Towards Scalable Federated Context-Aware Stream Reasoning

Alexander Dejonghe(✉)

Department of Information Technology (INTEC), Ghent University - iMinds, Gaston Crommenlaan 8 Bus 201, 9050 Ghent, Belgium
Alexander.Dejonghe@intec.ugent.be

Abstract. With the rising interest in internet connected devices and sensor networks, better known as the Internet of Things, data streams are becoming ubiquitous. Integration and processing of these data streams is challenging. Semantic Web technologies are able to deal with the variety of data but are not able to deal with the velocity of the data. An emerging research domain, called stream reasoning, tries to bridge the gap between traditional stream processing and semantic reasoning. Research in the past years has resulted in several prototyped RDF Stream Processors, each of them with its own features and application domain. They all cover querying over RDF streams but lack support for complex reasoning. This paper presents how adaptive stream processing and context-awareness can be used to enhance semantic reasoning over streaming data. The result is a federated context-aware architecture that allows to leverage reasoning capabilities on data streams produced by distributed sensor devices. The proposed solution is stated by use cases in pervasive health care and smart cities.

Keywords: Semantic stream processing · Stream reasoning · Context awareness · Internet of Things (IoT)

1 Introduction

In recent years we saw an increasing interest in internet connected devices and sensors, also called the Internet of Things (IoT) [18]. This led to a plethora of new applications in different domains like smart cities, traffic monitoring, pervasive healthcare, smart energy grids, smart buildings and environmental sensing. The data generated by these IoT devices is a heterogeneous, voluminous, and a possibly noisy or incomplete set of time-varying data elements called data streams. The combination of these characteristics makes it a challenging task to integrate, interpret and process data streams on the fly. Moreover, to create added value out of these data streams, the data must be combined with domain knowledge.

For example, in smart nursing homes and hospital, rooms are equipped with Wireless Sensor Networks (WSNs) monitoring the environment, and patients' health parameters are monitored by Body Area Networks (BANs). Lots of data

H. Sack et al. (Eds.): ESWC 2016, LNCS 9678, pp. 803–812, 2016.
DOI: 10.1007/978-3-319-34129-3_49

events are generated by these sensor networks. To take advantage of the collected streaming data, integration with domain knowledge containing diseases, symptoms and the patient's Electronic Health Record (EHR) is important. Reasoning on this information can infer new knowledge of the patients' current condition. Well integrated and aggregated information helps to monitor the patients more closely and allows the nursing resources to operate more efficiently.

Semantic Web technologies like the data model RDF[1], the ontology language OWL[2], and the RDF query language SPARQL[3] allow to represent, integrate, query and reason on heterogeneous data. However, these technologies were developed for static or slow changing data sources. On the other end of the spectrum Data Stream Management Systems (DSMS) and Complex Event Processing (CEP) systems allow to query homogeneous streaming data structured according to a fixed data model. They are not able to deal with heterogeneous data sources and lack support for the integration of domain knowledge. To bridge this gap, stream reasoning has emerged as a challenging research area that focuses on the adoption of Semantic Web technologies for streaming data. Della Valle et al. [7] describes stream reasoning as a high-impact research area with a multidisciplinary approach that can provide the abstractions, foundations, methods, and tools required to integrate data streams, the Semantic Web, and reasoning systems.

As a result of stream reasoning research conducted in the past years, different prototypes of RDF Stream Processing (RSP) engines have been presented. These RSP engines can filter and query RDF data streams, but are not able to deal with complex reasoning tasks. Reasoning over large complex ontologies is computationally intensive and slow compared to the velocity of the data streams. Traditional reasoners, like FaCT++ [19], HermiT [15] and Pellet [16], that have the capabilities of performing such complex OWL 2 DL reasoning tasks are designed to process static or slow evolving data, and are not able to manage frequently changing data streams.

Despite all initiatives taken, stream reasoning is not yet mature and there is a need for algorithms, protocols and approaches that support a scalable, efficient and complex stream reasoning [11].

