Knowledge-Based Processing of Complex Stock Market Events

Kia Teymourian
Freie Universität Berlin
Berlin, Germany
kia@inf.fu-berlin.de

Malte Rohde
Freie Universität Berlin
Berlin, Germany
malte.rohde@inf.fu-berlin.de

Adrian Paschke
Freie Universität Berlin
Berlin, Germany
pascke@inf.fu-berlin.de

ABSTRACT

Usage of background knowledge about events and their relations to other concepts in the application domain, can improve the quality of event processing. In this paper, we describe a system for knowledge-based event detection of complex stock market events based on available background knowledge about stock market companies. Our system profits from data fusion of live event stream and background knowledge about companies which is stored in a knowledge base. Users of our system can express their queries in a rule language which provides functionalities to specify semantic queries about companies in the SPARQL query language for querying the external knowledge base and combine it with event data stream. Background makes it possible to detect stock market events based on companies attributes and not only based on syntactic processing of stock price and volume.

Categories and Subject Descriptors

C.2.4 [**Distributed Systems**]; C.4 [**Performance of Systems**]: Modeling techniques; I.2.4 [**Knowledge Representation Formalisms and Methods**]

Keywords

Complex Event Processing, Semantic Event Processing

1. MOTIVATION

The reality in many business organizations is that some of the important complex events cannot be used in process management, because they are not detected from the workflow data and the decision makers can not be informed about them. Detection of events is one of the critical factors for the event-driven systems and business process management. Semantic models of events can improve event processing quality by using event meta-data in combination with ontologies and rules (knowledge bases). The successes of the knowledge representation research community in building standards and tools for technologies such as formalized and declarative rules, are opening novel research and application areas. One of these promising application areas is *semantic event processing*.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
EDBT 2012, March 26–30, 2012, Berlin, Germany.
Copyright 2012 ACM 978-1-4503-0790-1/12/03 ...$10.00

Several complex event processing system are already proposed and developed[10]. Existing methods for event processing can be categorized into two main categories, rule-based approaches and non-rule-based approaches [15]. One of the processing models for CEP are non-deterministic finite state automata that are used in systems such as Cayuga and SASE+[1, 7], or ESPER[1]. One of the rule-based approaches is introduced in [14] which proposes a homogeneous reaction rule language for complex event processing. It is a combinatorial approach of event and action processing, formalization of reaction rules in combination with other rule types such as derivation rules, integrity constraints, and transactional knowledge. Also several event processing languages have been proposed such as Snoop, Cayuga Event Language, SASE, XChangeEQ[4, 1, 9, 3].

Some of the commercial CEP products are: TIBCO BusinessEvents, Oracle CEP, Sybase CEP[2] Some of these CEP systems can integrate and access external static or reference data sources. But these systems do not provide any inferencing on external knowledge bases and do not consider reasoning on relationships of events to other non-event concepts.

Also several data stream processing systems are proposed like Telegraph[6], Stream[8] which are targeting at handling continuous queries over high throughput data streams. These systems are also related to the event processing systems[5].

Previously, we proposed in [17, 16] a new approach for the Semantic enabled Complex Event Processing (SCEP). We claim that semantic models of events can improve the quality of event processing by using event stream data in combination with background knowledge about events and other related concepts in the target application domain. We described how to semantically query and filter events and how to formalize complex event patterns based on a logical knowledge representation (KR) interval-based event/action algebra, namely the interval-based Event Calculus [11, 12, 13].

In this paper, we describe a demonstration system for knowledge-based complex event processing to extract complex stock market events by using live stock market events and background knowledge about companies and other related concepts. Fusion of event data streams and background knowledge can build up a more complete knowledge about events and their relationships to other concepts.

The rest of this paper is organized as follows. In Section 2, we focus on use case scenario and show which kind of complex events can be detected using a background knowledge base. The use case is described by providing a concrete example. Section 3 describes our method for knowledge-based event processing which includes methods for data fusion with the background knowledge base. In Section 4 we describe our demonstration system in details

[1]Esper: `http://esper.codehaus.org`
[2] `http://www.tibco.com/` `http://www.oracle.com` `http://www.sybase.de`

and provide an other example.

