Efficient Context-Aware Nested Complex Event Processing over RFID Streams

Shanglian Peng[1,2(✉)] and Jia He[2]

[1] School of Information Science and Technology, Southwest Jiaotong University, Chengdu 610031, China
psl@cuit.edu.cn

[2] College of Computer Science, Chengdu University of Information Technology, Chengdu 610225, China
hejia@cuit.edu.cn

Abstract. With large scale of utilization of monitoring devices such as RFID, sensors and mobile phones, events are generated in a high-speed fashion. Decisions should be made in real time during business processes. Complex Event Processing (CEP) has become increasingly important for tracking and monitoring anomalies and trends in event streams. Nested event detection of RFID event stream is one of the most import class of queries. Current optimization of nested RFID event detection mainly considers caching intermediate results to reduce re-computation of similar results for nested subexpression. In this paper, we use context information of an RFID scenario to optimize nested event detection. We formalize context of an RFID scenario as spatial and temporal constraints and transform these constraints into rules over a nested NFA. Further, we present rewriting context rules to optimize nested event query plan. Experimental results show that with context information introduced, response time had been reduced greatly compared with counterpart methods.

Keywords: Context aware · Complex event processing · Nested pattern · NFA · Data stream · RFID

1 Introduction

Radio Frequency IDentification (RFID) has been extensively used in monitoring scenarios including logistics, health care monitoring, supply chain management and asset tracking, etc. These systems depend heavily on real time analysis of event streams to make decisions. Complex Event Processing (CEP) [1] has become one of the most critical technologies in an RFID enabled system. Application systems utilize CEP to work through many layers. Patterns are typically specified as regular expressions over event attributes, then predicates and correlations are defined over the patterns. So pattern queries can be complex (for example, pattern length can be large, pattern form can be sequential or nested, etc.), incurring great computational complexity to the CEP engine that are running the event queries.

S. Song and Y. Tong (Eds.): WAIM 2016 Workshops, LNCS 9998, pp. 125–136, 2016.
DOI: 10.1007/978-3-319-47121-1_11

The state-of-the-art CEP models such as SASE [2–5] and ZStream [6] do not support definition of nested queries. Though the Cayuga system [7,8] proposes complex nested queries, they process negation filter only over single primitive event type within the SEQ query. CEDR [9] allows applying negation operator over complex event types, but the authors do not present details of the execution model for such nested queries. NEEL [10–14] is a nested CEP language that supports nested operators of SEQ, AND, OR and Negation. The authors also present an iterative nested execution strategy for processing nested event queries expressed in NEEL. Also, the authors proposed caching to optimize execution of the nested CEP. However, as the query window (sliding window) slides continuously over the RFID event streams, instances valid for a certain sliding window are possibly valid in the next window. Although the authors propose caching and query sharing methods to reduce complex event processing overhead, we propose to introduce context during nested CEP query evaluation which is not considered in many current CEP engines.

In this paper, we aim to exploit context aware nested complex event processing over RFID event streams. We make the following contributions: (1) We introduce context model into definition of nested event queries. (2) To evaluate context aware nested CEP queries, we transform a context aware nested event query into corresponding Non-determined finite automata (NFA), context information is transformed into context constraints. (3) To reduce partial instances (partial matches), we propose efficient context aware query plan rewritten rules to optimize query execution. Experimental comparisons with methods proposed in NEEL over different data sets verify effectiveness of our method.

Organization of this paper is as follows: related complex event processing works are introduced in Sect. 2; the event model and context model used in this paper are presented in Sect. 3; the context-aware nested CEP evaluation model is described in Sect. 4; experimental studies of the proposed method compared with NEEL are shown in Sect. 5; finally, the work is concluded in Sect. 6.