The aim of this research is to present a federated context-aware system that integrates adaptive stream processing and complex reasoning. The federated system should exploit context-awareness to bring together low-level stream reasoning and high-level complex reasoning. The low-level stream reasoning should be performed close to the data stream sources while complex reasoning should be performed deeper in the network.

This approach might especially be useful in large scale sensor applications where an effective processing of the event streams is of high importance. By using context information about the environment, irrelevant sensor data can

[1] http://www.w3.org/TR/rdf-syntax-grammar/.
[2] http://www.w3.org/TR/owl-overview/.
[3] http://www.w3.org/TR/sparql11-overview/.

be filtered and only applicable domain knowledge can be take into account for reasoning.

2 State of the Art

Several RSP solutions such as C-SPARQL [4], CQELS [10], EP-SPARQL [2] and SPARQLStream [6], which all mainly focus on stream processing, have been developed in the past years. They extend SPARQL by using proven techniques from DSMSs and CEP systems, namely sliding windows and continuous queries [3]. A continuous query is registered once and produces results continuously over time as the streaming data in the considered window changes.

The prototyped RSP engines enable processing of a continuous flow of data and can provide real-time answers to registered queries. However, each of them has different semantics and targets different scenarios. Steps towards a unifying semantics query model are being taken by the W3C RDF Stream Processing Community Group[4].

Other solutions like Sparkwave [9] and INSTANS [14] make use of extensions of the RETE algorithm [8] for pattern matching. With this approach queries are translated into a RETE network through which the data flows. The resulting network consists of a set of nodes which can memorize partial pattern matches in the streaming data.

All RSP engines, with the exception of INSTANS, support integration of domain knowledge in the querying process, but reasoning capabilities are limited (Table 1). None of the proposed systems is able to perform complex OWL 2 DL reasoning on streaming data. C-SPARQL is the only engine supporting full RDFS reasoning using the Jena Rule Engine. EP-SPARQL, that is build around the Prolog engine ETALIS, also supports RDFS reasoning but the domain ontologies have to be converted into Prolog rules and facts in advance. Sparkwave supports an RDFS subset but as for EP-SPARQL preprocessing is necessary. Both, the domain knowledge and the query conditions have to be compiled into nodes of RETE network in advance. SPARQLStream, CQELS and INSTANS do not provide reasoning features.

Incremental reasoning helps reasoners to handle streaming data by incrementally maintaining the materialization of the knowledge base. By only considering the data that is subject to change, incremental reasoning tries to avoid re-materializing the complete knowledge base [5,13]. However, also this approach is subject to limitation and assumptions.

Despite all effort, reasoning capabilities remain limited due to the gap between the changing frequency of streaming data and the computing time demanded by complex reasoning algorithms. Cascading reasoning is a concept presented by Stuckenschmidt et al. [17] to deal with this problem. The aim is to construct a processing hierarchy by exploiting the trade-off between the complexity of the reasoning method and the frequency of the data stream the reasoner

[4] http://www.w3.org/community/rsp/.

Table 1. Reasoning support in state of the art RDF Stream Processing solutions

	Background knowledge	Reasoning capabilities
C-SPARQL	Y	RDFS
SPARQLStream	Y	N
EP-SPARQL	Y	RDFS (in Prolog)
CQELS	Y	N
Sparkwave	Y	RDFS subset
INSTANS	N	N

is able to handle. At the lower levels of the hierarchy, with high frequency data streams, we focus on filtering to reduce the change frequency. The higher in the hierarchy the more complex reasoning can be applied. This approach helps to avoid feeding high frequency data directly to complex reasoners.

StreamRule [12] is a 2-tier approach combining stream processing with rule-based non-monotonic incremental Answer Set Programming (ASP) to enable the ability of reasoning over data streams. The novelty of this approach is that the size of the input stream towards the reasoner is reduced by the stream processor as the reasoning task becomes more computationally intensive. To allow this a feedback loop from the reasoner towards the stream processing is needed.

