Processing of Complex Stock Market Events Using Background Knowledge

Kia Teymourian, Malte Rohde, and Adrian Paschke

Corporate Semantic Web Research Group
Institut for Computer Science, Freie Universität Berlin
http://www.inf.fu-berlin.de/groups/ag-csw/
{kia, malte.rohde, paschke}@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 streams and background knowledge about companies which are 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 RDF SPARQL language for querying the external knowledge base and combine it with event data streams. 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.[1]

Keywords: Complex Event Processing, Knowledge-Based Complex Event Processing

1 Motivation

The reality in many business organizations is that some of the important complex events can not 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 metadata 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*.

Several complex event processing systems are already proposed and developed [4]. Existing methods for event processing can be categorized into two main categories, logic-based approaches and non-logic-based approaches [9]. One of the logic-based

[1] 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.

approaches is introduced in [8] 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.

Examples of commercial CEP products are: TIBCO BusinessEvents[2], Oracle CEP [3], Sybase CEP[4] Some of these existing CEP systems can integrate and access external static or reference data sources. However, 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. Previously, we proposed in [11, 10] 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 interval-based event/action algebra, namely the interval-based Event Calculus [5–7]. Other related approaches like [2, 3] are also relevant for our approach, but these approaches are not combining the complex event detection based on event correlations and detection semantics with the relationships between events and other non-event concepts/individuals in the background knowledge base.

In this paper, we describe a demonstration system for knowledge-based complex event processing to extract complex stock market events 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 and provide an other example.

2 Use Case Scenario

Stock market brokers gather information from different parties and monitor different stock market graphs to be able to make the best possible stock market handling strategy. They have to be able to make the right decision at the right time. They get all of the "chunks" of information and are face with the difficult tasks of mentally/intuitively combining them together, enriching/aggregating and inferencing on them. The vision of our system is to have real-time information processing support system for stock market brokers.

Consider that Mr. Smith is a stock broker and has access to stock exchange event stream like listed in Listing 1.1. He is interested in special kinds of stocks and would

[2] `http://www.tibco.com/`
[3] `http://www.oracle.com`
[4] `http://www.sybase.de`

like to be informed if there are some interesting stocks available for purchasing. His special interest or his special stock handling strategy can be described in a high level language which describes his expressed interest using background knowledge about companies.

Mr Smith would like to start the following query on the event stream: 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*.

