$, e.g., where the contracted fluent dependency graph $G^{cdfl}_{\mathcal{E}_2}$ of $\mathcal{E}_2$ matches with the fluent dependency graph in Figure 1b, FVPs *interaction*$(P_1, P_2) =$ *greeting* and

Algorithm 1 $compile(\mathcal{E})$.

1: $G^{cdfl}_{\mathcal{E}} \leftarrow construct\_contracted\_fluent\_dependency\_graph(\mathcal{E})$
2: $level \leftarrow compute\_fvp\_level(G^{cdfl}_{\mathcal{E}})$
3: **for each** rule r **in** $\mathcal{E}$ **do**
4: $\quad F = V \leftarrow get\_fvp\_in\_head(r)$
5: $\quad$ **for each** condition "[not] holdsAt($F' = V'$, T)" **in** the body of r **do** ▷ not is optional.
6: $\quad\quad$ **if** $level[F' = V'] = level[F = V]$ **then**
7: $\quad\quad\quad$ **replace** "[not] holdsAt($F' = V'$, T)" **with** "[not] holdsAtCyclic($F' = V'$, T)" **in** r
8: **return** $\mathcal{E}$

$interaction(P_1, P_2) = talking$ have level *3* because the level of vertex $v_{interaction(P_1,P_2)}$ in $G^{cdfl}_{\mathcal{E}_2}$ is *3*. In the case of event description $\mathcal{E}_3$, the vertex of the contracted fluent dependency graph corresponding to fluents $movement(P_1, P_2)$ and $interaction(P_1, P_2)$ has level *2* (see Figure 1d). Thus, FVPs $interaction(P_1, P_2) = greeting$, $movement(P_1, P_2) = gathering$, $interaction(P_1, P_2) = talking$ and $movement(P_1, P_2) = abrupt\_gestures$ have level *2*.

We can devise a local stratification of an event description by following bottom-up the levels of FVPs, as specified in Definition 14. For each level with cyclic dependencies, we introduce an additional stratum per time-point, following an ascending temporal order.

▶ **Proposition 15** (Semantics of RTEC$_{fl}$)**.** *An event description is a locally stratified logic program.*

According to Proposition 15, RTEC$_{fl}$ supports every event description $\mathcal{E}$ that follows Definition 1. If the dependency graph of $\mathcal{E}$ contains FVPs with the same fluent whose vertices are in different levels of the graph, then these FVPs are assigned the same level, following the definition of FVP level in RTEC$_{fl}$ (see Definition 14), avoiding the issues described in Section 3.

4.2 Compiler

We developed a compiler that assigns a level to each FVP of an input event description $\mathcal{E}$ and marks the holdsAt body conditions of the rules in $\mathcal{E}$ that must be evaluated with incremental caching, in order to guarantee correct reasoning. The compilation is performed before the commencement of run-time reasoning, in a process transparent to the event description developer. Algorithm 1 outlines the compilation steps. First, we derive the levels of FVPs by following Definitions 13 and 14. We construct the contracted fluent dependency graph $G^{cdfl}_{\mathcal{E}}$ of $\mathcal{E}$ (line 1 of Algorithm 1). Then, we assign a level to each FVP in $\mathcal{E}$ based on the level of the corresponding vertex of $G^{cdfl}_{\mathcal{E}}$ (line 2). In order to identify the holdsAt conditions that need to be evaluated with incremental caching, the compiler works as follows. For each holdsAt($F' = V'$, T) or "not holdsAt($F' = V'$, T)" condition in the body of a rule in $\mathcal{E}$, the compiler checks whether the level of FVP $F' = V'$ is equal to the level of the FVP in the head of the rule (lines 3–6). If this is the case, then we translate condition holdsAt($F' = V'$, T) (resp. "not holdsAt($F' = V'$, T)") into holdsAtCyclic($F' = V'$, T) (resp. "not holdsAtCyclic($F' = V'$, T)") (line 7). At run-time, RTEC$_{fl}$ evaluates the conditions with holdsAtCyclic using incremental caching (recall the second case presented in Section 2.2) and the conditions with holdsAt using the interval retrieval operation (see the first case of Section 2.2). A further discussion on run-time reasoning is presented in the section that follows.