3 Problem Statement and Contributions

The available RSP engines aim filtering and querying on streaming data but lack support for complex OWL 2 DL reasoning. To perform such complex reasoning we need traditional DL reasoners which are computationally intensive and not capable of handling streaming data. From the analysis of the state of the art, we identified two problems we would like to address with this research:

P1: When integrating an RSP engine together with an OWL reasoning engine using a pipeline architecture, as worked out by Mileo et al. [12] for ASP reasoning, effective stream processing is necessary. Because of the limitations of the reasoners towards streaming data, it is important to neglect irrelevant data streams. Moreover, it is important to choose appropriate window parameters. Today, window parameters and query conditions are defined in advance. There is a lack of adaptive stream reasoning taking into account the changes in stream characteristics and domain knowledge at runtime.

P2: IoT and sensor networks will only bring more streaming data flooding our networks. One of the solutions to deal with this is to (pre-)process data close to its source [11]. This approach fits with IoT architectures proposing edge computing as an intermediate layer between data acquisition and the cloud based processing layer [1]. This intermediate layer allows filtering and aggregation of data, resulting in reduced network congestion, less latency and improved scalability. We believe that this edge computing approach, is

in line with the idea of cascading reasoners to reduce the change frequency of the data [17], and can be useful to increase reasoning capabilities on streaming data.

Out of the discussed problems we formulate the following two hypotheses:

H1: When integrating an RSP engine and an OWL reasoner in a pipeline architecture, a context-dependent controller using stream characteristics and domain knowledge can adapt at runtime, the window parameters and filtering conditions of the RSP engine, in such a way it increases the throughput of the pipeline.

H2: A federated reasoning platform, making use of context-awareness, to combine low-level reasoners positioned close the data stream generator, and high-level reasoners positioned deeper in the network, will leverage the reasoning capabilities over streaming data. By considering the context in which the data streams are generated the applicable domain knowledge can be narrowed, streaming frequency can be reduced and reasoning capabilities can be leveraged.

The following research questions are targeted to prove these hypotheses:

Q1: *How can adaptive stream reasoning be implemented?*
The aim is to investigate how the actual stream characteristics and domain knowledge can be used to adapt window parameters and query conditions at runtime.

- **Q1.1:** *Do there exists relationships between stream characteristics, window parameters and throughput which can be used to adapt the window parameters at runtime?*
- **Q1.2:** *Can information incorporated in the domain knowledge be used to adapt the window parameters at runtime?*
- **Q1.3:** *Can information incorporated in the domain knowledge be used to change query conditions at runtime?*

Q2: *How can context-awareness be exploited to manage domain knowledge among distributed reasoning systems?*
Low-level reasoning should be performed close to the network edges and the data stream generators. To reduce the size and complexity of the knowledge bases among the different systems context information should be used.

- **Q2.1:** *How can context-awareness be used to distribute domain knowledge among different reasoning systems?*
- **Q2.2:** *How to manage the distributed domain knowledge when the context changes?*
- **Q2.3:** *How to deal with changes in the domain knowledge that is shared among different reasoning systems?*

Q3: *How to build a federated context-aware reasoning system in which distributed adaptive stream reasoning is combined with a global reasoning system using the results of Q1 and Q2?*
Adaptive RSP engines (Q1) should be placed close to the data stream generators and being supervised by a more complex reasoning system for supporting global decision making (Q2).

4 Research Methodology and Approach

The different steps and approaches we will use to answer the research questions are discussed in the following section making use of Fig. 1.

We start our research by creating an integrated reasoning environment combining both an RSP engine and a OWL DL reasoner (Fig. 1 Q1). The output of the RSP engine will be forwarded to and processed by an OWL DL reasoner. The resulted framework allows us to perform experiments with different RSP engines and OWL DL reasoners to process sensor data streams. The tests will focus on throughput and latency and keep into account different semantics of current RSP engines. The outcome of the conducted experiments will lead to the selection of an RSP engine and OWL reasoner pair which will be used in later research.

