Introducing Knowledge-Enrichment Techniques for Complex Event Processing

Sebastian Binnewies and Bela Stantic

School of Information and Communication Technology
Griffith University, Brisbane, Australia
`sebastian.binnewies@griffithuni.edu.au`
`b.stantic@griffith.edu.au`

Abstract. Complex event processing received an increasing interest during the last years with the adoption of event-driven architectures in various application domains. Despite a number of solutions that can process events in near real-time, their effectiveness for decision support relies heavily upon human domain knowledge. This poses a problem in areas that require vast amounts of specialized knowledge and background information, such as medical environments. We propose four techniques to enrich complex event processing with domain knowledge from ontologies to overcome this limitation. These techniques focus on preserving the strengths of state-of-the-art systems and enhancing them with existing ontologies to increase accuracy and effectiveness. The viability of our approach is demonstrated in a multifaceted experiment.

Key words: Complex Event Processing, Ontologies, Rule Management, Semantic Complex Event Processing

1 Introduction

Everyday life is surrounded by events – which can be any type of occurrence in an environment – and human actions are influenced by these events. Many organizations have adopted this perspective for their IT-supported processes and have incorporated an event-driven architecture [6]. A core component of these architectures is *complex event processing*. Complex event processing systems receive events in the form of data, process them to generate meaningful complex events, and trigger responses upon creation of complex events. Events can take a multitude of forms, e.g., clickstream data or sensor data from radio frequency identification (RFID) networks, and generally occur in high-volume streams.

Within the last ten years, complex event processing has replaced database-centric systems with real-time decision support in areas such as algorithmic trading, foreign exchange pricing, network security and energy smart metering. However, despite a wide range of potential application areas, complex event processing is not a silver bullet solution in itself. Processing is performed on the basis of rules, which have to reflect the requirements of each specific application

domain. Therefore, rule specification and, consequently, the effectiveness of complex event processing for decision support are highly dependent on the amount and quality of knowledge about that domain. This becomes an issue particularly when complex event processing is to be used in environments that are rich in domain knowledge, for instance, in hospital settings. The persons responsible for rule specification may possess enough knowledge to be considered medical domain experts, but a vast amount of specialized knowledge and background information is disregarded.

We propose to enrich complex event processing with *domain knowledge from ontologies* to address this issue. Ontologies model knowledge about a domain and enable automated reasoning over this knowledge. We present four techniques to increase the accuracy of rule specification and evaluation by accessing and reasoning over ontologies. The four techniques allow a relaxation of rule conditions and integration of background information into complex event processing. In contrast to existing work, our lightweight approach focuses on utilizing existing ontologies and preserving the strengths of recent advances in complex event processing. The viability of our approach is supported by the results of a multi-faceted experiment.

2 Background

Our research incorporates the two disciplines complex event processing and knowledge representation in ontologies. The following two sections briefly introduce the recent advances in each area.

2.1 Complex Event Processing

In short, the term complex event processing stands for the processing of simple events to generate complex events. The notion of an *event* refers to any occurrence in an environment that can be expressed in the form of data, for example, the time, location and object identifier of an RFID reading. Events are sent from an event source to the complex event processing engine for processing. The engine filters incoming events, creates complex events upon detection of specific event patterns, and routes these complex events to an event sink or back to the engine for further processing. An essential component of complex event processing are event pattern rules. They specify a certain combination of events, their attributes and values – an event pattern – and an action to be executed if the event pattern is satisfied.

Despite several effective systems that address the inherent challenges of complex event processing, a single best solution does not exist. These challenges include, for instance, high volumes of input data, unordered event streams, uncertainty of event occurrences, temporal relationships, and processing performance, which are addressed by current commercial and academic systems (e.g., SASE [18], TESLA [4], ETALIS [2], TMS-RFID [11]). Yet, different systems have different strengths, suggesting that there is no single, silver bullet solution in

complex event processing. In fact, a one-fits-all solution may not be feasible at all, due to the wide range of situations in which complex event processing is applied. Therefore, we emphasize that a system with its specific strengths should be chosen according to the area of application and its requirements.

