Temporal Constraints for Rule-Based Event Processing

Karen Walzer
SAP AG
SAP Research CEC Dresden
Chemnitzer Strasse 48
01187 Dresden, Germany
Karen.Walzer@sap.com

Alexander Schill
Chair of Computer Networks
Dept. of Computer Science
University of Technology
Helmholtzstr. 10
01062 Dresden, Germany
Alexander.Schill@tu-dresden.de

Alexander Löser
SAP AG
SAP Research CEC Dresden
Chemnitzer Strasse 48
01187 Dresden, Germany
Alexander.Loeser@sap.com

ABSTRACT

Complex event processing (CEP) is an important technology for event-driven systems with a broad application space ranging from supply chain management for RFID, systems monitoring, and stock market analysis to news services. The purpose of CEP is the identification of patterns of events with logical, temporal or causal relationships out of single occurring events.

The Rete algorithm is commonly used in rule-based systems to trigger certain actions if a corresponding rule holds. It allows for a high number of rules and is therefore ideally suited for event processing systems.

However, traditional Rete networks are limited to operations such as unification and the extraction of predicates from a knowledge base. There is no support for temporal operators.

We propose an extension of the Rete algorithm for support of temporal operators. Thereby, we are using interval time semantics. We present the issues created by this extension as well as our pursued methodology to address them.

Categories and Subject Descriptors

D.1.6 [**Software**]: Logic Programming

General Terms

Algorithms

Keywords

Rete Algorithm, Event, Temporal, Rule-based

1. INTRODUCTION

Complex event processing (CEP) is the technology used to perform the detection of complex events consisting of combinations of single events. Complex event detection is the process of identifying patterns of events with logical, temporal or causal relationships out of the single event occurrences. A description language is used to specify the patterns of interest. This specification is also called *subscription* to a complex event. On the occurrence of the specified event pattern, an action such as the creation of a new event or the execution of certain function is performed.

CEP is an important component of current information-systems with possible application in supply chain management for RFID (Radio Frequency Identification), real-time stock trading and systems monitoring [31]. A growing number of companies are developing products in this space and many designs have been proposed for complex event processing systems, e.g. [23], [32], [26], [8].

The detection of complex events is difficult, since it requires a representation of the single events with the relations between them as well as the recording of the current state of a complex event, i.e. if a partial complex event has been detected it needs to be kept in storage until other events occur to complete it.

Complex event detection can be performed using different methods to represent the participating single events, the used operators and the state of detection, e.g. finite state automata [14], [27], colored Petri nets [13] or tree-based approaches [6] have been used in the past.

Time operators are not always supported in CEP systems. Current systems often only offer simple temporal conditions, e.g. the checking of a timestamp of a single event or the calculation of the duration between single events. If they exist, temporal conditions are rarely made explicit, but often hard-coded in the system and difficult to determine [33].

Data Stream Management Systems (DSMS) are a powerful alternative to traditional complex event processing systems, since they present an efficient means of detecting patterns in data streams [18]. They provide extensive temporal support offering sliding window operators as well as operations for evaluating timestamps or calculating duration, e.g. confer to [2].

However, traditional DSMS are not designed for event monitoring. It is possible to express event patterns, but the resulting queries are often difficult to optimize. Furthermore, in complex event processing, there is a large number of concurrent subscriptions that the event processing system has to deal with. But DSMS only support a small number of concurrent queries [9].

Pattern matching algorithms used in expert systems such as the Rete algorithm [11] are traditionally used to detect a

PIKM'07, November 9, 2007, Lisboa, Portugal.

high number of predicates (subscriptions). The efficiency of the Rete algorithm is independent of the number of rules[1] it contains, it is therefore ideally suited to process a high number of subscriptions for complex events.

Li and Jacobsen [19] first proposed using Rete for complex event detection. They showed that a rule-based event detection engine is very efficient and suitable for large scale publish/subscribe systems, matching a publication against 200,000 subscriptions in just 4.52ms in their system Padres.