We tested the compiler of RTEC$_{\mathit{fl}}$ on event descriptions from various CER applications, including human activity recognition [2], city transport management [3] and maritime situational awareness [29, 30]. Moreover, we have used our compiler in applications that involve the monitoring of the normative positions of agents in multi-agent systems, such as e-commerce [35] and voting protocols [31]. In all cases, the compilation time amounted to a few milliseconds, and thus we do not show these times here. The compiler is available with the code of RTEC$_{\mathit{fl}}$[1].

4.3 Reasoning & Complexity

RTEC$_{\mathit{fl}}$ follows RTEC$_\circ$ and processes FVPs in ascending FVP level order. When processing a rule that includes a holdsAtCyclic$(F' = V', T)$ condition, RTEC$_{\mathit{fl}}$ computes the changes in the value of F' between T_{leq} and T, where T_{leq} is the last time-point before T where the truth value of holdsAt$(F' = V', T_{leq})$ has been evaluated and cached. In the worst-case, the cost of this process is $\mathcal{O}(\omega k)$, where ω is the size of the window and k is the cost of computing whether an FVP is initiated or terminated at a given time-point (see [3] for an estimation of k). This is the same incremental caching technique as the one used in RTEC$_\circ$, thus yielding the same cost [26]. In the case of a holdsAt$(F' = V', T)$ condition, RTEC$_{\mathit{fl}}$ retrieves the maximal intervals of $F' = V'$ from its cache and checks whether T belongs to one of the retrieved intervals. Since the cached intervals are temporally sorted, this is achieved with a binary search, while the number of cached intervals of $F' = V'$ is bounded by ω. Therefore, the cost of an interval retrieval operation in RTEC$_{\mathit{fl}}$ is $\mathcal{O}(log(\omega))$, which is the same as the cost of this operation in RTEC$_\circ$. As a result, RTEC$_{\mathit{fl}}$ yields the same worst-case time complexity as RTEC$_\circ$, while supporting a wider range of temporal specifications. By following Definition 14 for FVP level, RTEC$_\circ$ reasons with incremental caching only when it is necessary, i.e., only when the required intervals may not be present in the cache.

5 Summary and Future Work

We proposed RTEC$_{\mathit{fl}}$, an extension of RTEC$_\circ$, which detects composite activities based on their Event Calculus definitions, in order to support every possible set of such definitions. We described the syntax and semantics of RTEC$_{\mathit{fl}}$, demonstrating that activity specifications in RTEC$_{\mathit{fl}}$ are locally stratified logic programs. Afterwards, we proposed a compiler for RTEC$_{\mathit{fl}}$, identifying the conditions of activity definitions that may be evaluated with an efficient cache operation, without sacrificing correctness, with the goal of improving reasoning efficiency at run-time. We outlined the worst-case time complexity of RTEC$_{\mathit{fl}}$, showing that it yields the same cost as RTEC$_\circ$. As a result, RTEC$_{\mathit{fl}}$ supports a wider range of temporal specifications than RTEC$_\circ$, while maintaining its high reasoning efficiency. The code of RTEC$_{\mathit{fl}}$ is publicly available[1].

In the future, we aim to compare RTEC$_{\mathit{fl}}$ with automata-based activity recognition frameworks, such as [10, 40].

References

1 Elias Alevizos, Anastasios Skarlatidis, Alexander Artikis, and Georgios Paliouras. Probabilistic complex event recognition: A survey. *Commun. ACM*, 50(5):71:1–71:31, 2017. doi:10.1145/3117809.

2 Alexander Artikis, Marek J. Sergot, and Georgios Paliouras. A logic programming approach to activity recognition. In *EIMM Workshop in MM*, pages 3–8, 2010. doi:10.1145/1877937.1877941.

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. doi:10.1109/TKDE.2014.2356476.

4 Hamid R. Bazoobandi, Harald Beck, and Jacopo Urbani. Expressive stream reasoning with laser. In *ISWC*, volume 10587, pages 87–103, 2017. doi:10.1007/978-3-319-68288-4_6.

5 Harald Beck, Minh Dao-Tran, and Thomas Eiter. LARS: A logic-based framework for analytic reasoning over streams. *Artif. Intell.*, 261:16–70, 2018. doi:10.1016/J.ARTINT.2018.04.003.