2.2 Ontologies

Ontologies are used to model knowledge about a domain and to reason about this knowledge. Generally, an ontology represents the concepts of a domain as classes, individuals (instances of classes) and properties (relationships between classes and individuals). In addition, if an ontology captures the *formal semantics* of concepts, i.e., if it formally defines the meaning of these concepts through logical axioms, the semantics can be accessed and processed by an automated reasoner.

A growing number of ontologies are being developed by various communities and many of them are freely accessible on the World Wide Web. The Semantic Web community has produced several standards related to ontologies, such as RDF, OWL, RIF and SPARQL.¹ The life sciences have also adopted the technology of ontology modeling and its benefits, especially in the fields of biology and medicine, as evident in the prominent ontologies SNOMED CT², the Gene Ontology³ and further ontologies from the OBO Foundry⁴. Our focus lies on utilizing existing ontologies to enrich the functionality of complex event processing. We are specifically interested in biomedical environments that are rich in domain knowledge, where ontologies can provide highly specialized information and background knowledge to increase processing accuracy.

3 Related Work

While research has been conducted in the individual fields of ontologies and event processing for years, finding possible synergies has only recently been undertaken. Combining ontologies and complex event processing is closely related to the broader research area of *semantic complex event processing*. The latter is a relatively new term and refers to complex event processing that is enhanced with reasoning capabilities. Reasoning over domain knowledge captured in an ontology is a sub-area of semantic complex event processing and still an emerging research field, where few publications exist to date.

A first complete solution was recently presented [1], introducing EP-SPARQL, an extension to SPARQL. EP-SPARQL combines event processing and SPARQL to retrieve the semantics of events captured in RDF serialized knowledge bases. While this may be a solution for event input in the form of RDF triples, data in other formats would have to be converted into RDF triples or graphs. Further investigation is required how this conversion affects event processing performance.

¹ <http://www.w3.org/2001/sw/wiki>

² <http://www.ihtsdo.org/snomed-ct/>

³ <http://www.geneontology.org/>

⁴ <http://www.obofoundry.org/>

In addition, EP-SPARQL queries would require the serialization of OWL concepts into less expressive RDF.

Only few other solutions have been proposed, which are limited in one way or another and differ greatly from our approach. Three different approaches ([5], [12], [14]) each implement a working combination of ontologies and complex event processing. However, each of them is restricted to one specific application domain. The remaining publications ([15], [16], [17]) primarily discuss the idea of a solution approach and lack the presentation of a working implementation. Two features are common to all these approaches. Firstly, they create a new ontology for the purpose of event processing. Secondly, they propose a tight integration of ontology and complex event processing by using a combined rule language and by executing event processing completely through reasoning over ontology concepts. Our approach, on the other hand, aims to utilize existing ontologies and does not restrict itself to a specific domain. It is a lightweight approach that focuses on preserving the strengths of existing complex event processing systems and ontologies, which stands in contrast to a tight integration.

Semantic complex event processing is surrounded by several related disciplines. *Event stream processing* shares similarities with complex event processing as it also evaluates a stream of events to detect specific events. However, it focuses on maximizing throughput and minimizing latency [13], rather than on the expressiveness of event pattern rules. On the other hand, *business rule management systems* strongly focus on rules. These rules can be quite similar to event pattern rules in complex event processing. Yet, business rules are checked when explicitly called from an application, not when an event arrives [6]. Furthermore, business rule processing is done through inferencing, in contrast to the three-step event processing procedure outlined above. *Stream reasoning* is also concerned with inferencing, albeit through reasoning over dynamically changing sets of knowledge. Existing work in this field has focused mainly on data in RDF format, e.g., [3], and stream reasoning has yet to address in detail such challenges mentioned in Section 2.1 as complex event processing has.

