

Towards Net-based Formal Methods for Complex Event Processing

Michael Offel, Han van der Aa, and Matthias Weidlich

Department of Computer Science, Humboldt-Universität zu Berlin
{michael.offel,han.van.der.aa,matthias.weidlich}@hu-berlin.de

Abstract. Complex event processing (CEP) systems evaluate continuous queries over event streams in order to detect event patterns that characterise a situation of interest. Over the last decade, models and systems for CEP have been an active area of research. A plethora of languages for the definition of queries as well as algorithms and architectures for their efficient and robust evaluation have been proposed. However, we argue that much of this work adopted a pragmatic view, striving for technical solutions of problems that are rooted directly in specific applications. As such, formal methods that help in the design and implementation of CEP systems are underdeveloped. In this paper, we put forward the idea of exploiting well-established formalisms for concurrent systems in the analysis of CEP systems. We discuss query verification, query parallelisation, and out-of-order event handling as three exemplary use cases for formal methods for CEP. Moreover, we outline our initial results on addressing these use cases through a Petri-net-based formalisation of event queries, event streams, and evaluation architectures.

1 Motivation

In complex event processing (CEP), predefined queries are continuously evaluated over streams of events in order to detect situations of interest [1]. As such, CEP serves as the foundation of applications related to monitoring, detection, and online response in domains such as healthcare and urban transportation.

CEP Query Languages. Various languages for the definition of CEP queries have been proposed in the literature, differing in syntax and semantics. Yet, many languages adopt point-based event semantics (an event occurs at a particular point in time), an attribute-based data model (the payload of an event is structured by its type, given as a relational schema), and feature a set of common query operators [8]. The latter includes *sequencing*, the definition of a list of event types that shall occur in the respective temporal order; *negation*, the check for the absence of an event of a particular type; *value predicates*, the definition of attribute-based match conditions for events; and *windowing*, the specification of a time interval for the occurrence of events.

Consider, for instance, monitoring of clinical pathways in a hospital. Events may indicate that a patient received an infusion or was examined. Evaluating the query defined in Figure 1 over such events would then detect a situation where a patient gets two infusions without being examined in between.

```

Pattern SEQ(infusion i1, ! examination e, infusion i2)
Where i1.patient = e.patient And i1.patient = i2.patient
And e.scope = 'full vitals'
Within 2h

```

Fig. 1: Example query to detect a sequence of two infusions without examination.

CEP Systems. Aiming at low-latency evaluation of CEP queries over high-velocity event streams, various algorithms and architectures for efficient query processing have been proposed. Here, angles for optimisation include techniques for parallelisation and distribution of query evaluation, semantic rewriting of queries [6], or sub-pattern sharing among multiple queries [4].

Since event streams often originate in distributed sources, techniques to make query evaluation robust in a distributed setting have been another focal point of research. To cope with out-of-order arrival of events at a CEP system (i.e., their arrival order contradicts the timestamps assigned by event sources), techniques for event buffering and speculative query evaluation have been developed [2].

Use Cases for Formal Methods in CEP. Reflecting on CEP query languages and systems, we argue that much of the aforementioned work adopted a pragmatic view and was primarily driven by specific applications. The focus has been on the definition of query languages that are appropriate for a particular context. Yet, despite some notable attempts (e.g., based on data-enriched automata or logic-based rules), there is no commonly accepted, comprehensive model that could serve as a formal foundation for the semantics of CEP queries. In the same vein, techniques for optimised and robust query evaluation are mostly built on empirical arguments (e.g., by tuning cost models for query evaluation). Formal foundations of CEP architectures (e.g., parallelisation schemes or buffering methods) have been largely neglected in the literature.

These observations are despite the potential value of formal methods for various questions related to the design and implementation of CEP systems. Examples of such questions are:

- *Query verification:* Can a query deployed in a CEP system match at all, if event generation follows a set of domain-specific constraints?
- *Sound workload parallelisation:* If the queries of a workload are evaluated concurrently (e.g., through data- or task-parallelism) [3], will the result become non-deterministic due to interactions between different queries?
- *Robustness guarantees:* If events arrive out-of-order at a CEP system [2], will the result of a query be affected and, if so, what kind of guarantees can still be given (e.g., no false positives or no false negatives)?

To enable reasoning on the above questions, we advocate the use of well-established formalisms to model complex event processing. However, we note that this requires a comprehensive formalisation that integrates the semantics of event streams, event queries, and evaluation architectures. Below, we outline our initial steps towards such a unified formal foundation of CEP.

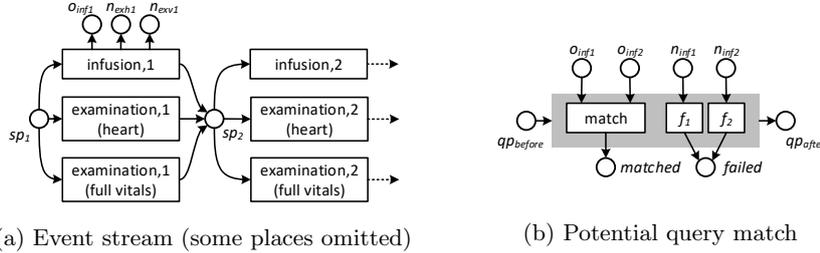


Fig. 2: Excerpts of Petri-nets modelling CEP concepts.