Listing 1.1: 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)} , ... }
```

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 1.2. This background knowledge should be integrated and processed together with event data stream in 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 the 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 1.2: An Excerpt of a Knowledge Base about Companies

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

3 Semantic Enabled 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 to store all of the background knowledge in knowledge bases and start pulling 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,

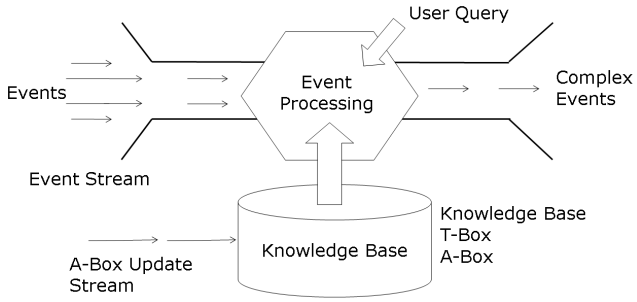

Fig. 1: High Level Architecture of Semantic-Enabled Complex Event Processing

the size of background knowledge is high, or even when expressive reasoning should be done on the knowledge base.

Event Query Pre-Processing:

We propose to do an Event Query Pre-Processing (EQPP) before the event processing is down 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 illustrates 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 disadvantage 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 most of the use cases the rate of background knowledge updates is not very high as the rate of the main event stream.

4 System Architecture and Demonstration

In this section we describe the architecture of our system and describe how our demonstrator works. The architecture of our implementation is shown in Figure 2, it shows different components of our system, event procudcer, APIs and user interfaces, main event processing engine and a knowledge component base which can be used to store background knowledge and doing reasoning on background knowledge base.

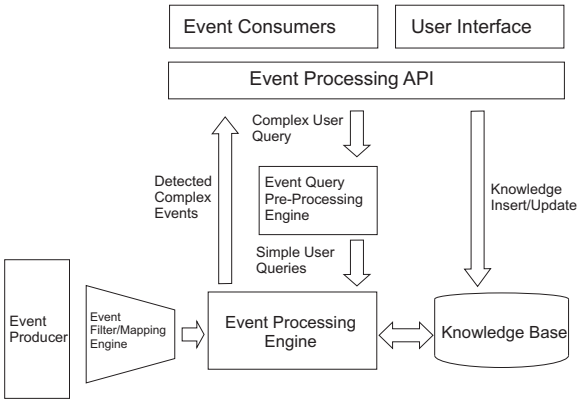

Fig. 2: Architecture of our SCEP Implementation

As main event processing engine, we use Prova[5] as a rule-based execution which can be used as event processing engine and as reaction rule language formalization. Prova uses reactive messaging[6], reaction groups and guards[7] for complex event processing. Multiple messages can be received 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 a Prova *sparql_select* built-in[8] to run SPARQL queries from Prova which can start a SPARQL query from inside Prova on an RDF file or a SPARQL endpoint. This Prova buit-in can use results which come from the SPARQL query and use them inside Prova.

During the processing of a SPARQL query inside Prova rules, the rule engine sends the embedded SPARQL query to the triple store and gets the results back. Afterwards it waits for incoming events to process. It processes the sequence of events using the provided results from the knowledge base. The SPARQL_select has the following syntax: sparql_select(QueryString, [SetOfOutputVariables], [SetOfInputVariables], ServiceEndpoint). The set of output variables are the results which come from the SPARQL query, the set of input variables provide the possibility to replace variables

[5] Prova, ISO Prolog syntax with extensions `http://prova.ws` , July 2011

[6] Prova Reactive Messaging `http://www.prova.ws/confluence/display/RM/Reactive+messaging` , July 2011

[7] Event Processing Using Reaction Groups `http://www.prova.ws/confluence/display/EP/Event+processing+using+reaction+groups`, July 2011

[8] Source codes for Semantic Web extensions in Prova 3 can be found in `https://github.com/prova/prova/tree/prova3-sw` , October 2011

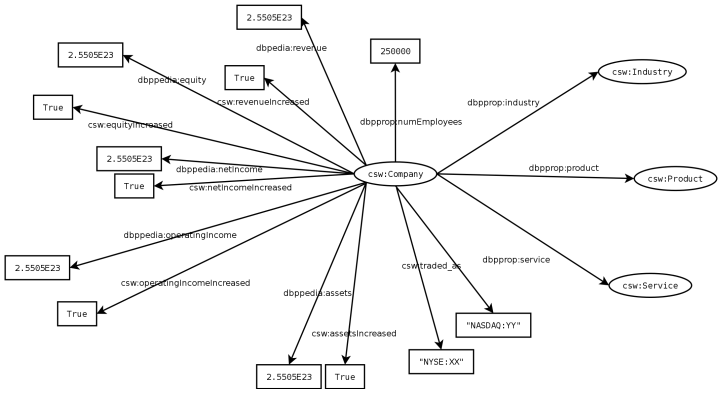

Fig. 3: A Simple Company Ontology

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. We analyzed the available data from DBpedia, about 500 Companies from the S&P 500 Index[9], we managed to extract about 302 different properties referring to companies. We included only the most important properties in our ontology. Our system is able to query DBpedia and other linked data endpoints directly. The stocks of company can be detected based on a set of properties for the concept "Company" and available knowledge on the background knowledge base. The knowledge base might be updated by the users but with a very lower frequency than the stock market events.

Listing 1.3: Prova Example for Semantic Event Detection.