2 Related Works

CEP has been extensively studied in active database [15,16]. There are many event processing engines with different evaluation models. For instance, SASE [2,3] utilizes NFA to evaluate an event query. In SASE, event queries are parsed into different NFAs, and for a coming event, it may trigger many runs of different instances. ZStream [6] evaluates event queries over a tree model. Concerning RFID CEP optimization algorithms, Hirzel et al. [17] utilizes partition constructs to parallelize event detection. Wu [18] partitions events into round-robin manner so that each operator has an access to a shared state. Schneider et al. [19] evaluate event processing queries across a cluster of machines based on Cayuga. NEEL [10–14] proposes to define and evaluate nested CEP queries systematically. But their optimization based on subexpression sharing and caching still do not work well in a sophisticated RFID scenario especially when the context is of critical in system monitoring.

Recently some work about context-aware event processing has been proposed. Opher et al. [20,21] describe an event processing framework with context support in a common event processing system and they also define many kinds of context in CEP which we would utilize in nested CEP definition. Kulkarni [22] proposes context aware CEP framework and methods which utilizes ontology to model context. Teymourian et al. [23] also introduce ontology and declarative rules into event processing engines to detect complex event more intelligently. Cao et al. [24] focus on context-aware distributed complex event processing in applications of Internet of Things, but their model is a probabilistic model. Most of the works on context-aware RFID CEP lack a detail model and evaluation of context in the CEP engines.

3 Event Model and Context Model

3.1 Event Model

In an RFID application, an event is defined as occurrence of a reading of an RFID over a RFID tag. RFID event is usually in the form of <RID, TID, timestamp, otherattributs>, where RID is the reader identifier, TID is the tag identifier, timestamp is the reading time of the tag, and otherattributs are other attributes of the event. If an event cannot be divided into smaller events, it is called a basic/simple event. For example, an RFID reading e1 = (Shelf1, Tag101, 2016-01-01 20:35:21) is a basic event that indicates a tag, namely, Tag101 is read by reader at Shelf1 at the time 2016-01-01 20:35:21. This event cannot be divided into smaller events. Event type is one of the attributes of a simple event that indicates a specific type of an event. Take the above RFID event. The RFID identifier Shelf1 is the event type which means where the event occurs. In this paper, we simply use uppercase letters such as A, B, or C to denote event type.

Complex event operators are used to connect basic/complex events in order to form a new complex event. Generally, complex event operators used in CEP include: logic AND, logic OR, NOT, SEQ (sequence). The NOT operator (always uses "!" in event definition) constructor is a unary operator, AND and OR operators are binary operators, SEQ operator defines the occurrence order of the events, for example SEQ(A, B, C) [1 h] defines a sequence of events of types A, B and C occur in order ABC within an hour. Here [1 h] is a sliding window. Sliding window defines the lifespan of an event existing during event processing. Sliding window can be considered as a temporal context in CEP.

3.2 Event Specification Language

In this paper, we use NEEL [10] as event specification language. To support context definition in pattern specification, we extend the NEEL language with HAVING clause. The language has the following overall structure:

```
[PATTERN <event pattern>]
[WHERE <qualification>]
```

```
[HAVING <pattern filtering condition, context list>]
[WITHIN <window>]
```

in which, the `PATTERN` clause contains a sequence construct in particular order, whose components are the occurrences or non-occurrences of primitive events; the `WHERE` clause filters events through predicate constraints which involve attribute value comparison; the `HAVING` clause specifies context definition; the `WITHIN` clause specifies the sliding window during which the whole sequence of events should occur.

3.3 Context Model

Within event processing languages, contexts may be explicit, implicit, or partially explicit. Explicit context means that context primitives are first class primitives in the language [21]. For example, in NEEL, the sliding window is a temporal context which is defined explicitly in a patter definition. Some languages do not support any notion of context, and some support partial notion of contexts. A survey of contexts in various languages can be found in [21]. [21] shows the context dimensions: temporal, spatial, state oriented, and segmentation oriented as shown in Fig. 1.

