Complex Event Processing for Event-Based Process Querying

Han van der Aa(✉)

Department of Computer Science, Humboldt-Universität zu Berlin, Berlin, Germany
han.van.der.aa@hu-berlin.de

Abstract. Process querying targets the filtering and transformation of business process representations, such as event data recorded by information systems. This paper argues for the application of models and methods developed in the general field of Complex Event Processing (CEP) for process querying. Specifically, if event data is generated continuously during process execution, CEP techniques may help to filter and transform process-related information by evaluating queries over event streams. This paper motivates the use of such event-based process querying, and discuss common challenges and techniques for the application of CEP for process querying. In particular, focusing on event-activity correlation, automated query derivation, and diagnostics for query matches.

Keywords: Complex event processing · Process querying · Query derivation

1 Introduction

Process querying is concerned with models and methods to filter and transform representations of business processes [14]. As such, it supports various use cases, including process modeling support, variation management, performance simulation, and compliance verification. Although process querying is often performed based on model-based process representations, this paper focuses on querying techniques that target the abundance of event data recorded by information systems during the execution of business processes, i.e., *event-based process querying*.

The notion of *event data* relates to process representations that capture the recorded behavior of a process, such that an event denotes that a certain state has been reached (e.g., an order request has been received) or that an activity has been executed as part of a specific process instance [4]. Event data is often formalized as an *event log*, a set of traces, each trace being a finite sequence of events that denotes past behavior for a particular process instance. Process-related data may also be available as an *event stream*, a potentially infinite sequence of events that represent the ongoing behavior of a process.

Han van der Aa is funded by a fellowship from the Alexander von Humboldt Foundation.

C. Di Francescomarino et al. (Eds.): BPM 2019 Workshops, LNBIP 362, pp. 625–631, 2019.
https://doi.org/10.1007/978-3-030-37453-2_50

Complex Event Processing (CEP) defines models and methods to make sense of such event streams [7]. It defines languages to express queries, which are then evaluated over an event stream. CEP methods employ continuous filtering, transformation, and pattern detection, which are closely related to aspects of process querying. Therefore, it suggests itself to adopt event-based process querying through CEP when process-related information is represented by event streams. While CEP is developed for *online* event processing, event-based process querying also enables various use cases for *offline* event analysis. This is achieved by replaying event logs, which renders online event-based techniques applicable to static event data.

This paper outlines how CEP methods can be used for event-based process querying. It serves as accompanying material for the keynote presentation at the 4th International Workshop on Process Querying and comprises a shortened version of a book chapter on event-based process querying using CEP [2]. In the remainder, Sect. 2 motivates the use of event-based process querying, Sect. 3 briefly introduces CEP, Sect. 4 highlights essential challenges and techniques, prior to concluding in Sect. 5.

2 Motivation

Event-based process querying can be seen as a special variant of process querying, where the filtered and transformed process representations assume the form of event data. Given the event data of a process, event-based querying supports a variety of analysis questions, which may relate to qualitative as well as quantitative properties of a process.

Qualitative process properties relate to recorded execution dependencies [6], e.g., whether two activities have been executed in a specific order or a particular number of times. The analysis of such properties is, in particular, relevant to *compliance verification* of a process, in which recorded event data is compared to the expected behavior of a process, cf., [6,10]. Based on a formalization of compliance requirements, event-based process querying helps to identify cases of non-compliant process execution [22]. Such mechanisms are particularly useful if applied to event streams representing the most recent behavior of a process: Detecting a compliance violation shortly after it occurred enables the immediate implementation of mitigation and compensation schemes.

Quantitative process properties may be defined in terms of execution and wait times, or costs assigned to activity executions [17]. The analysis of such properties is part of *performance monitoring*, which recognizes the importance of efficient process execution for many processes. Event-based process querying helps to measure these properties: It selects events that are used as input for the computation of performance indicators [8], e.g., the average activity execution time, the delay with which a particular activity is executed after activation, or the accumulated costs induced by a specific type of process instance. At the same time, outliers that represent process execution with anomalous performance can be extracted [16]. Again, given that immediate detection of performance issues

is a prerequisite for effective countermeasures, the analysis of event streams performed by event-based querying can be highly beneficial.