2 Formal Analysis of CEP Systems

To analyse CEP systems, we developed a formal model of event streams, event queries, evaluation architectures based on Petri-nets [5]. The choice of Petri-nets as a formalism is motivated by their truly *concurrent* (instead of interleaving) and *local* (instead of global) semantics, which naturally captures the characteristics of distributed event sources and concurrent query evaluation. Also, a large body of verification techniques (including tool support) are available for Petri-nets.

Below, we outline the general idea of our formalisation by means of an example. We further sketch how the above questions on the design and implementation of a CEP system can be approached based on reachability analysis.

Formalising Streams, Queries, and Evaluation Architectures. A Petri-net is a directed graph of places (denoted by circles) and transitions (denoted by rectangles). Its state is modelled by a marking, a distribution of tokens over places. Transitions model state changes: If all places in the pre-set of a transition carry a token, the transition may fire and, by doing so, consume one token from each place in its pre-set and produce one token in each place in its post-set.

We capture an event stream as shown in Figure 2a. If place sp_1 is marked, which indicates the start of the stream, at time unit 1, three transitions can be fired. They represent the generation of an event, at time unit 1, of type *infusion* or of type *examination*, the latter having attribute *scope* set to *heart* or *full vitals*. Upon firing, the transitions mark places that capture the occurrence (e.g., o_{inf1} for the infusion event) or non-occurrence (e.g., n_{exh1} for the heart examination event) of an event. By extending the net at place sp_2 , a complete event stream is modelled. This way, domain constraints may be represented, e.g., that *examination* events of different *scope* cannot follow upon each other directly.

For event queries, the model captures potential query matches, i.e., particular combinations of events that represent a pattern. For the example query, a potential match would be an occurrence of two *infusion* events following each other directly. The respective net, exemplified in Figure 2b, features one transition (*match*) for pattern occurrence, which may only fire if the places in its pre-set (potentially marked by the transitions representing the stream) indeed indicate the occurrence of the respective events. In addition, it also features two transitions (f_1 , f_2) to represent that a pattern did not occur. This particular potential match cannot occur, if either the first or the second stream event is not of type *infusion*.

The evaluation architecture is represented by how the nets of potential matches are glued together. For instance, to capture centralised, single-threaded query evaluation, the nets of potential matches are composed sequentially, connecting qp_{after} of one potential match with qp_{before} of another one. Then, the order of composition may not only capture the temporal order of potential matches, but can also represent evaluation priorities among queries in a workload. Parallel composition of nets, in turn, models data- or task-parallelism in query processing.

Reasoning on CEP Systems. A formalisation as sketched above enables reasoning on CEP systems based on reachability analysis, i.e., by exploring whether certain states in the Petri-net can be reached from an initial marking. Specifically, for query verification, the reachability of markings that cover places *matched* in all nets of potential query matches determines whether domain constraints on the event stream preclude the detection of a pattern through a specific query. Soundness of a workload parallelisation scheme is established by verifying whether the reachability of markings covering places *matched* changes, when varying the composition of the nets that model potential query matches. Finally, out-of-order arrival of events may directly be encoded when glueing together the nets of a stream and potential query matches. Again, the reachability of markings covering places *matched* and *failed* after incorporating these changes enables conclusions on the possibility of generating false positives and false negatives.

Note that the above formalisation will create Petri-nets of significant complexity for realistic scenarios. However, state-of-the-art model checkers for Petri-nets are able to handle dozens of millions of states [7], rendering such analysis feasible.

3 Conclusion

We put forward the idea to develop formal methods for CEP, based on well-established formalisms for concurrent systems. Specifically, we presented initial results on how to approach query verification, query parallelisation, and out-of-order event handling through a Petri-net-based formalisation of event queries, event streams, and evaluation architectures. Challenges are still imposed by the expressiveness of CEP languages, featuring concepts such as Kleene closure operators and advanced event consumption policies. Also, we are in the process of developing tool support for the envisioned verification of CEP systems.

References

1. Cugola, G., Margara, A.: Processing flows of information: From data stream to complex event processing. *ACM Comput. Surv.* 44(3), 15:1–15:62 (2012)
2. Liu, M., et al.: Sequence pattern query processing over out-of-order event streams. In: *ICDE*. IEEE CS (2009)
3. Mayer, R., et al.: SPECTRE: Supporting consumption policies in window-based parallel complex event processing. In: *Middleware*. pp. 161–173. ACM (2017)
4. Ray, M., et al.: Scalable Pattern Sharing on Event Streams. In: *SIGMOD*. pp. 495–510 (2016)
5. Reisig, W.: *Understanding Petri Nets*. Springer (2013)
6. Weidlich, M., et al.: Optimizing Event Pattern Matching Using Business Process Models. *IEEE TKDE* 26(11), 2759–2773 (2014)
7. Wolf, K.: Petri net model checking with LoLA 2. In: *PETRI NETS*. LNCS 10877, pp. 351–362. Springer (2018)
8. Zhang, H., et al.: On complexity and optimization of expensive queries in complex event processing. In: *SIGMOD*. pp. 217–228. ACM (2014)