In this paper, we mainly focus on temporal and spatial context dimensions, other dimensions are our future work. Some temporal and spatial contexts can be

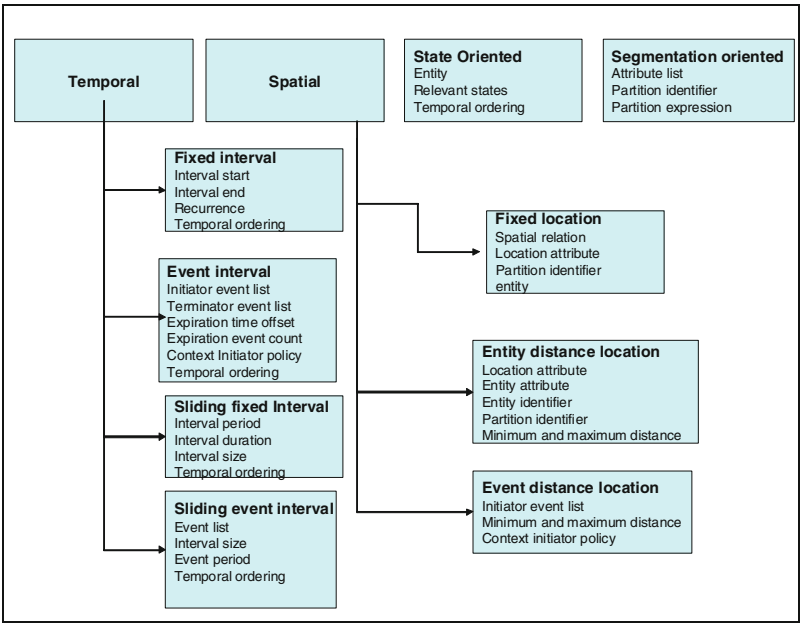


Fig. 1. Context dimensions [21]

explicitly found in event query specification in the form of event operator (SEQ) and clauses (where) indicating the time and spatial order of sub events in NEEL. But as shown in Fig. 1, not all the temporal and spatial context dimensions can be defined in NEEL without introducing the `HAVING` clause. Generally, there are four types of temporal contexts: fixed interval, event interval, sliding fixed interval and sliding event interval [20,21]. Fixed temporal interval context utilizes one or more temporal intervals which are defined as event timestamp constants. This context can be a one time interval: [June 6 2015 14:20, June 6 2015 17:00); This context also can be stated as [June 6 2015 14:20, + 1.5 h), where Te (end of the interval) is an offset relative to Ts (start of the interval) [20,21].

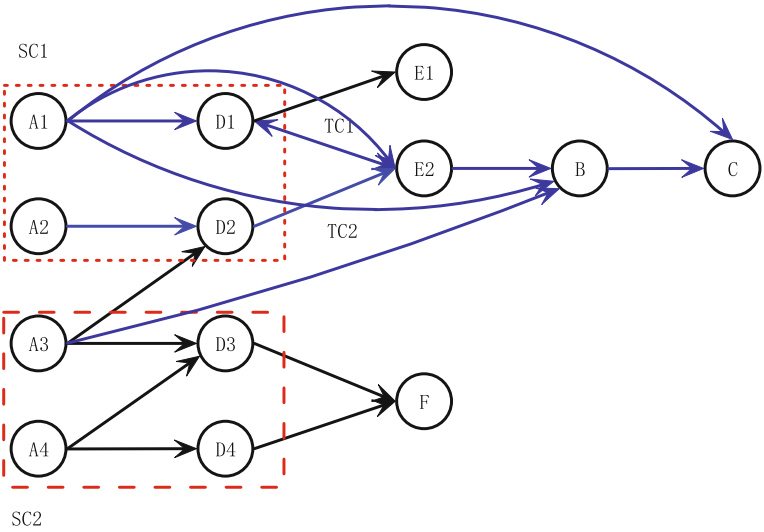
Event interval is a temporal interval which specifies the opened or closed time when one or more events occur. Note the meaning of "event occurs" is determined by the temporal ordering parameter and can be interpreted either as the detection time or the occurrence time as defined in pattern specification. The collection of events that open such an interval are called initiator (trigger) events, and the collection of events that close the temporal interval is called terminator events. An interval may also expire after a certain time offset is reached. For example, in RFID enabled hospital monitoring, temporal interval is initiated when an equipment is admitted to a hospital sanitation process and ends at the end of the sanitation process.