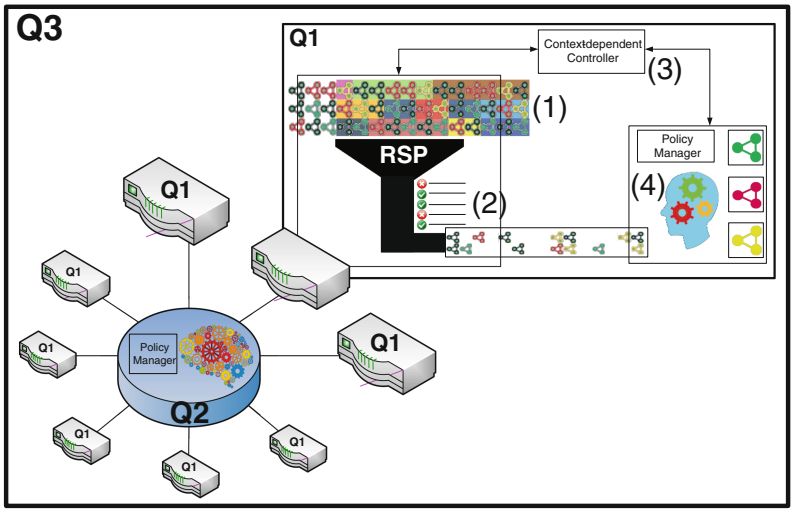

Fig. 1. Research approach

In a second phase we look how adaptive windows (Fig. 1 Q1(1)) and adaptable query conditions (Fig. 1 Q1(2)) can be integrated in the RSP engine. We investigate how these features can be used to increase overall querying performance. More specific, we want to know the impact of different window parameters on throughput, latency, network load and resource usage.

Using the outcome of the previous phase, we introduce a context-dependent controller (Fig. 1 Q1(3)) that controls the RSP engine based on available domain knowledge. We investigate how context information included in the knowledge base, can be used by the context-dependent controller to determine windows parameters and query conditions used by the RSP engine. Results of the previous phase, about the impact of window parameters, should be taken into account by the controller when determining window parameters.

Next, we focus on the OWL DL reasoning (Fig. 1 Q1(4)). We investigate how modularization of the ontology in combination with context-awareness can be used to improve reasoning performance. For making this possible, we think about the introduction of semantic policies and rules which allow to neglect or deactivate certain modules based on context information available in the knowledge base.

In the next phase, we focus on the distribution of the domain knowledge among reasoners positioned in different locations (Fig. 1 Q2). A centralized node will be the only reasoning system dealing with the complete knowledge base. It will manage and distribute knowledge over the edge reasoners depending on their context. To apply context-awareness we have to identify which context information is available, how we can access it, and how we will manage it. Once the distributed system is running, we have to deal with synchronization of domain knowledge and changes in the context of the different systems.

The latter part of the research consist of putting together the solutions found in Q1 and Q2, in such a way we get a federated context-aware stream reasoning platform (Fig. 1 Q3). Success of this step is highly dependent of previous results.

5 Initial Investigation

As initial step a literature study on the state of the art of stream reasoning and the RSP engines has been conducted. The most mature technologies where presented in Sect. 2. At the moment we are working on a test environment for RSP engines.

To perform credible tests with the RSP and reasoning engines, it is important to do some tests in a real-life context, and not only in a simulated environment. For this we can make use of the iMinds iLab.t[5] testing facilities. These include: the Virtual Wall, the w-ilab.t testbed and the iMinds Homelab. The Virtual Wall, a fully configurable set of servers, can be used to test distributed reasoning in the future. The w-ilab.t testbed consisting of 200 sensor nodes can be used to generate and test stream reasoning and RSP engines.

6 Evaluation Plan

The researched algorithms will be evaluated using two cases. The first one consists of a pervasive healthcare use case based on a continuous care OWL DL ontology. It will be used to evaluate the performance of the solutions presented for the different research questions. On the one hand, we are interested in throughput and latency to verify processing speed of the solutions. We expect the reasoning component on the gateway (Fig. 1 Q1(1)) to be the bottleneck but we strive to prevent queuing of events. On the other hand, we want to see the impact of the choices we made on network traffic. Using local reasoning at the gateways should result in a decrease of the number of events arriving at the central reasoner component.