4 Enrichment Techniques and Their Benefits

As the previous section demonstrated, the concept of integrating complex event processing and ontologies is still an emerging research area, and only a few research projects have been conducted. However, none of these projects has clearly identified the general opportunities and benefits for complex event processing. We thoroughly analyzed the common characteristics of complex event processing, on the one hand, and of ontologies and Semantic Web technologies, on the other hand, to derive the potential of an integrated approach. The following sections introduce four techniques to enrich complex event processing with domain knowledge from ontologies, which we define as *ontology-enhanced complex event processing* (OECEP). Section 4.5 then emphasizes how these techniques can benefit complex event processing in general, since they can be used to increase its functionality irrespective of a specific implementation. To illustrate the

four techniques and their benefits, we introduce a possible real-world scenario first:

At an airport, a large number of containers with food products arrive daily. Information about the containers and their content is often retrieved through scanning a barcode or reading an RFID tag at the airport's loading dock. The containers are then forwarded to the next destination based upon an evaluation of this information. Some of the containers may come from a different country and contain food products that have to be inspected by customs and quarantine. In this case, such critical containers have to be separated from non-critical containers and routed to quarantine. A complex event processing engine can be employed to evaluate container information on the basis of event pattern rules and trigger a response in the form of a routing decision. Nevertheless, due to the variety of food products and their ingredients, an impracticably large number of rules would be needed.

Assuming that information about the food domain is represented in an ontology, a complex event processing engine could use this ontology to significantly reduce the number of rules needed. Furthermore, rules may be stated in such a way that they use background knowledge encoded in the ontology. This can be achieved through the following four techniques.

4.1 Rule Rewriting

In 'standard' complex event processing, each condition of a rule has to be explicitly specified to create a complex event, e.g., all event types that will satisfy this rule. OECEP allows a relaxation of conditions while still creating the same complex event. This is realized through a rule rewriting process that includes reasoning over the ontology. The rule rewriting process is initiated with the engine passing a relaxed condition to the ontology reasoner. The reasoner then draws all inferences relating to this condition from the ontology. These inferences, called *inferred conditions*, are passed back to the engine. The engine then substitutes the relaxed condition with the inferred conditions.

In the airport example, one condition to deem a container as critical could be that the content is some kind of seafood. If a container with seafood in it arrives, the information gathered from the barcode or RFID tag (and possibly from a central database) would include the type of seafood, e.g., 'Tuna'. Normally, each type of seafood, i.e., 'Tuna', 'Swordfish', 'Flounder' etc., would have to be formulated in one or more rules so that the engine can correctly detect all seafood containers. With OECEP, it is only necessary to declare one rule, which has 'Seafood' as a condition. This condition is passed to the reasoner, which retrieves all instances of 'Seafood' and its subclasses from a food ontology. These instances are passed back to the engine. The engine then rewrites the rule and evaluates container information against the inferred conditions. Figures 1 and 2 depict the reasoning step and the process for this rule rewriting example, respectively.

Fig. 1. Graph-based representation of ontology concepts including rule rewriting and input rewriting paths (based on the Food Ontology)

4.2 Input Rewriting

OECEP also makes it possible to rewrite input events. In input rewriting, information from an incoming event is passed to the ontology reasoner as a pre-processing step. The reasoner computes all relevant inferences and passes them back to the engine. The engine rewrites the input event with this inferred knowledge, creating *inferred event(s)*. These inferred events are then processed by the engine as usual.

Referring back to the airport example, the information that a recently arrived container holds a specific type of seafood, e.g., ‘Tuna’, is sent to the reasoner. The reasoner infers from the ontology that ‘Tuna’ is an instance of ‘NonBlandFish’ and of the superclasses ‘Fish’ and ‘Seafood’, and returns this information to the engine. The engine then substitutes the value ‘Tuna’ with the inferred information for the contents attribute of the event. This is done either by creating a new instance of this event for each inferred value, or by inserting multiple values in the contents attribute of this event if this is supported by the engine. The input rewriting path and process are displayed in Figures 1 and 2, respectively.