2. USE CASE SCENARIO

Consider that Mr. Smith is a stock broker and has access to a stock exchange event stream like listed in Listing 1. He is interested in special kinds of stocks and would like to be informed if there are some interesting stocks available for sale. His special interest or his special stock handling strategy can be described in high level language which describe the interest using background knowledge about companies.

Listing 1: Stock Exchange Event Stream

```
{ .. , {(Name, ''GM'')(Price , 20.24)(Volume, 8,835)},
       {(Name, ''SAP'')(Price , 48.71)(Volume, 8,703)},
       {(Name, ''MSFT'')(Price , 24.88)(Volume, 46,829)},
  ... }
```

Mr Smith would like to start a query on the event stream similar to the following query:

> **Buy** Stocks of Companies, **Who** have *production facilities in Europe* **and** produce products from *Iron* **and** have more than *10,000 employees* **and** are at the moment in *reconstruction phase* **and** their price/volume *increased stable* in the *past 5 minutes*.

As we can see the above query cannot be processed without having background knowledge which can define the used concepts in this query. Mr. Smith needs an intelligent system which can use background knowledge about companies like listed in Listing 2. This background knowledge should be integrated and processed together with the event data stream in a real-time manner so that interesting complex events can be timely detected.

We can also consider that Mr. Smith works for a company and may need to share this knowledge base with other brokers. Each of these brokers may be able to gather new information about companies and update this knowledge base, e.g., the Opel company is not in reconstruction phase, or the Apple company has a new chief executive officer.

Listing 2: An Excerpt of Knowledge Base which Store Background Knowledge about Companies.

```
{ (OPEL, belongsTO, GM ), (OPEL, isA, automobilCompany),
  (automobilCompany, build, Cars), (Cars, areFrom, Iron),
  (OPEL, hatProductionFacilitiesIn, Germany),
  (Germany, isIn, Europe),
  (OPEL, isA, MajorCorporation),
  (MajorCorporation, have, over10,000employees),
  (OPEL, isIn, reconstructionPhase), ... }
```

3. SEMANTIC EVENT PROCESSING

The fusion of background knowledge with the data from an event stream can help the event processing engine to know more about incoming events and their relationships to other related concepts. We propose to use an external knowledge base which can provide background conceptual and assertional information about the events as it is shown in Figure 1. This means that events can be detected based on reasoning on their type hierarchy relationships, or temporal/spatial relationships. It can also be based on their connections to other relevant concepts from the domain, e.g., relationship of a stock price to the products or services of a company.

The realization of SCEP is a challenging task, because it should provide real-time processing and high scalability. The naïve approach for SCEP might be a storage-based approach. This means

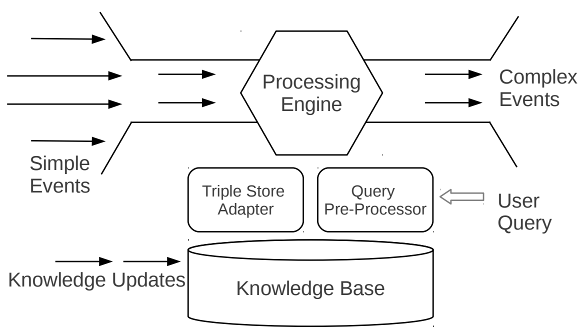

Figure 1: High Level Architecture of Semantic-Enabled Complex Event Processing

to store all of the background knowledge in knowledge bases and start polling the knowledge base, every time when a new event comes into the system, and then process the result from the external knowledge base with event data. This approach may have several problems when the throughput of the event stream is high, the size of background knowledge is high, or even when expressive reasoning should be done on the knowledge base.

3.1 Event Query Pre-Processing

We propose to do an Event Query Pre-Processing (EQPP) before the event processing is done on the event stream. In this approach, the original complex event query can be pre-processed by use of a knowledge base and rewritten into a single *new query*. This *new query* is a query which can be syntactically processed only with the knowledge from the event stream and without an external knowledge base.