For sliding fixed interval context, windows are defined in NEEL using the `WITHIN` clause. New windows are sliding at regular intervals. Each window has a fixed size, defined either as a time interval (for example, 1 h) or a count of event instances (for example 1000 events).

For spatial context there are three classic types of spatial contexts: fixed location, entity distance location and event distance location [20,21]. Fixed location context partitions the location of a reference entity into geo-fence. An event instance is classified into a context partition if its location's attribute correlates with the spatial entity. Entity distance location context defines one or more context partitions, based on the distance between the event's location attribute and some other entity. Note that distance means the shortest distance between two spatial entities. Event distance location context specifies an event type and a matching expression predicate. For example, detecting the shopping activity of an item within a 10 h after detection of expiration of that batch of items.

3.4 Motivation Example

In this section, we present a motivation example to illustrate context aware nested CEP query. The scenario used in this paper is: suppose in an RFID enabled supply chain monitoring system, objects with an RFID tag move from one reading point to another point following pre-defined business flow, and the traces of different kinds of objects need to be monitored continuously in real time as shown in Fig. 2. Suppose the monitored query is shown as Query 1 in Fig. 2. Query 1 tends to detect complex events $SEQ(A; !AND(D, E); B; C)$ over RFID streams. This event query is a nested definition. In NEEL [10–14], there does not exist the "HAVING" clause, so evaluation of $SEQ(A; !AND(D, E); B; C)$ would




```
Query1:
PATTERN  SEQ(A; !AND(D,E); B; C)
WITHIN 30 minutes
HAVING  A in SC1[distance(Ai,Aj)<1km] AND
D in SC1[between(Ai,Ej)] AND
B in (TC1[T_B-T_D1<10min] or TC2[T_B-T_D2<20min])
```

Fig. 2. Motivation example of context aware nested CEP

be: first detect pattern $SEQ(A;B;C)$, and to verify if there exists complex pattern $AND(D,E)$, if there is a corresponding $AND(D,E)$, the complex event $SEQ(A;B;C)$ is deleted (not fulfilling the constraint), else output a complex event. Note that, during CEP, many partial matches need to be maintained until the end of the sliding window. In our motivation example, any event of type A would trigger a instance of the subexpression $SEQ(A;B;C)$ and any event of type D or C would also trigger instances of $AND(D,E)$. So there would exist many potential instances during evaluation of nested RFID event queries which would incur great system overhead in a high speed realtime decision making monitoring system. In NEEL [10–14], although authors propose optimization methods such as caching and partition to reduce partial matches. However, they does not consider context information during event detection, which, we think, would be critical in nested CEP evaluation. Note that in this scenario, we use A_i and D_i to represent different reading points of the same event types, respectively. Spatial and temporal contexts in this scenario are denoted as SC_i and TC_i. Funtions *distance*() and *between*() are spatial context operators. T_B and T_{D1} are used to denote arriving time of a specified event at reading point B and D1. Due to space limitation, some details are omitted.

4 Context Aware Nested Complex Event Detection Model

4.1 NFA Model of Context Aware Nested CEP

As a nested complex event is defined using declarative languages NEEL, NFA is a natural execution model which is widely used in many CEP engines. Figure 3 shows NFA of Query1.

As shown in Fig. 3, a nested CEP query is parsed into corresponding NFA. Event types are transformed into NFA states, sliding window constraint is transformed into time discrepancy between different event types. Negation nested subexpression is transformed into small group pattern. Note that spatial and temporal contexts are evaluated with the help of distance matrix and betweeness lists which we can predefine in an RFID scenario.