4.3 Output Rewriting

Output rewriting is a similar process to input rewriting, with the difference that a complex event is rewritten. When a complex event has just been created, its information is passed to the reasoner to infer further knowledge. This information is returned to the engine to rewrite the complex event into one or more *inferred complex event(s)*. Finally, the engine forwards these inferred complex events to the event sink or takes them as input for further processing.

Fig. 2. Example processes of rule rewriting and input rewriting

In the airport example, a complex event is created when a container with fish is detected. This output could be rewritten into a complex event with the information that a container is detected whose contents need to be kept frozen.

4.4 Incorporating Domain Knowledge

In addition to the rewriting capabilities, OECEP also allows to include specific background knowledge in complex event processing. The extent to which this is possible depends on the knowledge that is actually captured in the ontology and on the purpose of complex event processing in this domain. Incorporating such domain knowledge into complex event processing can happen in many forms. For instance, a rule may specify as a condition that an event has to be of type A, and A has to participate in at least one relationship α (to an event of type B). Although the engine can check the type of incoming events, the evaluation of the relationship α can only be executed by an ontology reasoner. As another example, a rule may state that the maximum path length between A and B can only be 5 (if the ontology is modeled as a graph).

A customs and quarantine rule could include a condition that checks if a food product has seeds. It is assumed that the ontology models a relationship 'hasSeed' between certain foods and their seeds – e.g., stone, single seed, multiple seeds. When the engine evaluates such a rule, it passes event information to the reasoner. The reasoner checks if there exists at least one relationship 'hasSeed' for this event and passes a boolean value back to the engine.

Conceptually, rule rewriting, input rewriting and output rewriting are a form of incorporating domain knowledge. Here, we define the concept of incorporating domain knowledge to refer to all enhancements, which do not belong to a rewriting technique. Obviously, it is possible to use any combination of rule

rewriting, input rewriting and output rewriting, as well as incorporating domain knowledge, for rule specification and evaluation in complex event processing.

4.5 Benefits

From the descriptions above, it becomes evident that utilizing domain knowledge from ontologies enables *more intelligent* complex event processing. Firstly, rule rewriting, input rewriting and output rewriting can greatly reduce the number of rules that need to be formulated. This decreases the workload of human domain experts. Secondly, the engine can refer to background information. This allows specification of more expressive rules and increases the accuracy of processing. Thirdly, these general techniques have specifically been designed to be independent. They can be used independent of a specific ontology, complex event processing system or application domain. Nevertheless, a few challenges appear when looking more closely at the inherent features of complex event processing and ontologies, such as a formalization of complex event processing language to match the formal semantics in ontologies, a suitable interaction mechanism, expressiveness and conflicting characteristics of event and ontology languages, and processing performance.

4.6 Relevance

We see a high relevance of our approach specifically to the area of e-Health. In health care environments, we find an array of devices and sensors that monitor patients during an operation, recovery in hospital or at their home. Complex event processing can process this stream of data, detect complex events and trigger a response. However, most responses require a thorough understanding of medical background information related to an event, e.g., the type of illness [10]. Currently, complex event processing cannot effectively incorporate this kind of information, because it is stored in a variety of formats, such as electronic patient records, clinical databases and free text reports [7]. In order to access and interlink data from these diverse sources, a common definition of terms is needed. SNOMED CT has recently been chosen by the Australian National e-Health Transition Authority for standardizing clinical terminology [8] and already supports medical software in many other countries [9]. With the help of SNOMED CT, a complex event processing engine could access and utilize medical background information to generate more accurate responses, which would support better decision-making in e-Health environments.