In this paper, we are addressing a simple pre-processing of event queries and illustrate the potential of such a pre-processing approach for SCEP. In our method the user query is pre-processed and rewritten into a single new query which has the same semantic meaning as the original one. The advantage of this method is that the user can define event queries in a high level abstraction view and does not need to care about some details, e.g., the user can specify queries like, *"companies who produce products from iron"* and does not need to know all of the products of companies which might not be simple for humans to remember. One other advantage is that the SCEP system is able to provide real-time event processing as events arrive into the system because the external reasoning on knowledge base is done in advance. On the other side, one dist-advantage of this approach is that the query needs to be updated each time when the knowledge base is changed (or when a part of the KB is changed). We assume that in some of the use cases the rate of background knowledge updates is not as high as the rate of the main event stream updates. For example the number of news about an specific stock, e.g., MSFT are not as much as the number of incoming trade events about that stock.

4. DEMONSTRATION

The architecture of our implementation is shown in Figure 2, it shows event data stream, a main processing engine and a knowledge base which stores background knowledge about events.

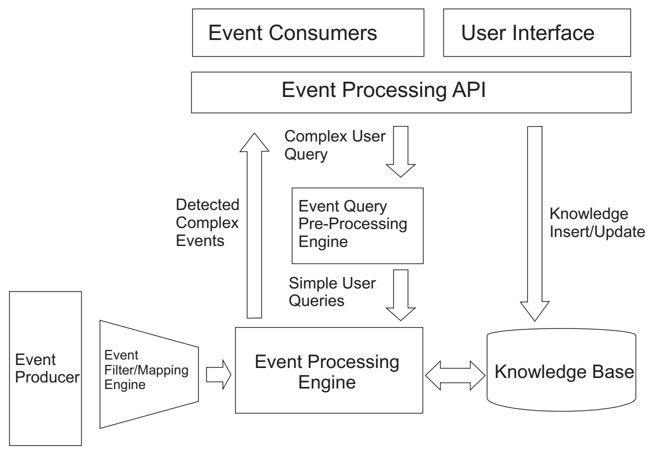

Figure 2: Architecture of our SCEP Implementation

Listing 3: Prova Example for Semantic Event Detection.

```
:- eval(server()).
server() :-
   sparqlrule(QueryID),
   rcvMult(XID, Protocol, Sender, event,
{time->Time, symbol->Symbol, name->Name, lastprice ->
      Lastprice,
volume->Volume, high->High, low->Low}),

sparql_results(QueryID, CompanySymbol, CompanyEmployees)
[Symbol = CompanySymbol, CompanyEmployees > 50000 ],

println(['Found: ', Symbol]),
sendMsg(XID, Protocol, Sender, testrule, {name->Name}).

sparqlrule(QueryID) :-

Query = '
 PREFIX DBPPROP: <http://dbpedia.org/property/>
 PREFIX DBPEDIA: <http://dbpedia.org/resource/>
 PREFIX CSW: <http://corporate-semantic-web.de/scep/>

 SELECT ?symbol ?employees WHERE {
  ?company DBPPROP:industry DBPEDIA:Computer_software .
  ?company CSW:traded_as ?symbol .
  ?company DBPPROP:numEmployees ?employees .
       } ',
sparql_select(Query, QueryID, [],
'http://dbpedia.org/sparql').
```

For our implementation, we use Prova[3] as a reaction rule language formalization and as a rule-based execution which can be used as an event processing engine. In the Listing 3, we provide an excerpt of the Prova code example which illustrate our implementation. In this query, a broker is interested in software companies which have more than 50000 employees.

Prova uses reactive messaging[4], reaction groups, and guards[5] for complex event processing. Multiple messages can be revived using revMult(XID, Protocol, Destination, Performative, Payload) ; XID, a conversation id of the message; Protocol, message passing protocol; Destination, an endpoint; Performative, message type; Payload, the content of message. Prova implements a new inference extension called literal guards. During the unification only if a guard condition evaluates to true, the target rule will proceed with further evaluation.

We implemented the *sparql_select* built-in[6] to run SPARQL queries from Prova which can start a SPARQL query from inside Prova on an RDF file or a SPARQL endpoint. The built-in injects the results of the SPARQL query as facts into the Prova knowledge base in order to be able to separate retrieval and use of the information. Those facts can then be accessed by using the *sparql_results* predicate. This has several advantages over pulling the results directly into the unification process after issueing the query. First, this way the *sparql_select* call will not result in several branches in unification and thus succeeding statements will not get evaluated/executed several times. This is especially important for follow-up calls to reactive messaging built-ins such as *rcvMult*. Second, it allows us to gradually feed results into the Prova knowledge base, for example when data in the external knowledge base changes.