Traditionally, Rete does not support temporal operators, but is limited to operations such as unification and predicate extraction [22]. However, a simple matching using comparisons with timestamp attribute values is possible. Extensions of the algorithm to allow for timeouts of operations have been suggested [3], however sophisticated temporal operators, e.g. for sliding windows, were not regarded so far.

1.1 Contributions

This thesis presents methods to extend the Rete algorithm with temporal operators for complex event detection. The focus is on the integration of a sliding windows operator in Rete. The developed concepts will be realized and evaluated in a prototype.

The main contributions of the thesis are:

- A flexible event-model is defined which is well suited for simple and complex events occurring in the field of business activity monitoring and in the manufacturing environment.
- The theoretical foundation of the temporal operators, esp. the sliding window operator, as well as efficient algorithms to realize these operators are outlined.
- The Rete algorithm is extended to be suitable for temporal event detection and an existing restricted description language is enriched with the necessary constructs to specify the temporal complex events.
- An efficient matching algorithm extending the Rete algorithm with sliding windows is developed. The algorithm supports interval-based event semantics and an automatic transformation from the description language into the respective Rete network.

1.2 Outline

The thesis proposal is structured as follows: We start with an introduction of the used terms in Section 2. This is followed by the presentation of a motivating example for our work in section 3. Then, we state the problems related to sliding window support in the Rete algorithm. Section 5 proposes our solution and discusses the possibilities to approach it. Finally, section 6 describes related work and section 7 summarizes the paper.

2. TERMS AND DEFINITIONS

Events signify a state change of a system. Complex events are logical, causal or temporal combinations of single events [20]. They are also referred to as *event composition* or *composite event*, since they aggregate single events.

[1]A rule defines a predicate and the action that should take place, if the predicate holds. This corresponds to defining a complex event and executing an action, if the complex event is detected.

Typical examples for operators and predicates used to combine the single events are [33]:

- logical operators: conjunction, disjunction, negation
- relative and absolute timing predicates: before (events occurring before a certain time frame), after (events occurring after a certain time frame), sequence (events occurring after each other), concurrency (events occurring at the same time), time boundaries (events occur within a certain time frame, e.g. a sliding time boundary)
- data predicates: comparison operators, e.g. $\leq, \geq, \neq$; aggregation operations, e.g. min, max, avg, sum; arithmetical operations, e.g. $+, -, *$

We adapt the notion of [18] for sliding windows. Our sliding window operator assigns a validity for a sequence of events based on a given time frame. A window size ω is defined. Then each element in the event sequence is valid for ω time units starting from the start of its corresponding validity frame.

3. MOTIVATION: FRIDGE COMPANY

In the following section, we present a supply chain scenario using RFID technology to show the relevance of sliding window support for a CEP system.

Today companies produce Just In Time (JIT) and Just In Sequence (JIS) to reduce their inventory and its associated costs. This increases dependence of the companies on the timely delivery of their suppliers which creates the need to respond immediately to arising problems. Nevertheless, existing production systems often lack the required transparency to identify risks early on, since problem detection and solution are manual processes.

RFID technology uses radio-frequency waves to communicate between readers and tagged objects. It allows for more automation and visibility than traditional identification technologies like the barcode, since it requires no line of sight between readers and tagged objects. Furthermore, it allows for bulk reading, thus offering the possibility for product-level tagging, i.e. a unique identification and tracking possibility for every product.

RFID data are time-dependent, they change dynamically, occur in large volumes, and carry implicit information, e.g. the detection of a tagged pallet at a location stands for the movement of the items that the pallet carries. Observations such as the reading of a tag are recorded and associated with a timestamp of their occurrence. Locations as well as hierarchical containment relationships change over time.

In supply chain applications, RFID is used to provide greater process transparency and efficient inventory management as well as to allow for product tracking and monitoring. It also reduces labor costs [29].