5 Experimental Evaluation

To demonstrate the viability of our enrichment techniques, we conducted an extensive experiment. The experiment design variables are presented in the following section. Based on the experiment design, we formulated three hypotheses, which are given in Section 5.2. Section 5.3 discusses the experiment results.

5.1 Experiment Design

As shown in Table 1, we assessed two ontologies, two rewriting techniques and two interaction mechanisms in two possible real-world scenarios, which are explained in detail below. All tests were executed on a PC with Microsoft Windows XP Professional Version 2002 SP 3, Intel Core 2 CPU 1.80GHz and 2GB of RAM. Programming code was written in Java using Java 1.6.0.

Table 1. Experiment Design Variables

	Scenario 1	Scenario 2
Ontologies	Food Ontology (OWL DL <i>SHOIN</i>)	Vaccine Ontology (OWL DL <i>SHOIN</i>)
Techniques	Rule Rewriting / Input Rewriting	Rule Rewriting / Input Rewriting
Interaction Mechanism	OWL API 3.2.2 with HermiT 1.3.3 / SPARQL through Jena API 2.6.4	OWL API 3.2.2 with HermiT 1.3.3
Avg. Num. of Input Events	3050	83

Scenario 1. Scenario 1 resumes the airport example introduced in Section 4 and extracts domain knowledge from the Food Ontology⁵. To recapitulate, cases with food arrive by airplane, are checked for critical contents and are routed to quarantine if critical. An additional requirement in this scenario is that special food products should be processed quickly by quarantine. This is specified in the following two rules.

Rule 1:

If a case arrives at the loading dock that contains a critical food item,
then notify the pusher to route this case to quarantine.

Rule 2:

If a case waits to be assessed by quarantine for a certain period of time
and the case contains some seafood,
then produce a notification.

Each case is assumed to be equipped with an RFID tag and both the loading dock and the quarantine hall with RFID readers. It is further assumed that the data from a case reading can be mapped to information about the type of food that is in the case. Each reading represents a simple event and consists of data about the read time, the read place and the contents of the case that is read.

⁵ <http://www.w3.org/TR/2004/REC-owl-guide-20040210/food.rdf>

The Food Ontology models knowledge about food in 138 named classes, 206 individuals, 17 properties and 356 axioms. We utilize this ontology for two rewriting techniques – rule rewriting and input rewriting. In rule rewriting, all individuals of the classes which represent critical food items (for Rule 1) or fish items (for Rule 2) are computed and substituted for the original conditions. For instance, if Rule 2 specifies the class ‘Seafood’, it is rewritten to ‘Tuna’, ‘Oysters’, ‘Crab’, Thus, if a case content matches any one of these individuals, the rule is satisfied. In input rewriting, all superclasses of the individual representing the case content are computed. Each input event is then rewritten by substituting the original case content with its superclasses. For example, if a case contains ‘Tuna’, the event would be rewritten to have as contents ‘NonBlandFish’, ‘Fish’, ‘Seafood’,

In the experiment, rule rewriting and input rewriting are evaluated separately. Evaluation is performed through an implementation of the Rete algorithm. For both rewriting techniques, we tested two different mechanisms to access the ontology, SPARQL and the OWL API. For each of these variations, we performed ten runs having an average number of 3050 events with randomly created timestamps. The results are given in Figure 3 and analyzed below.

Scenario 2. Scenario 2 describes a situation in which vaccines are transported between supply chain partners. Vaccines that do not contain a preservative have to be kept at a certain temperature. If such vaccines are transported and a refrigeration device fails, an alarm should be triggered. This is expressed in the following rule.

If the temperature rises above 6 °C
and the vaccine does not contain a preservative,
then produce a notification.

Similar to Scenario 1, event data would arrive from RFID sensors, with tags attached to the vaccine containers and read at each station during transport, supplemented with data from temperature sensors.