The *sparql_select* built-in has the following syntax:
sparql_select(QueryString, QueryID, [SetOfInputVariables], ServiceEndpoint).

The QueryID is used to associate the query with the corresponding results, it can either be provided by the user or will be generated by the built-in. The set of input variables provide the possibility to replace variables in SPARQL string which are starting with $ with variables in Prova, and the service endpoint is the SPARQL endpoint.

We created a light-weight ontology for companies as shown in Figure 3, our system is able to extract knowledge about a specific company from DBpedia. A stock market event of company can be detected based on the properties for concept company. The knowledge base can be updated by the users.

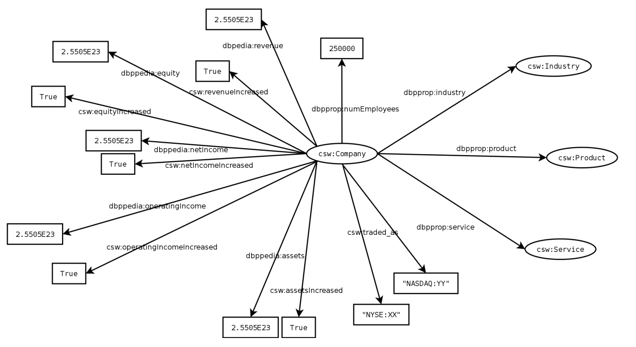

Figure 3: A Simple Company Ontology

The complete pre-processing step should be updated on the knowledge base, whenever there is a change in the knowledge base, e.g., if new products are added to the product lists of a company. In many use case like ours, the frequency of such updates can be considered not to be very high. Here, one useful approach is to implement the updates also in an event-based manner, if any relevant changes are done on the knowledge base a notification informs the event processing engine to update the event query.

Prova follows a workflow paradigm in event processing. It is possible to use Prova for the realization of Plan-based complex

[3]Prova, ISO Prolog syntax with extensions `http://prova.ws`
[4]Prova Reactive Messaging `http://www.prova.ws/confluence/display/RM/Reactive+messaging`
[5]Event Processing Using Reaction Groups `http://www.prova.ws/confluence/display/EP/Event+processing+using+reaction+groups`

[6]Source codes for Semantic Web extensions in Prova 3 can be found in `https://mandarax.svn.sourceforge.net/svnroot/mandarax/prova3/prova-compact/branches/prova3-sw/`

event detection across distributed event sources [2]. But in our experiments, we assume that all of the event streams come to a central processing point.

Our demonstrations shows that the EQPP can achieve a better performance than the naïve storage-based approach (or polling approach). They also show that the EQPP approach is an applicable approach for the above described use case.

It shows also that the scalability of SCEP systems has five different dimensions;

- Discharge rate of events,
- Number of rules in main memory,
- Number of triples in the knowledge base (amount of knowledge),
- Rate of knowledge updates,
- Expressive level of reasoning on background knowledge.

Our demonstration system can be found online at `http://slup.imp.fu-berlin.de/scepdemo/`.

The user interface of our demonstration system consist of four parts, each of them have different functionalities; The first part is *CEP Engine Status*, the user can see the current status of the system, can start or stop the CEP engine. The second part is called *Event Query*, the user can give an event query in form of a Prova rule. The third part is the *Knowledge Base* of the system, in this part the user can specify a SPARQL endpoint as external knowledge base, can send SPARQL queries to the end point to see what is available on the knowledge base and the user can also post and add RDF data to the external RDF store. The last part is the configuration of the event sources, the user can select between live stream of stock market from the Yahoo finance system or a stored stream from a database.

5. CONCLUSION AND OUTLOOK

We have described our initial work on semantic event processing and semantic pre-processing of event queries and illustrated the potential of this approach by use of a demonstration. Our future steps are to work on semantics of event processing languages and define which semantics can be adequate semantic for Complex Event Processing. Furthermore, we are working on an algorithm for the rewriting of complex event queries to several simple queries which can be distributed on an event processing network to achieve high performance and scalability.