A sophisticated event-driven middleware can utilize RFID technology in order to detect supply shortages and exceptional situations. On the occurrence of complex events, a combination of RFID events and high-level business events, the necessary actions can be automatically triggered. This kind of middleware needs the possibility to support temporal operators for complex event definition to specify the situations that need a reaction. Fulfilling this requirement, the middleware allows for individual event-driven monitoring and control for JIT and JIS production.

Consider a fridge company producing 600 fridges in an 8 hour shift. Each fridge is equipped with an RFID tag. RFID readers are installed at the end of the assembly line and at the entry of the storage area in the company. The production manager is responsible for supervising the correct and on-time manufacturing of fridges. He wants to get automatically informed, if the amount of produced fridges in a time window of half an hour is below a certain threshold, e.g. 70. His business activity monitoring (BAM) software should inform him of this incident by sending him a message. This is achieved by defining the following rule:

> "If the sum of all occurring read events at the assembly line is smaller than 75 over the last 30 minutes, then send a message to the production manager."

This simple example illustrates how useful the support of a window operator is during event detection. A definition language for complex events should offer the possibility to express these kind of rules.

4. PROBLEM STATEMENT

The following section describes the occurring problems that the thesis addresses. First, the necessary extensions for the Rete algorithm to support complex event processing are briefly explained. Then, the problem of supporting temporal operators in Rete is outlined, focusing on the sliding window operator.

4.1 Rete for complex event processing

By default, the Rete algorithm considers a fact base and realizes rules which trigger certain action like a change of the fact base. It needs to be extended in order to react to events and to be able to trigger new events when actions are executed. Furthermore, the Rete algorithm is designed for high speed, but suffers from a high memory usage due to the creation of possibly large Rete networks.

It needs to be extended to only consider the occurring input coming from events and to discard events after a certain time span. The concept of only populating events and thus data changes is similar to Δ-dataflow networks as suggested by [22]. These networks only propagate changes of objects in order to reduce the high memory consumption of the networks. This approach limits the major drawback of Rete, its high memory consumption.

4.2 Sliding Windows in Rete

As stated before, the ability to detect temporal relationships is an important feature of CEP systems. Furthermore, detecting temporal relationships also leads to the detection of causal relationships, since a causal relationship implies a temporal relationship according to Luckham's cause-time axiom [20].

As illustrated in the example, support for sliding windows is a feature that CEP systems should offer. Even though, there are DSMS supporting this operator, it is not desired to always use DSMS to perform event detection, since rule-based algorithms like Rete can be of advantage when a high number of subscriptions is needed.

The Rete algorithm [11] is a pattern matching algorithm traditionally used for production-based logical reasoning systems. A set of rules is defined, each containing a premise (left-hand-side of the rule) and a set of actions. The algorithm creates an acyclic network of rule premises, also called Rete network where the nodes correspond to certain operations, e.g. a join combining two inputs. The leaves of the network correspond actions to be triggered. Facts from a knowledge base are propagated in a forward-chaining fashion through the network to determine the satisfied premises, consequently the actions of the rules whose premises hold are executed, one at a time. All newly asserted facts are passed through the network which stores the partial results in order to detect premises which are fulfilled by the occurrence of the new facts.

To illustrate the concept of a Rete network, figure 1 shows a simple network for a sample rule of our fridge company. In the fridge company, every 10 minutes, an event is created containing the number of defect fridges recorded in that time. The shown Rete network corresponds to the following rule:

> "If the number of defect fridges exceeds the number of three and the current production order is finished, then inform the production manager."

The two left hand-side nodes are the input of the network, namely the event containing the number of defect fridges and the *production order finished*-event. The node with the larger-than symbol performs the test whether the number of fridges exceeds the value three. If this is the case, the event is passed through the node, otherwise it is discarded. The JOIN node combines the *production order finished*-event with the result of the test node. If an event passes the test node and the *production order finished*-event is passed to the JOIN node, this node forwards its information to the end of the network which automatically triggers the action "send message".