We created this scenario to utilize the Vaccine Ontology⁶, a much more comprehensive ontology than the Food Ontology, for rule rewriting and input rewriting. The Vaccine Ontology is an effort to standardize vaccine data such as vaccine components and vaccine categorization. At present, the Vaccine Ontology contains 3530 named classes, 120 individuals, 80 properties and 21,548 axioms. In rule rewriting, all subclasses of the class ‘vaccine preservative’ are computed and substituted for that class in the rule. In input rewriting, each input event representing a vaccine and its specific contents is rewritten with the superclasses of these contents. For example, if a vaccine contains the preservative ‘phenoxyethanol’, the input event would be rewritten with the information ‘vaccine preservative’, ‘vaccine additive’,

Again, we evaluated rule rewriting and input rewriting separately. The OWL API was used to access the ontology and compute the inferences. We executed

⁶ <http://www.violinet.org/vaccineontology>

ten runs with an average number of 83 events. The results are presented in Figure 4 and discussed below.

5.2 Hypotheses

Hypothesis 1. Rule rewriting is faster than input rewriting.

Rule rewriting is generally performed only once, i.e., when the rule is entered into the rulebase of the engine, and rule rewriting time is independent of the number of input events. Input rewriting, on the other hand, is performed for each input event that is represented in the ontology. Therefore, we expect its total time for all events to be larger than total rule rewriting time.

Hypothesis 2. The OWL API offers greater rewriting functionality but inferior scalability compared to SPARQL (through the Jena API) as a mechanism to interact with an ontology.

The OWL API offers direct access to OWL ontologies, whereas SPARQL requires the serialization of OWL into less expressive RDF. We thus expect a larger range of rewriting functionality from the OWL API. We also expect that greater functionality comes at the expense of scalability and anticipate that processing time with the OWL API is higher than with SPARQL.

Hypothesis 3. The complexity of an ontology is the key influence on rewriting time.

The complexity of an ontology can be measured along several metrics [19]. We chose the number of concepts as the crucial metric and expect a larger number of concepts to have a positive effect on rewriting time, i.e., the higher the complexity, the longer the processing time.

5.3 Results and Analysis

Figure 3 shows the average processing time per event in microseconds for each variation of Scenario 1, calculated and averaged over 10 runs each. When the OWL API was used, the processing time consisted of loading and pre-computing the ontology and reasoning over the ontology. As can be seen from the chart, the results are mixed across the variations, ranging from $460\ \mu s$ (Rule Rewriting with SPARQL) to $7332\ \mu s$ (Rule Rewriting with the OWL API). We can draw three conclusions from the results. Firstly, rule rewriting is not necessarily faster than input rewriting. While rule rewriting with SPARQL is approximately one fifth of the time for input rewriting with SPARQL, rule rewriting with the OWL API scales much worse than input rewriting with the OWL API. Secondly, SPARQL does not always perform better than the OWL API. Although SPARQL has a significantly lower processing time than the OWL API in rule rewriting, the OWL API outperforms SPARQL in input rewriting. Thirdly, when using the OWL API, it can be either the loading time or the reasoning time that is the major influence on total processing time. While loading the ontology makes

up only 15% of the total time in rule rewriting, it accounts for 90% in input rewriting.

Fig. 3. Comparison of rewriting techniques and interaction mechanisms for Scenario 1 (Food Ontology)

Figure 4 depicts the total processing time in milliseconds for rule rewriting and input rewriting with the OWL API in Scenarios 1 and 2, calculated and averaged over 10 runs each. We can again derive three conclusions from the chart. Firstly, the complexity of an ontology influences rewriting time. Utilizing the complex Vaccine Ontology for rewriting takes considerably longer than using the less comprehensive Food Ontology. Secondly, as already seen in Figure 3, rule rewriting is not necessarily faster than input rewriting. When using the Food Ontology, input rewriting performs better than rule rewriting. When using the Vaccine Ontology, both rewriting techniques perform similarly. Thirdly, SPARQL offers less rewriting functionalities than the OWL API. SPARQL (through the Jena API) could not be used for rule rewriting or input rewriting in Scenario 2 since it cannot handle relative URIs requiring specification of multiple base URIs. Therefore, only the OWL API was used in Scenario 2.