6. ACKNOWLEDGEMENTS

This work has been partially supported by the "InnoProfile-Corporate Semantic Web" project funded by the German Federal Ministry of Education and Research (BMBF) and the BMBF Innovation Initiative for the New German Länder - Entrepreneurial Regions.

7. REFERENCES

[1] Jagrati Agrawal, Yanlei Diao, Daniel Gyllstrom, and Neil Immerman. Efficient pattern matching over event streams. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, SIGMOD '08, pages 147–160, New York, NY, USA, 2008. ACM.

[2] Mert Akdere, Uğur Çetintemel, and Nesime Tatbul. Plan-based complex event detection across distributed sources. *Proc. VLDB Endow.*, 1:66–77, August 2008.

[3] François Bry and Michael Eckert. Rule-based composite event queries: The language xchangeeq and its semantics. In *Proceedings of First International Conference on Web Reasoning and Rule Systems, Innsbruck, Austria (7th–8th June 2007)*, volume 4524 of *LNCS*, pages 16–30, 2007.

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

[5] Sharma Chakravarthy and Qingchun Jiang. *Stream Data Processing: A Quality of Service Perspective Modeling, Scheduling, Load Shedding, and Complex Event Processing*. Springer Publishing Company, Incorporated, 1st edition, 2009.

[6] Sirish Chandrasekaran, Owen Cooper, Amol Deshpande, Michael J. Franklin, Joseph M. Hellerstein, Wei Hong, Sailesh Krishnamurthy, Samuel R. Madden, Fred Reiss, and Mehul A. Shah. Telegraphcq: continuous dataflow processing. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, SIGMOD '03, pages 668–668, New York, NY, USA, 2003. ACM.

[7] Alan Demers, Johannes Gehrke, Mingsheng Hong, Mirek Riedewald, and Walker White. Towards expressive publish/subscribe systems. In *10. International Conference on Extending Database Technology*, pages 627–644, Munich, Germany, 2006.

[8] The STREAM Group. Stream: The stanford stream data manager. Technical Report 2003-21, Stanford InfoLab, 2003.

[9] Daniel Gyllstrom, Eugene Wu 0002, Hee-Jin Chae, Yanlei Diao, Patrick Stahlberg, and Gordon Anderson. Sase: Complex event processing over streams. *CoRR*, abs/cs/0612128, 2006.

[10] Alessandro Margara and Gianpaolo Cugola. Processing flows of information: from data stream to complex event processing. In *Proceedings of the 5th ACM international conference on Distributed event-based system*, DEBS '11, pages 359–360, New York, NY, USA, 2011. ACM.

[11] A. Paschke. Eca-lp / eca-ruleml: A homogeneous event-condition-action logic programming language. In *RuleML-2006*, Athens, Georgia, USA, 2006.

[12] Adrian Paschke. Eca-ruleml: An approach combining eca rules with temporal interval-based kr event/action logics and transactional update logics. *CoRR*, abs/cs/0610167, 2006.

[13] Adrian Paschke and Martin Bichler. Knowledge representation concepts for automated sla management. *Decis. Support Syst.*, 46(1):187–205, 2008.

[14] Adrian Paschke, Alexander Kozlenkov, and Harold Boley. A homogeneous reaction rule language for complex event processing. *CoRR*, abs/1008.0823, 2010.

[15] Kay-Uwe Schmidt, Darko Anicic, and Roland Stühmer. Event-driven reactivity: A survey and requirements analysis. In *SBPM2008: 3rd international Workshop on Semantic Business Process Management in conjunction with the 5th European Semantic Web Conference (ESWC'08)*. CEUR Workshop Proceedings (CEUR-WS.org, ISSN 1613-0073), June 2008.

[16] Kia Teymourian and Adrian Paschke. Semantic rule-based complex event processing. In *RuleML 2009: Proceedings of the International RuleML Symposium on Rule Interchange and Applications*, 2009.

[17] Kia Teymourian and Adrian Paschke. Towards semantic event processing. In *DEBS '09: Proceedings of the Third ACM International Conference on Distributed Event-Based Systems*, pages 1–2, New York, NY, USA, 2009. ACM.