Generally, NFA-based CEP evaluation model is implemented into stack-based execution model. In this model, nested query evaluation suffers from several inefficiencies. First, partial results of SEQ(A;B;C)generated may be discarded later. Another potential overhead is that complete matches for the negation event AND(D;E) are constructed. Iterative execution method in NEEL does not solve these problems [10]. To overcome these problem, we utilize rewriting techniques to optimize context aware nested CEP queries.

4.2 Rewriting Rules for Context Aware CEP

As we introduce context into evaluation of nested event queries, we exploit rewriting context to reduce instance numbers and earlier partial matches pruning. Rewriting rules for nested event queries are described as follows.

Spatial-Share Rule. This rule transforms sequence operator with other operators in a single sequence operator if they have the same spatial context. Transformation follows the operators priorities. For Query1, $SEQ(A; !AND(D, E); B; C)$

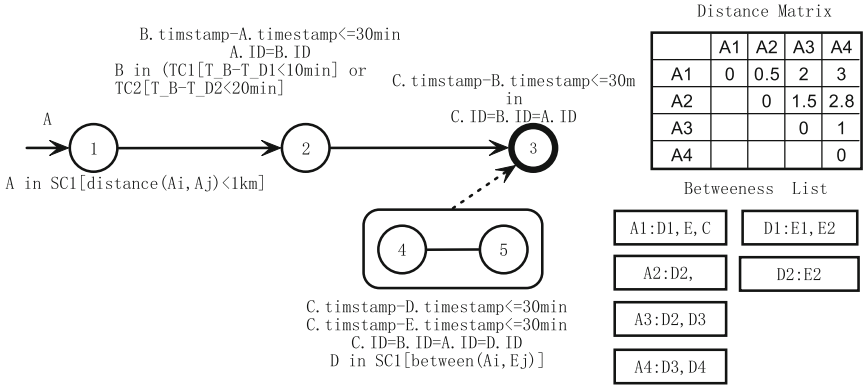


Fig. 3. NFA of the motivation example

is rewritten as $SEQ(SEQ(A;!D));!E;B;C)$. With this rewritten rule, instances that match SEQ(A;D) would be deleted earlier during event detection.

Temporal-Share Rule. This rule transform event types into the same temporal context when evaluation begins. For Query1, $SEQ(A;!AND(D,E);B;C)$, as we define temporal context over events of type B and D, so we transform Query1 into $SEQ(SEQ(A;!D;B[10\,\text{min}]));!E;C)$. This transformation means if events time discrepancy between A and B is less than 10 min, we would omit the evaluate of !AND(D,E); otherwise, we still need to wait for the verification of !AND(D,E) before output of complex event. Due to space limitation, we omit formal semantics of these rewriting rules.

5 Experimental Evaluation

To evaluate the proposed method, we have implemented a prototype CEP engine using C++. We compared our methods with the corresponding methods of NEEL. Comparisons are made between the iterative processing technique, the alternative caching techniques of NEEL on query execution time. Experimental event streams used in this paper is explained in the next section.

5.1 Experiment Data Description

As we do not have real RFID scenario event streams, we generated event stream in our motivation example. We simulate objects's moving route in an RFID enabled supply chain. Locations are denoted by (A,B,C,D,E, F). An object's moving route is simulated by a walk on the work flow graph consisting of these nodes. The elapsed time of moving between two locations are set with normal distribution within an interval $[t_1, t_2]$. In a supply chain, spatial and temporal context of some reading points are pre-defined and kept as matrix or list in the CEP engine. Nested event queries over RFID streams are generated with spatial and temporal context number changed with number 2, 4, and 6. We have utilized different operators in the nested subexpressions including AND, SEQ, OR and Negation, length of subexpression is set with 2, 4 and 6 respectively. The outer level of the event query is a SEQ operator. The sliding window is set at different size to compare these methods. For our context aware nested CEP method, we have implemented a stack-based method and a query rewritten based method.