3 Complex Event Processing

Complex Event Processing (CEP) emerged as a computational paradigm to handle streams of event data [7]. The focus of CEP is on the detection of specific event patterns, which are sets of events that are correlated in terms of their ordering, payload data, and context. Such event patterns are detected by applying *event queries* to *event streams*.

Events. An *event* is a recording of some state change that is considered to be of relevance for a particular setting. In the context of process querying, such state changes typically refer to the progress of process execution as, for instance, indicated by the execution of an activity in a process instance. Whereas events can be defined using different formalisms, events should at least be associated with a unique *identifier*, a *timestamp*, and an *event type*. Using a relational model for the payload of events, Table 1 lists three exemplary events for a Lead-to-Quote process. Each event is of type `Act` (representing that an activity has been executed), while an attribute `name` captures milestones (e.g., `QR` for quote request received) and activities (e.g., `ED` for entering details).

Table 1. Three example events for a Lead-to-Quote scenario.

id	timestamp	type	order_id	name	client	price
11	21.09.18,15:12:36	Act	O23	QR	Franklin	745,00
12	21.09.18,15:12:36	Act	O67	QR	Meyers	282,00
42	24.09.18,09:72:10	Act	O23	ED	Franklin	745,00

Event Streams. An *event stream* is defined by a potentially infinite set of events E and an order relation $\prec \subseteq E \times E$ (either partial or total). The fact that a stream is potentially infinite means that, in practice, processing is based on the stream at a specific point in time, i.e., the prefix of the stream up to this time. The identical timestamps of events `11` and `12` in Table 1 illustrate that events may happen concurrently (e.g., a batch of quote requests is received), inducing a partial order over the stream.

Event Query Languages. Numerous models and languages for the definition of event queries have been proposed in recent years (see [13] for an overview). There is currently no common standard for event query languages. Rather, CEP systems define their own languages, differing in syntax and semantics. Nevertheless, it has been noted that many languages for the specification of event queries, share at least a set of common operator types, such as *disjunction and conjunction*, *sequencing*, *Kleene closure*, and *data predicates* [23].

4 Challenges and Solutions

The application of CEP models and methods as discussed above in the context of event-based process querying has to cope with several challenges. This section considers three main challenges in this regard and discusses how state-of-the-art techniques address them. In particular, we focus on: event-activity correlation (Sect. 4.1), automatic event query derivation (Sect. 4.2), and diagnostics for event query matches (Sect. 4.3).

4.1 Event-Activity Correlation

A fundamental requirement for analysis techniques involving event data alongside other representations of a process, i.e., process models, is that observed event types can be linked to process model elements, such as activities or decision points. For instance, in compliance verification, the expected behavior of a process may be formalized by a process model, against which the recorded events are compared. Typically, however, such required event-activity correlation is not readily available [12]. *Manually* establishing correlation is often unfeasible because analysts rarely possess the necessary knowledge on the details of a process implementation [21]. Consequently, it is highly beneficial to establish event-activity correlation in an *automated* fashion. However, to reliably achieve this, challenges including noisy and non-compliant behavior, as well as complex event-activity relations must be taken into account [3].

Several techniques have been developed that aim to overcome such challenges, cf., [5,18]. These techniques consider various aspects to establish a proper correlation, most prominently analyzing label and behavioral information. The labels assigned to recorded events and process activities represent valuable information for the establishment of event-activity correlation, e.g., an event with the label *project information submitted* may correlate to an activity labeled *enter project details*. Correlation techniques employ similarity measures that quantify the similarity between different labels, considering both syntactic and semantic similarity. Behavioral information can, furthermore, be highly relevant when establishing event-activity correlation. Correlation techniques consider behavioral properties in different ways, for instance by quantifying similarity based on the average position in which events or activities occur in process instances [1] or comparing behavioral relations among events and activities [5].

