Reasoning over Streams of Events with Delayed Effects

Periklis Mantenoglou[1] , **Manolis Pitsikalis**[3] , **Alexander Artikis**[2,3]
[1]Örebro University, Sweden
[2]University of Piraeus, Greece
[3]NCSR "Demokritos", Greece
periklis.mantenoglou@oru.se, manospits@iit.demokritos.gr, a.artikis@unipi.gr

1 Motivation & Contributions

Many modern applications require reasoning over continuous event streams to detect situations of interest with minimal latency. Examples include maritime situational awareness, monitoring of norm conformity in multi-agent protocols, and simulation tracking for biological processes.

A central challenge in stream reasoning arises when events produce *delayed effects*–consequences that occur after some time delay. For example, in maritime monitoring, a vessel manoeuvre may indicate the termination of a fishing activity several minutes later unless other manoeuvres occur in the meantime. Similarly, in biological systems the activation of a gene may influence protein concentrations only after a delay. Existing formalisms expressing such delayed effects lack scalable implementations for high-volume streams, while modern stream reasoning engines achieve high throughput but lack the expressive power and formal guarantees to model delayed effects in a principled way.

Our recent paper addresses this gap by extending the Run-Time Event Calculus (RTEC) (Artikis et al. 2015), a state-of-the-art logic programming stream reasoning engine, with the syntactic constructs and reasoning algorithms needed to handle events with delayed effects efficiently and with formal guarantees. The contributions are:

1. The development of RTEC$^{\rightarrow}$, i.e., an open-source[1] formal computational framework for reasoning over streams of events with delayed effects.
2. A suite of novel reasoning algorithms for handling delayed effects in streaming environments. These algorithms determine an efficient processing order for event patterns, identify the minimal information that must be transferred between execution windows, and employ incremental reasoning and caching techniques. These features led to significant efficiency improvements in reasoning over events with delayed effects.
3. A formal analysis of RTEC$^{\rightarrow}$, including proofs of correctness of its reasoning procedures and analyses of their worst-case time complexity.
4. A comprehensive and *reproducible* empirical evaluation using large synthetic and real-world event streams and complex temporal specifications. The evaluation demonstrated that RTEC$^{\rightarrow}$ is scalable to large-scale event streams and outperforms state-of-the-art approaches by orders of magnitude.

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

2 Summary of the Paper

We provide a brief overview of the paper. We outline some background on RTEC, then proceed with the novel features of RTEC$^{\rightarrow}$ with respect to knowledge representation, semantics and reasoning, and conclude with a discussion on our experimental evaluation and results.

Background. RTEC is a logic programming framework for recognising situations of interest by reasoning over streams of time-stamped input events. The language of RTEC includes time-points, events and "fluents", i.e., properties that may have different values at different points in time. Situations of interest are expressed by means of fluent-value pairs (FVPs). As an example from a multi-agent e-commerce protocol, the $quote(M, C, G)$ fluent may be used to express the status of an offer from a merchant M to a consumer C about goods G. Then, the FVPs $quote(M, C, G) = \mathsf{true}$ and $quote(M, C, G) = \mathsf{false}$ can be used to express whether the offer currently stands or not. RTEC assumes that fluents follow the law of inertia, i.e., their values do not change unless an explicit FVP initiation or termination takes place. Such initiations and terminations are defined via domain specific rules. For example, the following rules may be used to define the values of the $quote(M, C, G)$ fluent.

$$\mathsf{initiatedAt}(quote(M, C, G) = \mathsf{true}, T) \leftarrow \mathsf{happensAt}(present\_quote(M, C, G, P), T). \quad (1)$$

$$\mathsf{initiatedAt}(quote(M, C, G) = \mathsf{false}, T) \leftarrow \mathsf{happensAt}(accept\_quote(C, M, G), T). \quad (2)$$

According to rule (1), $quote(M, C, G) = \mathsf{true}$ is initiated at time-point T (i.e., $\mathsf{initiatedAt}(quote(M, C, G) = \mathsf{true}, T)$ holds), if merchant M presents a price P for goods G to consumer C (i.e., $\mathsf{happensAt}(present\_quote(M, C, G, P), T)$ holds). Then, $quote(M, C, G) = \mathsf{false}$ is initiated—implicitly terminating $quote(M, C, G) = \mathsf{true}$—when the offer is accepted by the consumer (see rule (2)).

Rules (1)–(2) capture immediate effects of events on the $quote(M, C, G)$ fluent, as the same time-point T is used in both the happensAt predicate, expressing the input events that affect the world, and the initiatedAt predicate, expressing the fluent changes based on those events. RTEC only supports events with immediate effects. In our publication, we

proposed RTEC$^{\rightarrow}$, extending RTEC with support for events that may change the values of fluents after a time delay.