5.2 Experimental Results

We measure evaluation efficiency with CPU time. We first generate event with fixed context number and fixed subexpression length. Experiment result is shown in Fig. 4. In this experiment, we set the sliding window as 100 events per slide. As we can see from Fig. 5, our method utilized context based query rewritten rules to optimize the query execution plan which greatly reduce CPU time. The NEEL-caching method performs better than our stack based method because

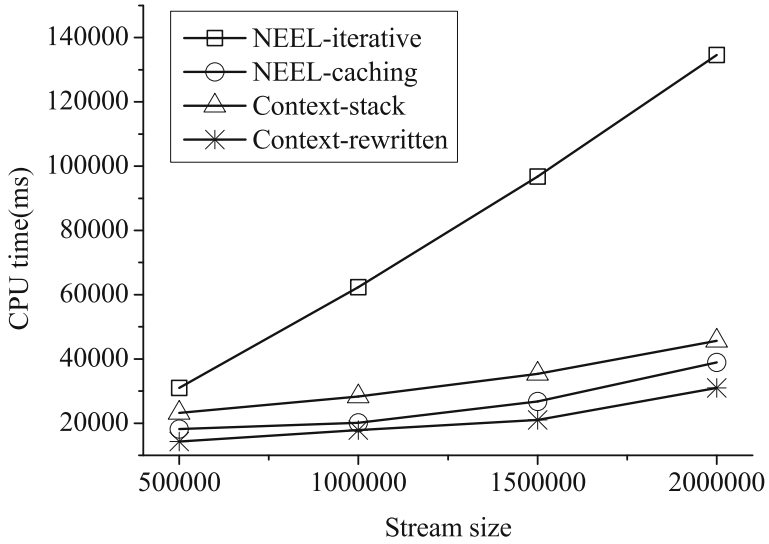


Fig. 4. Comparison with methods in NEEL (sliding window size 100, context number: 2, length of nested subexpression: 2)

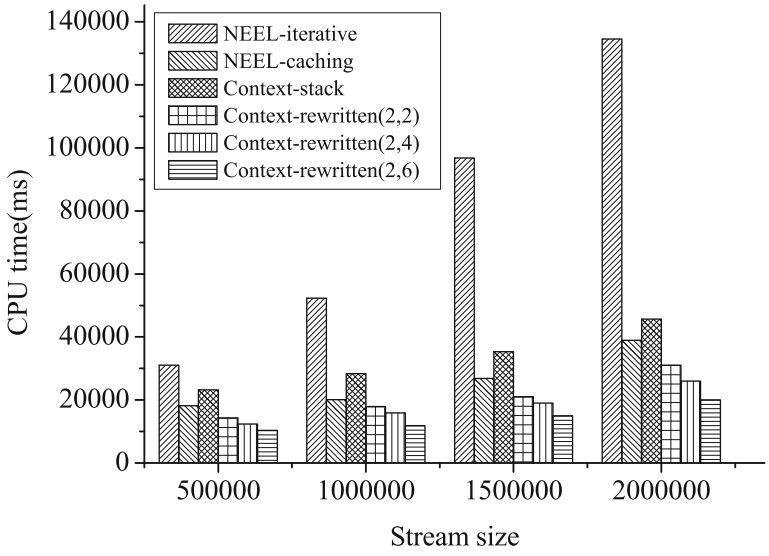


Fig. 5. Comparison with methods in NEEL (sliding window size 100, context number: 2, 4, 6, length of nested subexpression: 2)

caching utilizes effective partial match caching and discarding mechanism than our stack based method. We also tuned the context length and subexpression length parameters and generated different sizes of streams to evaluate these methods. Figures 5 and 6 are the performance comparisons with different para-