6 Harald Beck, Thomas Eiter, and Christian Folie. Ticker: A system for incremental asp-based stream reasoning. *Theory Pract. Log. Program.*, 17(5-6):744–763, 2017. doi:10.1017/S1471068417000370.

7 Stefano Bragaglia, Federico Chesani, Paola Mello, Marco Montali, and Paolo Torroni. Reactive event calculus for monitoring global computing applications. In *Logic Programs, Norms and Action - Essays in Honor of Marek J. Sergot on the Occasion of His 60th Birthday*, volume 7360, pages 123–146, 2012. doi:10.1007/978-3-642-29414-3_8.

8 Sebastian Brandt, Elem Güzel Kalayci, Vladislav Ryzhikov, Guohui Xiao, and Michael Zakharyaschev. Querying log data with metric temporal logic. *J. Artif. Intell. Res.*, 62:829–877, 2018. doi:10.1613/JAIR.1.11229.

9 Stefano Bromuri, Visara Urovi, and Kostas Stathis. icampus: A connected campus in the ambient event calculus. *Int. J. Ambient Comput. Intell.*, 2(1):59–65, 2010. doi:10.4018/JACI.2010010105.

10 Marco Bucchi, Alejandro Grez, Andrés Quintana, Cristian Riveros, and Stijn Vansummeren. CORE: a complex event recognition engine. *Proc. VLDB Endow.*, 15(9):1951–1964, 2022. doi:10.14778/3538598.3538615.

11 Francesco Calimeri, Marco Manna, Elena Mastria, Maria Concetta Morelli, Simona Perri, and Jessica Zangari. I-dlv-sr: A stream reasoning system based on I-DLV. *Theory Pract. Log. Program.*, 21(5):610–628, 2021. doi:10.1017/S147106842100034X.

12 Iliano Cervesato and Angelo Montanari. A calculus of macro-events: Progress report. In *TIME*, pages 47–58, 2000. doi:10.1109/TIME.2000.856584.

13 Hervé Chaudet. Extending the event calculus for tracking epidemic spread. *Artif. Intell. Medicine*, 38(2):137–156, 2006. doi:10.1016/J.ARTMED.2005.06.001.

14 L. Chittaro and A. Montanari. Efficient temporal reasoning in the cached event calculus. *Comput. Intell.*, 12(3):359–382, 1996. doi:10.1111/J.1467-8640.1996.TB00267.X.

15 Keith L. Clark. Negation as failure. In *Logic and Data Bases*, pages 293–322. Plemum Press, 1977. doi:10.1007/978-1-4684-3384-5_11.

16 Gianpaolo Cugola and Alessandro Margara. Processing flows of information: From data stream to complex event processing. *ACM Comput. Surv.*, 44(3), 2012. doi:10.1145/2187671.2187677.

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

18 Thomas Eiter, Paul Ogris, and Konstantin Schekotihin. A distributed approach to LARS stream reasoning (system paper). *Theory Pract. Log. Program.*, 19(5-6):974–989, 2019. doi:10.1017/S1471068419000309.

19 Nicola Falcionelli, Paolo Sernani, Albert Brugués de la Torre, Dagmawi Neway Mekuria, Davide Calvaresi, Michael Schumacher, Aldo Franco Dragoni, and Stefano Bromuri. Indexing the event calculus: Towards practical human-readable personal health systems. *Artif. Intell. Medicine*, 96:154–166, 2019. doi:10.1016/J.ARTMED.2018.10.003.

20 Nikos Giatrakos, Elias Alevizos, Alexander Artikis, Antonios Deligiannakis, and Minos N. Garofalakis. Complex event recognition in the big data era: a survey. *VLDB J.*, 29(1):313–352, 2020. doi:10.1007/S00778-019-00557-W.

21 Alejandro Grez, Cristian Riveros, Martín Ugarte, and Stijn Vansummeren. A formal framework for complex event recognition. *ACM Trans. Database Syst.*, 46(4):16:1–16:49, 2021. doi:10.1145/3485463.