A traditional Rete network is limited to operations such as unification and predicate extraction [22]. It does not provide temporal operators and therefore needs to be extended to allow for their usage.

Time windows are commonly used when only an excerpt of data is of interest. They also limit the memory requirements of stateful operators, e.g. in a join [15]. Therefore, means to extend the Rete algorithm to support the sliding window operator are needed.

However, the simple example shows already that it is not sufficient to add another operator node in order to realize the sliding window operator. The window operator aggregates the information from different consecutive states of the Rete network. This leads to the following problems to be dealt with:

- Actions are needed to observe the future states up to a limit given by a time window. These actions should be automatically created.
- There is chance that an additional action influences other networks, i.e. it can trigger another action. A mechanism to avoid the resulting actions influencing each other needs to be defined.
- The occurring events are equipped with a time interval holding the beginning and the end of the action that created the event. If the event is propagated through the network, it needs to be determined, if the beginning or the end time should be considered as limit

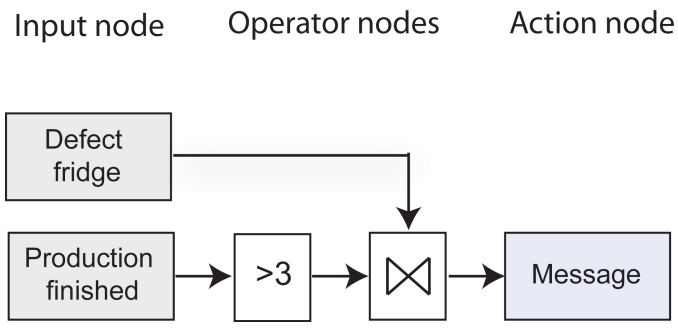

Figure 1: Sample Rete network

for the time window. Allen [1] identified thirteen possible combinations in which two events can occur in relation to each other when they are allowed to overlap, i.e. when time interval semantics is used. Consequently, the semantics of the window operator need to be specified explicitly or different alternatives for window definition need to be offered.

5. PROPOSED SOLUTION

5.1 Assumptions

We assume a loosely-coupled system with a global clock. The occurrence of a complex event can be viewed as the occurrence of a combination of single events across distributed systems. Lamport's happened-before relation holds. The single events are transmitted asynchronously to the central CEP engine which is based on the Rete algorithm. No messages are altered or spuriously introduced. A FIFO nature of channels is assumed.

Time is considered as an interval consisting of a start and end point for each event. This notion follows [4] and [12]. The alternative time-point semantic where each event only has one timestamp for its occurrence can be misleading and lead to incorrect event detection in the case of long-lasting events which cannot be excluded when regarding business processes.

5.2 Event Processing in Rete

A rule-based system using the Rete algorithm is extended to process occurring events instead by asserting them as facts to its knowledge base. It will discard events after a certain time span has passed when it can no longer be expected that they will occur in the near future. Furthermore, the actions to be triggered are adapted in order to trigger new events to be passed through the network.

5.3 Sliding Windows in Rete

It is possible to incorporate more operators into a Rete network by extending its nodes with new capabilities.

In the past, it was shown that it is possible to include clocks as an input node of the trigger network or as an addition to the operator nodes. Thus, one can measure certain temporal constraints, such as the distance between the occurrence of two input events or the absolute time of an event occurrence.

However, it is not possible to just add another operator node for aggregation functions like the sliding window operator, since they rely on more than one action, i.e. they do not rely on the current network state only, but also on past or future states.

Figure 2 presents a simple realization of a window operator in Rete. It realizes the following rule:

> "If the average of defect fridges exceeds the number of 3 in the next 5 minutes and the production orders are finished, then send a message to the production manager."

The figure shows input nodes with the events to be considered on the left and actions to be triggered on the right side. In between both are operator nodes performing joins or testing conditions. The letters "y" and "n" on the edges after condition nodes represent the evaluation of the condition to true or false respectively.