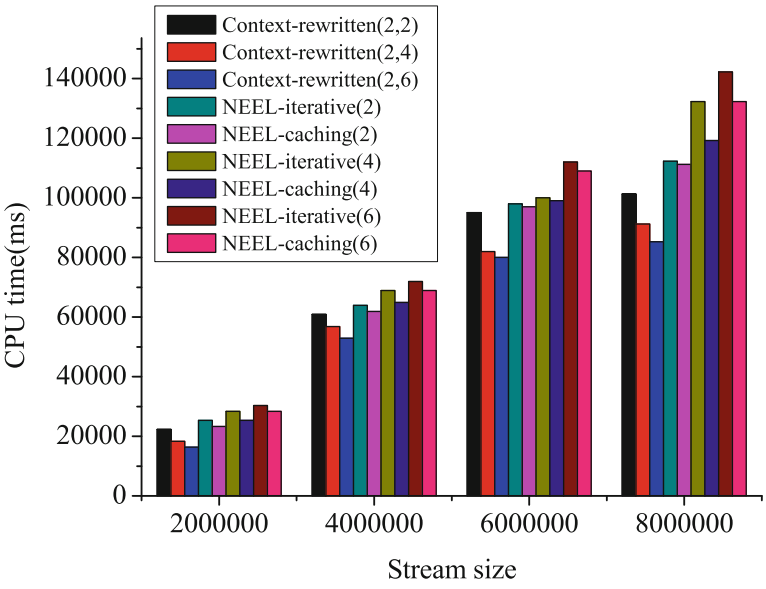


Fig. 6. Comparison with methods in NEEL (sliding window size 100, context number: 2, length of nested subexpression: 2, 4, 6)

meters. As we can see from Figs. 5 and 6, when context numbers and subexpression become larger, context based methods outperform other method because as context number increases, the instances generated and deleted during event detection are reduced greatly. When subexpression length become larger, partial matches that fulfill the subexpression pattern become less.

6 Conclusion

In this paper, we try to exploit context information to optimize evaluation of nested RFID complex event processing. We extend context semantic into NEEL language, and we propose transformation rules to incorporate context into the nested query evaluation model. Experimental analysis compared with methods proposed in NEEL verifies effectiveness of our proposed context aware method. However, some our work is not formalized in this paper, and we also need to experiment our method over some industry scenario and data sets.

References

1. Luckham, D.C.: The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems. Addison-Wesley, Boston (2002)
2. Wu, E., Diao, Y.L., Rizvi, S.: High-performance complex event processing over streams. In: SIGMOD, pp. 407–418 (2006)