Knowledge Representation. RTEC$^{\rightarrow}$ extends the language of RTEC with $\mathsf{fi}(F=V, F=V', R)$ facts, expressing that the initiation of FVP $F=V$ at a time-point T leads to the *future initiation* (fi) of FVP $F=V'$ at time-point $T+R$, provided that $F=V$ is not terminated between T and $T+R$. Additionally, RTEC$^{\rightarrow}$ uses $\mathsf{p}(F=V)$ facts to designate that the future initiation specified by $\mathsf{fi}(F=V, F=V', R)$ may be *postponed* in case another initiation of $F=V$ occurs between T and $T+R$. This additional expressivity allows us, e.g., to extend the scope of the definition of the $quote(M, C, G)$ fluent, by including facts $\mathsf{fi}(quote(M, C, G)=\mathsf{true}, quote(M, C, G)=exp, r_e)$ and $\mathsf{fi}(quote(M, C, G)=exp, quote(M, C, G)=\mathsf{false}, r_f)$, expressing that an offer is marked as being close to expiring r_e time-points after it is proposed, and that it expires r_f time-points thereafter.

Semantics. Events with delayed effects introduce additional dependencies among FVPs, and may therefore affect the semantics of the framework. In the case of RTEC, semantics are defined via a graph structure that encapsulates all dependencies among FVPs that are imposed by the initiation and termination rules in the logic program specifying the domain. Then, a model for the program is constructed by computing the time-points where FVPs hold, following a bottom-up order on this FVP dependency graph. To cater for the FVP dependencies stemming from fi facts in an RTEC$^{\rightarrow}$ program, we revised the definition of this graph by introducing a special type of edge, encapsulating these new dependencies. We proved that constructing an interpretation based on the updated graph yields a unique model for the program, thus allowing us to compute all FVP occurrences without compromising the semantics of the original system.

Reasoning Algorithms. We introduced a suite of reasoning algorithms for evaluating the FVPs in an RTEC$^{\rightarrow}$ dependency graph efficiently. This suite includes a compile-time algorithm, identifying the optimal order for processing a set of FVPs that depend on one another via at least one fi fact, and a run-time algorithm that follows this processing order incrementally, while caching appropriately intermediate results, thus minimising redundant computations. In addition, we developed an algorithm that identifies the minimal information that must be transferred between consecutive execution windows, allowing future effects to be derived correctly even when the triggering events occurred in earlier windows. These techniques have established correctness guarantees and low computational complexity, enabling stream reasoning over events with delayed effects.

Experimental Evaluation. We conducted an extensive empirical evaluation of RTEC$^{\rightarrow}$ in streaming applications that include events with delayed effects, such as multi-agent e-commerce and voting protocols, composite event recognition for maritime situational awareness, and simulation monitoring for biological feedback loops. The evaluation used both synthetic and real-world event streams, such as maritime datasets containing millions of vessel position signals, and an extensive set of baselines, including leading Event Calculus implementations and stream reasoners. The

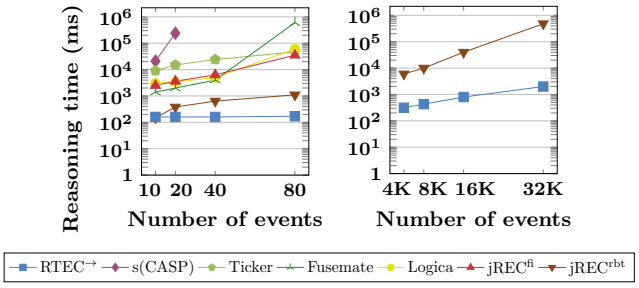

Figure 1: Stream reasoning for computing $quote$.

results demonstrated that RTEC$^{\rightarrow}$ scales efficiently to large streams and consistently outperforms competing approaches. In several cases, competing systems either failed to scale or required orders of magnitude more time to compute the same results (see, e.g., the results in Figure 1).

3 Significance & Relevance to KR

The significance of RTEC$^{\rightarrow}$ lies in both its theoretical and practical contributions. The formal treatment of events with delayed effects in a stream reasoning framework is novel, and also beneficial for further studies on stream reasoning expressivity. The reasoning algorithms of RTEC$^{\rightarrow}$ have performance and optimality guarantees, paving the way for robust and low-latency stream reasoning. At the same time, RTEC$^{\rightarrow}$ proved scalable in real-world domains, processing streams with millions of events and recognising situations of interest from a broad scope of domains with minimal latency, while outperforming the state of the art by orders of magnitude.

This work is relevant to the KR research agenda. Particularly, it is pertinent to the "logic programming", "temporal reasoning" and "reasoning about action and change" topics of KR. Moreover, work on RTEC has been accepted in previous iterations of KR. For example, an extension of RTEC with efficient processing of cyclic FVP definitions was presented in KR 2022 (Mantenoglou et al. 2022), while there was a work on incorporating Allen's interval relations in RTEC patterns in KR 2023 (Mantenoglou et al. 2023).

4 Limitations & Future Work

The future initiation mechanism of RTEC$^{\rightarrow}$ is restricted to FVPs with the same fluent, limiting its expressivity. Moreover, the novel reasoning algorithms of RTEC$^{\rightarrow}$are not optimised for out-of-order streams. Lifting both restrictions constitutes promising future work.

References

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

Mantenoglou, P.; Pitsikalis, M.; and Artikis, A. 2022. Stream reasoning with cycles. In *KR*.

Mantenoglou, P.; Kelesis, D.; and Artikis, A. 2023. Complex event recognition with allen relations. In *KR*, 502–511.