There are four input nodes on the left side of the network. Additional to the "production order finished"-event and the event containing the number of defect fridges, there is a timer to represent the passed time and an event with an attribute containing the average value of defect fridges.

The two upper joins check if both, the "production order finished"-event and the event with the defect fridges occurred. If so and the timer is smaller than the limit of 5 minutes, the defect fridges are summed up. If the timer is out of range, but still unequal zero, the average is calculated by dividing the current sum of defect fridges by the number of occurrences of the defect fridges event. The timer is set back to zero and in the next event propagation, it is checked, if the current average value is greater than three. If this is the case, a message is send to the production manager to inform him.

5.4 Validation

The main contribution of this work will be the development of the theoretical basis for the support of temporal operators in the Rete algorithm. The concepts will be evaluated by extending the rule engine JBoss Rules [17] with the operators and the ability to detect complex events by processing single events defined as JMS messages.

The following research approach is pursued:

Concepts and theoretical basis - current status A logical description of the operators is created.

The aim is the definition of an abstract conceptual framework containing the necessary concepts(agents) and relations required to support the temporal complex events in Rete. Furthermore, a class model, a

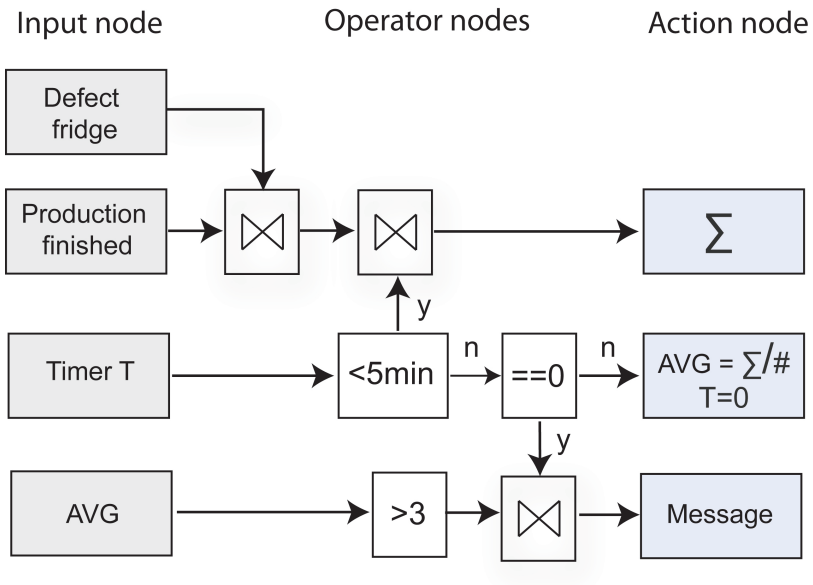

Figure 2: Sample Rete network with window operator realization

semantic description of the concepts and a definition of the constraints of the concepts is achieved.

New language constructs are added to the JBoss Rules-description language to allow for the specification of complex events using the operators.

Extending JBoss Rules for event processing The software JBoss Rules is extended to allow for event processing. This means, e.g. that sample input events are designed as well as sample transactions to be started.

Concepts and theoretical basis - algorithms The algorithms to support the temporal operators, incl. the sliding window operator, in Rete are defined. The aim is a theoretical description of how the selected operators can be realized using the Rete algorithm.

Extending JBoss Rules with new operators A physical description of the operators takes place, i.e. the implementation of the algorithms is carried out. The JBoss Rules description language is extended with the temporal operators. The focus of the algorithms is as in the Rete algorithm itself their run-time to detect a pattern, not their memory usage. Since the number of event producers constantly increases, performance is an important feature for event detection systems [22]. The result is an extension of JBoss Rules to support the temporal operators for event detection as well as to offer the possibility to describe the corresponding complex event subscriptions.