3. Zhang, H., Diao, Y., Immerman, N.: Recognizing patterns in streams with imprecise timestamps. PVLDB **3**(1), 244–255 (2010)
4. Nie, Y., Cocci, R., Cao, Z., Diao, Y., Shenoy, P.J.: SPIRE: efficient data inference and compression over RFID streams. IEEE Trans. Knowl. Data Eng. **24**(1), 141–155 (2012)
5. Zhang, H., Diao, Y., Immerman, N.: On complexity and optimization of expensive queries in complex event processing. In: International Conference on Management of Data, SIGMOD 2014, Snowbird, UT, USA, 22–27 June 2014, pp. 217–228 (2014)
6. Mei, Y., Madden, S.: Zstream: a cost-based query processor for adaptively detecting composite events. In: SIGMOD (2009)
7. Brenna, L., Demers, A., Gehrke, J., et al.: Cayuga: a high-performance event processing engine (demo). In: SIGMOD (2007)
8. Demers, A., Gehrke, J., Hong, M., et al.: Cayuga: a general purpose event monitoring system. In: CIDR (2007)
9. Barga, R.S., Goldstein, J., Ali, M.H., Hong, M.: Consistent streaming through time: a vision for event stream processing. In: CIDR 2007, Third Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, 7–10 January 2007, Online Proceedings, pp. 363–374 (2007)
10. Liu, M., Rundensteiner, E.A., Dougherty, D.J., Gupta, C., Wang, S., Ari, I., Mehta, A.: High-performance nested CEP query processing over event streams. In: Proceedings of the 27th International Conference on Data Engineering, ICDE, 11–16 April 2011, Hannover, Germany, pp. 123–134 (2011)
11. Liu, M., Ray, M., Rundensteiner, E.A., Dougherty, D.J., Gupta, C., Wang, S., Ari, I., Mehta, A.: Processing nested complex sequence pattern queries over event streams. In: Proceedings of the Seventh International Workshop on Data Management for Sensor Networks, DMSN 2010, pp. 14–19. ACM, New York (2010)
12. Ray, M., Liu, M., Rundensteiner, E.A., Dougherty, D.J., Gupta, C., Wang, S., Mehta, A., Ari, I.: Optimizing complex sequence pattern extraction using caching. In: Workshops Proceedings of the 27th International Conference on Data Engineering, ICDE, 11–16 April 2011, Hannover, Germany, pp. 243–248 (2011)
13. Liu, M., Ray, M., Zhang, D., Rundensteiner, E.A., Dougherty, D.J., Gupta, C., Wang, S., Ari, I.: Realtime healthcare services via nested complex event processing technology. In: 15th International Conference on Extending Database Technology, EDBT 2012, Berlin, Germany, 27–30 March 2012, Proceedings, pp. 622–625 (2012)
14. Ray, M., Rundensteiner, E.A., Liu, M., Gupta, C., Wang, S., Ari, I.: High-performance complex event processing using continuous sliding views. In: Joint 2013 EDBT/ICDT Conferences, EDBT 2013 Proceedings, Genoa, Italy, 18–22 March 2013, pp. 525–536 (2013)
15. Chakravarthy, S., Krishnaprasad, V., Anwar, E., Kim, S.: Composite events for active databases: semantics, contexts and detection. In: VLDB, pp. 606–617 (1994)
16. Gatsiu, S., Dittrich, K.R.: Events in an active object-oriented database system. In: International Conference on Rules in Database Systems, pp. 23–39 (1993)
17. Hirzel, M.: Partition and compose: parallel complex event processing. In: DEBS, pp. 191–200. Citeseer (2012)
18. Wu, S., Kumar, V., Wu, K.L., Ooi, B.C.: Parallelizing stateful operators in a distributed stream processing system: how, should you and how much? In: Proceedings of the 6th ACM International Conference on Distributed Event-Based Systems, pp. 278–289. ACM (2012)
19. Schneider, S., Hirzel, M., Gedik, B., Wu, K.L.: Auto-parallelizing stateful distributed streaming applications. In: Proceedings of the 21st International Conference on Parallel Architectures and Compilation Techniques, pp. 53–64. ACM (2012)

20. Etzion, O., Magid, Y., Rabinovich, E., Skarbovsky, I., Zolotorevsky, N.: Context-based event processing systems. In: Helmer, S., Poulovassilis, A., Xhafa, F. (eds.) Reasoning in Event-Based Distributed Systems. SCI, vol. 347, pp. 257–278. Springer, Heidelberg (2011)
21. Etzion, O., Niblett, P.: Event Processing in Action, 1st edn. Manning Publications Co., Greenwich (2010)
22. Taylor, K., Leidinger, L.: Ontology-driven complex event processing in heterogeneous sensor networks. In: Antoniou, G., Grobelnik, M., Simperl, E., Parsia, B., Plexousakis, D., De Leenheer, P., Pan, J. (eds.) ESWC 2011, Part II. LNCS, vol. 6644, pp. 285–299. Springer, Heidelberg (2011)
23. Teymourian, K., Paschke, A.: Enabling knowledge-based complex event processing. In: Proceedings of the 2010 EDBT/ICDT Workshops, EDBT 2010, pp. 37:1–37:7. ACM, New York (2010)
24. Cao, K., Wang, Y., Wang, F.: Context-aware distributed complex event processing method for event cloud in internet of things. Adv. Inf. Sci. Serv. Sci. **5**(8), 1212 (2013)