4.2 Event Query Derivation

Establishing relevant event queries is a crucial aspect of event-based process querying to enable, for instance, compliance verification of a process. The actual translation of process properties into event queries is cumbersome, since it requires the formalization of the requirements in a (commonly declarative) query model. To overcome this problem, automatic techniques for query derivation have been developed. Two general directions are followed by such approaches: model-driven and data-driven derivation.

Model-Driven Query Derivation. If a specification of the expected behavior of a process is available in terms of a process model (and an event-activity correlation has been obtained), event queries may be formulated to detect any deviation of the recorded from the modeled behavior. These queries can be derived by first computing behavioral relations for the process model, e.g., using (causal) behavioral profile relations [20] or the relations of the 4C spectrum [15]. Such relations define constraints that should hold between process activities according to a process model. Given a set of behavioral relations, these can then be turned into monitoring queries, where each query identifies when the behavioral relation is not satisfied, i.e., when a compliance violation occurs.

Data-Driven Query Derivation. In case a suitable process model is not available, historic event data may serve as the basis for process querying when such data has been annotated with the situation of interest, e.g., a compliance violation or the attainment of a milestone. The necessary annotations can be obtained by retrospectively recording when such a situation of interest occurred in a process. While the annotations then identify the point in time at which the situation occurred, the actual event pattern leading to the situation is not necessarily known. The problem of event query discovery then becomes a supervised learning problem, which aims to construct a query that matches whenever an annotation indicates that the situation of interest occurred.

For sequential query patterns, query discovery can be framed as frequent sequence mining [19], detecting the subsequences that are shared among all annotated sequences. A few tailored algorithms have been proposed for more complex event query discovery, e.g., iCEP [11] and the IL-Miner [9], which discover queries built of sequence operators, data predicates, and time windows.

4.3 Diagnostics for Event Query Matches

The interpretation of results obtained by event-based process querying can be challenging. Query matches need to be interpreted as a violation of some normative behavior. Here, diagnostics are important, as fine-granular queries may lead to an overload of monitoring alerts for certain behavioral anomalies: e.g., a single out-of-order event may trigger a large amount of order violations, even though these all stem from the same source. To avoid such an overload, monitoring alerts can be filtered by identifying the earliest indicator of non-compliant behavior in a set of compliance violations, i.e., the *trigger* of the violations. Feedback on compliance violations can, furthermore, be filtered by recognizing violations that logically follow from the violations already observed, so-called *consecutive violations* [22].

Next to the identification of particular events that resulted in compliance violations, it is possible to assess whether there are dependencies between violations and their occurrence context as reflected in data attributes associated with process instances. That is, the goal is to check for attribute values that differentiate cases with the violation from cases without the violation. In this way, it is possible to discover that certain violations occur in a specific context, e.g., purchase orders originating from a particular *country* and which are

related to a specific *order type* may be more likely to be delivered too late. The automatic detection of such context-related issues can be achieved by applying *classification* techniques on an annotated set of process instances. Classifiers that produce human interpretable output, such as *decision trees*, are particularly useful for this setting. These techniques can produce clear rules indicating in which contexts monitoring violations have been observed.

5 Discussion

This paper outlined how Complex Event Processing methods can be leveraged for process querying that works on event-based representations of processes. In particular, event-based process querying can be used both for online querying, through the analysis of event streams, as well as for offline querying, by replaying event logs containing static event data. We argued that using CEP methods for event-based process querying faces several challenges, which may be addressed by three essential techniques: event-activity correlation, automatic event query derivation, and diagnostics for event query matches.

Open research directions in relation to the challenges discussed in this paper include: dealing with uncertainty in event-activity correlations, identifying the semantically most relevant monitoring queries, and optimizations of event query discovery algorithms in order to overcome scalability issues.