JBoss Rules prototype evaluation A centralized system generating events in a predefined manner is used to evaluate the prototype in order to demonstrate that the event detection using the extended Rete algorithm in JBoss Rules works. Performance and memory analyses of the algorithm are carried out and the extended algorithm is compared to an existing stream-based system with a high number of subscriptions, e.g. Coral8.

6. RELATED WORK

This section introduces related work in the area of event processing systems as well as temporal support in DSMS and the Rete algorithm.

6.1 Event Processing

In the field of distributed processing of single events substantial work has been done already resulting in systems such as SIENA [5], JEDI [7] or REBECCA [10], HERMES [25]. A comprehensive overview of existing work is given in [24].

6.2 Data Stream Management Systems

Complex events were first introduced in the field of active databases where events act as triggers to start certain database transactions.

Later, the developed concepts were applied in Streaming Database systems in order to detect patterns of interest there. In the following, suggested operators and approaches in this field are outlined.

Bai *et al.* [2] propose a stream query language for temporal event detection specifically for RFID data processing. They suggest a sequence operator detecting specific sequences from multiple streams where two event tuples have timestamps occurring after each other. They further describe sliding windows and tuple pairing modes on top of the sequence operator. Two operators are suggested to identify iterative events and exceptional event sequences. The work only considers timepoints, no intervals. However, this work can be a basis for the requirements of the extension of the Rete algorithm.

Wang *et al.* [30] describe *AND, OR* and *NOT* as non-temporal complex event constructors as well as periodic and aperiodic sequence operators with or without time distance constraints. Furthermore, interval-constrained events can be specified as temporal complex event constructors. Interval-based semantics are used, but the possibilities to express

events are limited, e.g. one cannot express the complex event that one event started a specific time after another event finished. Sliding windows are not supported.

In [29], a dynamic Entity Relationship (ER) model is suggested as a temporal extension of the traditional ER model. Dynamic relationships for location, containment relationship or reader location changes are modeled using relational databases. It is suggested to include lifespans (start and end timepoint) for relationships and timestamps for event-based dynamic relationships, e.g. observations. This is then mapped into a relational database management system. The model supports simple calculations using the timestamps, e.g. the time an object needs from one location to another is calculated by subtracting the end from the start time. More complex temporal functions need to be user-defined, e.g. the calculation of overlapping time-windows. The model does not allow for any sophisticated temporal operators, but utilizes standard SQL with timestamp attributes.

Krämer and Seeger [18] define a sound and well defined temporal operator algebra including a window operator. The logical operator algebra specifies the semantics of each operation in a descriptive way over temporal multisets, while the physical operator algebra provides adequate implementations in form of stream-to-stream operators.

6.3 Rete algorithm

It could be shown in the past that basic temporal support can be incorporated in Rete. Related work will be outlined in the following. However, none of the existing systems supports complex temporal operators, such as a sliding window operator.

Teodosiu and Pollak [28] present a method to remove obsolete temporal facts for a rule-based language for industrial production systems. The used Rete network is extended with timers in order to discard events after a specified time interval elapsed, similar to a garbage collector.

Temporal conditions are specified in the form of time intervals of fixed length between discrete events containing numerical timestamps which are derived from an internal system clock. Only relative temporal dependencies between events can be specified.

In [21], a traditional production system based on the Rete algorithm is extended with temporal reasoning by storing past and developing events in a temporal database, a so-called time map. An interval time representation is used. The system supports detection of events occurring *during, before* or *after* other events. It is further possible to model uncertain relationships. However, the semantics of the operators as well as the conceptual details remain unclear. It is not stated whether the starting or the end timepoint of the interval are used for the *before* and *after* operators.

The Padres [19] event processing system is based on the Rete algorithm. It can be studied as example for Rete-based event processing.

Berstel [3] defines extensions for the Rete algorithm for event management. The Rete algorithm is extended to incorporate clocks. Timestamps are used and *before* as *after* predicates are introduced. The idea of incorporating clocks can be reused by our extension.