```
:- eval(server()).

server() :-
sparqlrule(CompanySymbol, CompanyEmployees),
rcvMult(XID, Protocol, Sender, event,
{time->Time, symbol->Symbol, name->Name, lastprice->Lastprice, volume->Volume,
     high->High, low->Low})
[Symbol = CompanySymbol, CompanyEmployees > 5000],
sendMsg(XID, Protocol, Sender, testrule, {name->Name}).

sparqlrule(CompanySymbol, CompanyEmployees) :-
  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, [symbol(CompanySymbol), employees(CompanyEmployees)], [], '
     http://localhost:8890/sparql').
```

[9] http://www.standardandpoors.com/

The Listing 1.3 provides 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 5000 employees.

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 cases like ours, the frequency of such updates can be considered to not 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 event detection as proposed in [1]. However, in our experiments we assume that all of the event streams come to a central processing point. Our system shows clearly that the EQPP can achieve a better performance than the naïve storage-based (or pulling) approach. It also demonstrates 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; 1. Discharge rate of events,2. Number of rules in main memory, 3. Number of triples in the knowledge base (amount of knowledge), 4. Rate of knowledge updates,5. Expressive level of reasoning on background knowledge. Our demonstration system can be found on the Web at `http://slup.imp.fu-berlin.de/scepdemo/`.

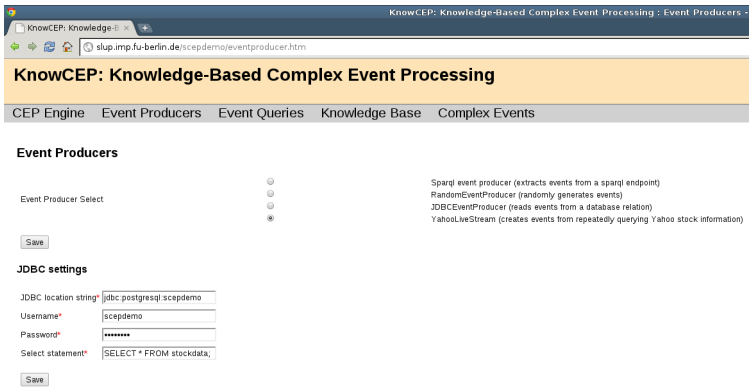

Fig. 4: Screenshot of KnowCEP: Knowledge-Based Complex Event Processing System

The user interface of our demonstration system consist of four parts, each of them have different functionalities; The first part is *CEP Engine Status*, a user can see the current status of the system, can start or stop the CEP engine. The second part is called *Event Query*, a user can give an event query in form of a Prova rule. The third part is *Knowledge Base* of the system, in this part a user can specify a SPARQL endpoint as an external knowledge base, can send SPARQL queries to end point and get the results back to the system, the user can also post and add RDF data to the external RDF store.

The next component is the configuration of the event sources, as shown in Figure 4, users can select between different event producers, live stock market data stream, stored stream from a database, or randomly generated stock market events for demonstration purposes. After the configuration of the CEP system, it can be started from the main system menu.

5 Conclusion and Outlook

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

References

1. Mert Akdere, Uğur Çetintemel, and Nesime Tatbul. Plan-based complex event detection across distributed sources. *Proc. VLDB Endow.*, 1:66–77, August 2008.
2. Liming Chen and Chris D. Nugent. Ontology-based activity recognition in intelligent pervasive environments. *IJWIS*, 5(4):410–430, 2009.
3. Claudia d'Amato, Nicola Fanizzi, Bettina Fazzinga, Georg Gottlob, and Thomas Lukasiewicz. Combining semantic web search with the power of inductive reasoning. In Amol Deshpande and Anthony Hunter, editors, *SUM*, volume 6379 of *Lecture Notes in Computer Science*, pages 137–150. Springer, 2010.
4. 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.
5. A. Paschke. Eca-lp / eca-ruleml: A homogeneous event-condition-action logic programming language. In *RuleML-2006*, Athens, Georgia, USA, 2006.
6. 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.
7. Adrian Paschke and Martin Bichler. Knowledge representation concepts for automated sla management. *Decis. Support Syst.*, 46(1):187–205, 2008.
8. Adrian Paschke, Alexander Kozlenkov, and Harold Boley. A homogeneous reaction rule language for complex event processing. *CoRR*, abs/1008.0823, 2010.
9. 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.
10. 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.
11. 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.