[5] http://ilabt.iminds.be/testbeds/.

Pervasive Healthcare. The proposed solution is explained using Fig. 2. The environment is a nursing home equipped with sensors monitoring both, the patients' room (yellow box) and the patients' health parameters (green box). The sensor data of a single room are captured by a smart gateway, presented as Q1 in Fig. 1. On their turn all these gateways are connected to a central reasoning system, presented as Q2 in Fig. 1. The continuous care ontology used in this use case models among others sensors, rooms, patients, medical diagnoses, nurses and nursing activities.

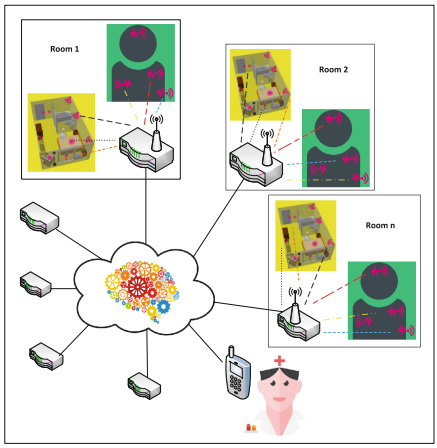

Fig. 2. Patient monitoring (Color figure online)

Suppose a patient, suffering from a concussion, stays in room 2 (Fig. 2). According to the domain ontology such a patient is sensitive to light and noise, hence has to recover in a dark and quiet environment. The patient and its room are monitored with different sensors including light, sound, temperature, blood pressure, heart rate and body temperature sensors.

Sensor values obtained on the patient and in the patient's room are collected by the smart room gateway. The gateway also receives the patients' EHR. Based on the medical diagnose in the EHR, the sense rate of the sensors, which is part of the domain knowledge, and the actual stream characteristics, the data streams can be filtered in an intelligent way. Light and sensor values, which are responsive values, should be monitored more closely than room temperature, room temperature, blood pressure and others which are of less significance.

When increasing light or sound values are registered, the gateway reasoner detects the possible unpleasant situation for the patient. A warning event will be sent to the central nursing reasoning system. Based on the current locations, competences and activities of the nurses the system decides who can visit the patient. A nursing task is assigned and sent to the mobile device of the nurse.

A second use case, about smart cities, will be used in a later part of the research to show applicability in an IoT domain different from healthcare.

Smart Cities. This use case is about parking and traffic information in smart cities. Consider a city where roads are equipped with traffic sensors and parking spot are monitored on their availability. In stead of collecting all sensor data in one single place, smart gateways capturing the data are distributed over the city. Each gateway monitors a district, a couple of streets or parking site.

Based on their location, gateways are informed about events taking place in their environment by a central reasoning system. For example public markets, manifestations or road works will lead to unaccessible roads and parking spots. Using this knowledge, smart gateways can filter out irrelevant sensor data about unaccessible places that would lead to false positives. As a result, more accurate information about parking spots can be forwarded to the central system.

In another situation, a sudden increase or decrease of the traffic intensity in a certain area, without any apparent reason, can be an indication of an unexpected incident. In this case a warning event should be sent to the central system.

In the central reasoning system, information of the different regions will be merged and used for traffic routing towards parking spots or for the creation of heat maps with information about parking spots, traffic and incidents.

7 Conclusions

This paper presented our vision on how adaptive stream processing and context-awareness can help to deal with challenges in stream reasoning. The aim is to exploit context-awareness in an attempt to bridge the gap between stream processing and complex reasoning. We described our approach and discussed it using use cases in pervasive healthcare en smart cities. With this approach we intend to contribute to future federated stream reasoning solutions.

Acknowledgments. This research was partly funded by the strategic research project DiSSeCt funded by the IWT and the CAPRADS Project co-funded by the IWT, iMinds, Luciad, Televic and JForce.