Hypothesis 1 could not be proven. Figure 3 showed that the superiority of rule rewriting over input rewriting (and vice versa) depends on the interaction mechanism. Additionally, input rewriting scaled significantly better than rule rewriting in Scenario 1, but only marginally better in Scenario 2. This can be explained by the type of reasoning that has to be executed. In Scenario 1, the reasoner has to compute relationships between classes and individuals, in Scenario 2 only between classes.

Hypothesis 2 was only partly supported by the results. Scenario 2 could not be evaluated with SPARQL because it lacks a functionality that the OWL

Fig. 4. Comparison of rewriting techniques in Scenarios 1 and 2 (Food Ontology and Vaccine Ontology)

API offers. In Scenario 1, the OWL API and SPARQL outperform each other depending on the rewriting technique. Thus, our hypothesis could be proven in regards to the superior functionality of the OWL API, but neither proven nor disproven in regards to scalability.

Hypothesis 3 was supported by the results. For both rule rewriting and input rewriting, reasoning over the more complex Vaccine Ontology took noticeably longer than reasoning over the Food Ontology, even though the reasoning task was more complex in the latter case.

In summary, although the scope of our experiment is limited, we obtained very interesting results. Both rule rewriting and input rewriting were realized, which suggests that the remaining two techniques – output rewriting and incorporating domain knowledge – are also feasible. Yet, even the fastest variation yields no more than approximately 2200 rewrites per second. This certainly leaves room for optimization. Moreover, the performance is highly dependent on the complexity of the ontology and on the type of knowledge to be extracted. Further investigation into the influence of these factors is required to determine the best strategy for integrating domain knowledge into complex event processing.

6 Conclusion

Our study identified a shortcoming in the current procedure of rule specification and evaluation in complex event processing and analyzed how domain knowledge from ontologies could be utilized to overcome this limitation. We proposed four techniques that enable the use of domain knowledge to increase the accuracy of complex event processing and to reduce the workload of rule specification.

Our approach emphasized the techniques' independence of a specific system or ontology, making them suitable for a wide range of application domains. Experimental results showed the feasibility of these techniques and indicated the direction for further research.

Future work will focus on the challenges mentioned throughout this paper and integrate our techniques into state-of-the-art complex event processing systems. We further aim to access modules of SNOMED CT with the help of the classifier Snorocket⁷ to support complex event processing in e-Health environments.

Acknowledgments. The authors would like to thank Kewen Wang for his contribution to the initial idea of this project.