Gordin and Pasik [16] describe a method to support set-oriented methods for forward chaining rule-based systems. The presented ideas may be used to support sliding windows in Rete.

7. SUMMARY

Rule-based systems such as those based on the Rete algorithm are ideal for the detection of a high number of complex events, consisting of logical combinations of single events or of events having a common context.

Traditionally, rule-based systems only offer limited support for detection of temporal constraints. The evaluation of timestamps is possible or the calculation of a duration between two successive events. Usage of temporal operators for data aggregation, as obtained by a sliding window operator, is not provided.

We propose the development of an extension of the Rete algorithm to include complex temporal operators, e.g. a sliding window operator. We will introduce the theoretical basis of the operators and develop algorithms to use them with other standard operators for logical or contextual combination of events.

This enhances the possibilities of complex event processing with the ability to detect temporal event patterns in order to create a temporal context for single event occurrences as well as to aggregate information carried by single events.

8. ACKNOWLEDGEMENTS

We wish to acknowledge Michael Ameling and Iris Braun for helpful comments on earlier drafts of this paper. Furthermore we thank Franz Weber, Sören Balko and Harald Schubert for the interesting discussions and the contribution to the presented concepts. Finally, we would like to thank Klaus Hänßgen for his support.

9. REFERENCES

[1] J. F. Allen. *Towards a general theory of action and time*, volume 23. Elsevier Science Publishers Ltd., Essex, UK, 1984.

[2] Y. Bai, F. Wang, P. Liu, C. Zaniolo, and S. Liu. RFID Data Processing with a Data Stream Query Language. In *ICDE*, pages 1184–1193, 2007.

[3] B. Berstel. Extending the RETE Algorithm for Event Management. In *TIME '02: Proceedings of the Ninth International Symposium on Temporal Representation and Reasoning (TIME'02)*, page 49, Washington, DC, USA, 2002. IEEE Computer Society.

[4] F. Bry and M. Eckert. Temporal order optimizations of incremental joins for composite event detection. In *Proceedings of Inaugural Int. Conference on Distributed Event-Based Systems, Toronto, Canada (20th–22nd June 2007)*. ACM, 2007.

[5] A. Carzaniga. *Architectures for an Event Notification Service Scalable to Wide-area Networks*. PhD thesis, Politecnico di Milano, Italy, December 1998.

[6] S. Chakravarthy and D. Mishra. Snoop: An expressive event specification language for active databases. *Data Knowl. Eng.*, 14(1):1–26, 1994.

[7] G. Cugola, E. D. Nitto, and A. Fuggetta. The JEDI event-based infrastructure and its application to the development of the OPSS WFMS. *IEEE Transactions on Software Engineering*, 27(9):827 – 850, September 2001.

[8] A. Demers, J. Gehrke, M. Hong, M. Riedewald, and W. White. Towards expressive publish/subscribe

systems. In *In Proceedings of the 10th International Conference on Extending Database Technology (EDBT 2006)*, Munich, Germany, March 2006.

[9] A. J. Demers, J. Gehrke, B. Panda, M. Riedewald, V. Sharma, and W. M. White. Cayuga: A general purpose event monitoring system. In *CIDR*, pages 412–422, 2007.

[10] L. Fiege, G. Mühl, and F. C. Gärtner. A modular approach to build structured event-based systems. In *Proceedings of the 2002 ACM symposium on Applied computing*, pages 385 – 392, Madrid, Spain, 2002.

[11] C. Forgy. Rete: A fast algorithm for the many patterns/many objects match problem. *Artif. Intell.*, 19(1):17–37, 1982.

[12] A. Galton and J. Augusto. Two approaches to event definition. In *Lecture Notes In Computer Science. Proceedings of the 13th International Conference on Database and Expert Systems Applications*, volume 2453, pages 547 – 556, 2002.

[13] S. Gatziu, A. Geppert, and K. R. Dittrich. The SAMOS active DBMS prototype. In *SIGMOD '95: Proceedings of the 1995 ACM SIGMOD international conference on Management of data*, page 480, New York, NY, USA, 1995. ACM Press.