References

1. van der Aa, H., Gal, A., Leopold, H., Reijers, H.A., Sagi, T., Shraga, R.: Instance-based process matching using event-log information. In: Dubois, E., Pohl, K. (eds.) CAiSE 2017. LNCS, vol. 10253, pp. 283–297. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-59536-8_18
2. Van der Aa, H., Artikis, A., Weidlich, M.: Complex event processing methods for process querying. In: Process Querying Methods (2019, in press)
3. Van der Aa, H., Leopold, H., Reijers, H.: Efficient process conformance checking on the basis of uncertain event-to-activity mappings. IEEE TKDE (2019, in press)
4. Van der Aalst, W.M.P.: Process Mining - Data Science in Action. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49851-4
5. Baier, T., Di Ciccio, C., Mendling, J., Weske, M.: Matching events and activities by integrating behavioral aspects and label analysis. Softw. Syst. Model. **17**(2), 1–26 (2017)
6. Carmona, J., van Dongen, B.F., Solti, A., Weidlich, M.: Conformance Checking-Relating Processes and Models. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-99414-7
7. 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)
8. Del-Río-Ortega, A., Resinas, M., Cabanillas, C., Cortés, A.R.: On the definition and design-time analysis of process performance indicators. Inf. Syst. **38**(4), 470–490 (2013)
9. George, L., Cadonna, B., Weidlich, M.: IL-Miner: instance-level discovery of complex event patterns. PVLDB **10**(1), 25–36 (2016)

10. Ly, L.T., Maggi, F.M., Montali, M., Rinderle-Ma, S., van der Aalst, W.M.P.: Compliance monitoring in business processes: functionalities, application, and tool-support. Inf. Syst. **54**, 209–234 (2015)
11. Margara, A., Cugola, G., Tamburrelli, G.: Learning from the past: automated rule generation for complex event processing. In: DEBS, pp. 47–58. ACM (2014)
12. Oliner, A., Ganapathi, A., Xu, W.: Advances and challenges in log analysis. Commun. ACM **55**(2), 55–61 (2012)
13. Polyvyanyy, A.: Business Process Querying. In: Sakr, S., Zomaya, A. (eds.) Encyclopedia of Big Data Technologies. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-63962-8
14. Polyvyanyy, A., Ouyang, C., Barros, A., van der Aalst, W.M.P.: Process querying: enabling business intelligence through query-based process analytics. Decis. Support Syst. **100**, 41–56 (2017)
15. Polyvyanyy, A., Weidlich, M., Conforti, R., La Rosa, M., ter Hofstede, A.H.M.: The 4C spectrum of fundamental behavioral relations for concurrent systems. In: Ciardo, G., Kindler, E. (eds.) PETRI NETS 2014. LNCS, vol. 8489, pp. 210–232. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-07734-5_12
16. Rogge-Solti, A., Kasneci, G.: Temporal anomaly detection in business processes. In: Sadiq, S., Soffer, P., Völzer, H. (eds.) BPM 2014. LNCS, vol. 8659, pp. 234–249. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10172-9_15
17. Sadiq, S., Governatori, G., Namiri, K.: Modeling control objectives for business process compliance. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, pp. 149–164. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-75183-0_12
18. Senderovich, A., Rogge-Solti, A., Gal, A., Mendling, J., Mandelbaum, A.: The ROAD from sensor data to process instances via interaction mining. In: Nurcan, S., Soffer, P., Bajec, M., Eder, J. (eds.) CAiSE 2016. LNCS, vol. 9694, pp. 257–273. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-39696-5_16
19. Wang, J., Han, J.: BIDE: efficient mining of frequent closed sequences. In: ICDE, pp. 79–90. IEEE Computer Society (2004)
20. Weidlich, M., Polyvyanyy, A., Mendling, J., Weske, M.: Causal behavioural profiles-efficient computation, applications, and evaluation. Fund. Informaticae **113**(3–4), 399–435 (2011)
21. Weidlich, M., Ziekow, H., Gal, A., Mendling, J., Weske, M.: Optimizing event pattern matching using business process models. IEEE TKDE **26**(11), 2759–2773 (2014)
22. Weidlich, M., Ziekow, H., Mendling, J., Günther, O., Weske, M., Desai, N.: Event-based monitoring of process execution violations. In: Rinderle-Ma, S., Toumani, F., Wolf, K. (eds.) BPM 2011. LNCS, vol. 6896, pp. 182–198. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-23059-2_16
23. Zhang, H., Diao, Y., Immerman, N.: On complexity and optimization of expensive queries in complex event processing. In: SIGMOD, pp. 217–228. ACM (2014)