References

1. Anicic, D., Fodor, P., Rudolph, S., Stojanovic, N.: EP-SPARQL: A unified language for event processing and stream reasoning. In: Proceedings of the 20th International Conference on World Wide Web, WWW 2011. pp. 635–644. ACM (2011)
2. Anicic, D., Fodor, P., Rudolph, S., Stühmer, R., Stojanovic, N., Studer, R.: A rule-based language for complex event processing and reasoning. In: Hitzler, P., Lukasiewicz, T. (eds.) Web Reasoning and Rule Systems, Lecture Notes in Computer Science, vol. 6333, pp. 42–57. Springer Berlin / Heidelberg (2010)
3. Barbieri, D., Braga, D., Ceri, S., Della Valle, E., Grossniklaus, M.: Incremental reasoning on streams and rich background knowledge. In: Aroyo, L., Antoniou, G., Hyvnen, E., ten Teije, A., Stuckenschmidt, H., Cabral, L., Tudorache, T. (eds.) The Semantic Web: Research and Applications, Lecture Notes in Computer Science, vol. 6088, pp. 1–15. Springer Berlin / Heidelberg (2010)
4. Cugola, G., Margara, A.: TESLA: A formally defined event specification language. In: Proceedings of the Fourth ACM International Conference on Distributed Event-Based Systems, DEBS 2010. pp. 50–61. ACM (2010)
5. Dunkel, J.: On complex event processing for sensor networks. In: 2009 International Symposium on Autonomous Decentralized Systems. pp. 1–6. IEEE (2009)
6. Etzion, O., Niblett, P.: Event Processing in Action. Manning, Greenwich, CT (2011)
7. Hansen, D., Karunanithi, M., Lawley, M., Maeder, A., McBride, S., Morgan, G., Pang, C., Salvado, O., Sarela, A.: Improving the use, analysis and integration of patient health data. In: Ishikawa, Y., He, J., Xu, G., Shi, Y., Huang, G., Pang, C., Zhang, Q., Wang, G. (eds.) Advanced Web and Network Technologies, and Applications, Lecture Notes in Computer Science, vol. 4977, pp. 74–84. Springer Berlin / Heidelberg (2008)
8. Hansen, D., Kemp, M., Mills, S., Mercer, M., Frosdick, P., Lawley, M.: Developing a national emergency department data reference set based on SNOMED CT. The Medical Journal of Australia 194(4), S8–S10 (2011)
9. International Health Terminology Standards Development Organisation: Results of Survey to Gather the Use, Benefits and Tools of SNOMED CT. <http://www.ihtsdo.org/index.php?id=826> (2011), Date: 20110119, Version: 1.0

⁷ <http://research.ict.csiro.au/software/snorocket>

10. Lawley, M.: Exploiting fast classification of SNOMED CT for query and integration of health data. In: *Proceedings of the 3rd International Conference on Knowledge Representation in Medicine (KR-MED 2008): Representing and Sharing Knowledge Using SNOMED*. pp. 8–14 (2008)
11. Li, X., Liu, J., Sheng, Q., Zeadally, S., Zhong, W.: TMS-RFID: Temporal management of large-scale RFID applications. *Information Systems Frontiers* pp. 1–20 (2009)
12. Liu, D., Pedrinaci, C., Domingue, J.: A framework for feeding Linked Data to complex event processing engines. In: *The 1st International Workshop on Consuming Linked Data (COLD 2010) at The 9th International Semantic Web Conference (ISWC 2010)* (2010)
13. Paschke, A., Kozlenkov, A.: Rule-based event processing and reaction rules. In: Governatori, G., Hall, J., Paschke, A. (eds.) *Rule Interchange and Applications*, *Lecture Notes in Computer Science*, vol. 5858, pp. 53–66. Springer Berlin / Heidelberg (2009)
14. Paulheim, H.: Efficient semantic event processing: Lessons learned in user interface integration. In: Aroyo, L., Antoniou, G., Hyvönen, E., ten Teije, A., Stuckenschmidt, H., Cabral, L., Tudorache, T. (eds.) *The Semantic Web: Research and Applications*, *Lecture Notes in Computer Science*, vol. 6089, pp. 60–74. Springer Berlin / Heidelberg (2010)
15. Schmidt, K., Anicic, D., Stühmer, R.: Event-driven reactivity: A survey and requirements analysis. In: *3rd International Workshop on Semantic Business Process Management*. pp. 72–86 (2008)
16. Teymourian, K.: Enabling knowledge-based complex event processing. In: *Proceedings of the 2010 EDBT Workshops*. pp. 1–7. ACM (2010)
17. Wienhofen, L.W.M., Toussaint, P.J.: Enriching events to support hospital care. In: *Proceedings of the 7th Middleware Doctoral Symposium*. pp. 26–30. MDS '10, ACM (2010)
18. Zhang, H., Diao, Y., Immerman, N.: Recognizing patterns in streams with imprecise timestamps. *Proceedings of the VLDB Endowment* 3(1), 244–255 (2010)
19. Zhang, H., Li, Y., Tan, H.: Measuring design complexity of semantic web ontologies. *Journal of Systems and Software* 83(5), 803–814 (2010)