[14] N. H. Gehani, H. V. Jagadish, and O. Shmueli. Composite event specification in active databases: Model & implementation. In *VLDB '92: Proceedings of the 18th International Conference on Very Large Data Bases*, pages 327–338, San Francisco, CA, USA, 1992. Morgan Kaufmann Publishers Inc.

[15] L. Golab and M. T. Özsu. Issues in data stream management. In *SIGMOD Record*, volume 32, pages 5–14, New York, NY, USA, June 2003. ACM Press.

[16] D. N. Gordin and A. J. Pasik. Set-oriented constructs: from Rete rule bases to database systems. In *SIGMOD '91: Proceedings of the 1991 ACM SIGMOD international conference on Management of data*, pages 60–67, New York, NY, USA, 1991. ACM Press.

[17] JBoss. *JBoss Rules*, 2007. http://labs.jboss.com/drools/.

[18] J. Krämer and B. Seeger. A temporal foundation for continuous queries over data streams. In *COMAD*, pages 70–82, 2005.

[19] G. Li and H.-A. Jacobsen. Composite subscriptions in content-based publish/subscribe systems. In *Middleware*, pages 249–269, 2005.

[20] D. Luckham. *The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems*. Addison Wesley Professional, May 2002.

[21] M. A. Maloof and K. Kochut. Modifying Rete to Reason Temporally. In *ICTAI*, pages 472–473, 1993.

[22] R. Manohar and K. M. Chandy. Delta dataflow networks for event stream processing. In *Proc. IASTED Parallel and Distributed Processing Symposium*, October 2004.

[23] M. Mansouri-Samani and M.Sloman. GEM: A generalized event monitoring language for distributed systems. *IEE/IOP/BCS Distributed Systems Engineering Journal*, 4(2):96–108, June 1997.

[24] G. Mühl, L. Fiege, and P. P. Pietzuch. *Distributed Event-Based Systems*. Springer-Verlag, Berlin Heidelberg, 2006.

[25] P. R. Pietzuch. *Hermes: A Scalable Event-Based Middleware*. PhD thesis, Queens' College University of Cambridge, 2004.

[26] P. R. Pietzuch, B. Shand, and J. Bacon. A Framework for Event Composition in Distributed Systems. In M. Endler and D. Schmidt, editors, *Proc. of the 4th ACM/IFIP/USENIX Int. Conf. on Middleware (Middleware '03)*, pages 62–82, Rio de Janeiro, Brazil, June 2003. Springer.

[27] C. Sánchez, M. Slanina, H. B. Sipma, and Z. Manna. Expressive completeness of an event-pattern reactive programming language. In *FORTE*, pages 529–532, 2005.

[28] D. Teodosiu and G. Pollak. Discarding unused temporal information in a production system. In *Proc. of the ISMM International Conference on Information and Knowledge Management CIKM-92*, pages 177–184, Baltimore, MD, 1992.

[29] F. Wang and P. Liu. Temporal management of RFID data. In *VLDB '05: Proceedings of the 31st international conference on Very large data bases*, pages 1128–1139. VLDB Endowment, 2005.

[30] F. Wang, S. Liu, P. Liu, and Y. Bai. Bridging physical and virtual worlds: Complex event processing for rfid data streams. In *EDBT*, pages 588–607, 2006.

[31] W. M. White, M. Riedewald, J. Gehrke, and A. J. Demers. What is "next" in event processing? In *PODS*, pages 263–272. ACM, 2007.

[32] E. Wu, Y. Diao, and S. Rizvi. High-performance complex event processing over streams. In *SIGMOD '06: Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, pages 407–418, New York, NY, USA, 2006. ACM Press.

[33] E. Yoneki and J. Bacon. Unified semantics for event correlation over time and space in hybrid network environments. In *OTM Conferences (1)*, pages 366–384, 2005.