22 Özgür Kafali, Alfonso E. Romero, and Kostas Stathis. Agent-oriented activity recognition in the event calculus: An application for diabetic patients. *Comput. Intell.*, 33(4):899–925, 2017. doi:10.1111/COIN.12121.
23 Nikos Katzouris, Georgios Paliouras, and Alexander Artikis. Online learning probabilistic event calculus theories in answer set programming. *Theory Pract. Log. Program.*, 23(2):362–386, 2023. doi:10.1017/S1471068421000107.
24 Robert A. Kowalski and Marek J. Sergot. A logic-based calculus of events. *New Gener. Comput.*, 4(1):67–95, 1986. doi:10.1007/BF03037383.
25 Periklis Mantenoglou, Dimitrios Kelesis, and Alexander Artikis. Complex event recognition with allen relations. In *KR*, pages 502–511, 2023. doi:10.24963/KR.2023/49.
26 Periklis Mantenoglou, Manolis Pitsikalis, and Alexander Artikis. Stream reasoning with cycles. In *KR*, pages 544–553, 2022.
27 Adrian Paschke. Eca-ruleml: An approach combining ECA rules with temporal interval-based KR event/action logics and transactional update logics. *CoRR*, abs/cs/0610167, 2006.
28 Adrian Paschke and Martin Bichler. Knowledge representation concepts for automated SLA management. *Decis. Support Syst.*, 46(1):187–205, 2008. doi:10.1016/J.DSS.2008.06.008.
29 Kostas Patroumpas, Alexander Artikis, Nikos Katzouris, Marios Vodas, Yannis Theodoridis, and Nikos Pelekis. Event recognition for maritime surveillance. In *EDBT*, pages 629–640, 2015. doi:10.5441/002/EDBT.2015.63.
30 Manolis Pitsikalis, Alexander Artikis, Richard Dreo, Cyril Ray, Elena Camossi, and Anne-Laure Jousselme. Composite event recognition for maritime monitoring. In *DEBS*, pages 163–174, 2019. doi:10.1145/3328905.3329762.
31 J. Pitt, L. Kamara, M. Sergot, and A. Artikis. Voting in multi-agent systems. *Comput. J.*, 49(2):156–170, 2006. doi:10.1093/COMJNL/BXH164.
32 Olga Poppe, Chuan Lei, Elke A. Rundensteiner, and David Maier. Event trend aggregation under rich event matching semantics. In *SIGMOD*, pages 555–572, 2019. doi:10.1145/3299869.3319862.
33 Teodor C. Przymusinski. On the declarative semantics of deductive databases and logic programs. In *Foundations of Deductive Databases and Logic Programming*, pages 193–216. Morgan Kaufmann, 1988. doi:10.1016/B978-0-934613-40-8.50009-9.
34 Nausheen Saba Shahid, Dan O'Keeffe, and Kostas Stathis. A knowledge representation framework for evolutionary simulations with cognitive agents. In *ICTAI*, pages 361–368. IEEE, 2023. doi:10.1109/ICTAI59109.2023.00059.
35 M. Sirbu. Credits and debits on the Internet. *IEEE Spectrum*, 34(2):23–29, 1997.
36 Efthimis Tsilionis, Alexander Artikis, and Georgios Paliouras. Incremental event calculus for run-time reasoning. *J. Artif. Intell. Res.*, 73:967–1023, 2022. doi:10.1613/JAIR.1.12695.
37 Przemyslaw Andrzej Walega, Mark Kaminski, and Bernardo Cuenca Grau. Reasoning over streaming data in metric temporal datalog. In *AAAI*, pages 3092–3099, 2019. doi:10.1609/AAAI.V33I01.33013092.
38 Przemyslaw Andrzej Walega, Mark Kaminski, Dingmin Wang, and Bernardo Cuenca Grau. Stream reasoning with datalogmtl. *J. Web Semant.*, 76:100776, 2023. doi:10.1016/J.WEBSEM.2023.100776.
39 Bo Zhao, Han van der Aa, Thanh Tam Nguyen, Quoc Viet Hung Nguyen, and Matthias Weidlich. EIRES: efficient integration of remote data in event stream processing. In *SIGMOD*, pages 2128–2141, 2021. doi:10.1145/3448016.3457304.
40 Bartosz Zielinski. Explanatory denotational semantics for complex event patterns. *Formal Aspects Comput.*, 35(4):23:1–23:37, 2023. doi:10.1145/3608486.