References

1. Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., Ayyash, M.: Internet of things: a survey on enabling technologies, protocols and applications. IEEE Commun. Surv. Tutorials **99**, 1 (2015)
2. 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, pp. 635–644 (2011)
3. Arasu, A., Babu, S., Widom, J.: The CQL continuous query language: semantic foundations and query execution. VLDB J. **15**(2), 121–142 (2006)
4. Barbieri, D.F.: C-SPARQL: SPARQL for continuous querying. In: Proceedings of the 18th International Conference on WWW. vol. 427, pp. 1061–1062 (2009)
5. Barbieri, D.F., Braga, D., Ceri, S., Della Valle, E., Grossniklaus, M.: Incremental reasoning on streams and rich background knowledge. In: Aroyo, L., Antoniou, G., Hyvönen, E., ten Teije, A., Stuckenschmidt, H., Cabral, L., Tudorache, T. (eds.) ESWC 2010, Part I. LNCS, vol. 6088, pp. 1–15. Springer, Heidelberg (2010)

6. Calbimonte, J.-P., Corcho, O., Gray, A.J.G.: Enabling ontology-based access to streaming data sources. In: Patel-Schneider, P.F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J.Z., Horrocks, I., Glimm, B. (eds.) ISWC 2010, Part I. LNCS, vol. 6496, pp. 96–111. Springer, Heidelberg (2010)
7. Valle, E.D., Ceri, S., Harmelen, F.V., Fensel, D.: It's a streaming world! Reasoning upon rapidly changing information. IEEE Intell. Syst. **6**, 83–89 (2009)
8. Forgy, C.L.: Rete: a fast algorithm for the many pattern/Many object pattern match problem. Artif. Intell. **19**(1), 17–37 (1982)
9. Komazec, S., Cerri, D., Fensel, D.: Sparkwave: continuous schema-enhanced pattern matching over RDF data streams. In: Proceedings of the 6th ACM International Conference on Distributed Event-Based Systems, pp. 58–68 (2012)
10. Le-Phuoc, D., Dao-Tran, M., Xavier Parreira, J., Hauswirth, M.: A native and adaptive approach for unified processing of linked streams and linked data. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E. (eds.) ISWC 2011, Part I. LNCS, vol. 7031, pp. 370–388. Springer, Heidelberg (2011)
11. Margara, A., Urbani, J., van Harmelen, F., Bal, H.: Streaming the web: reasoning over dynamic data. Web Seman. Sci. Serv. Agents World Wide Web **25**, 24–44 (2014)
12. Mileo, A., Abdelrahman, A., Policarpio, S., Hauswirth, M.: StreamRule: a nonmonotonic stream reasoning system for the semantic web. In: Faber, W., Lembo, D. (eds.) RR 2013. LNCS, vol. 7994, pp. 247–252. Springer, Heidelberg (2013)
13. Ren, Y., Pan, J., Zhao, Y.: Towards scalable reasoning on ontology streams via syntactic approximation. In: Proceedings of IWOD (2010)
14. Rinne, M., Nuutila, E., Törmä, S.: INSTANS: High-performance event processing with standard RDF and SPARQL. In: 11th International Semantic Web Conference ISWC 2012, vol. 914, pp. 101–104 (2012)
15. Shearer, R., Motik, B., Horrocks, I.: HermiT: A Highly-efficient OWL reasoner. In: OWLED, vol. 432 (2008)
16. Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., Katz, Y.: Pellet: a practical OWL-DL reasoner. Web Seman. Sci. Serv. Agents World Wide Web **5**, 51–53 (2007)
17. Stuckenschmidt, H., Ceri, S., Della Valle, E., van Harmelen, F.: Towards expressive stream reasoning. In: Proceedings of the Dagstuhl Seminar on Semantic Aspects of Sensor Networks, pp. 1–14 (2010)
18. Sundmaeker, H., Guillemin, P., Friess, P.: Vision and Challenges for Realising the Internet of Things. Publications Office of the European Union, Luxembourg (2010)
19. Tsarkov, D., Horrocks, I.: FaCT++ description logic reasoner: system description. In: Proceedings of the Third International Joint Conference pp. 292–297 (2006)