A scalable adaptive method for complex reasoning over semantic data streams

Thu-Le Pham

National University of Ireland Galway
thule.pham@insight-centre.org

Abstract. Data streams are the infinite sequences of data elements that are being generated by companies, social network, mobile phones, smart homes, public transport vehicles and other modern infrastructures. Current stream processing solutions can handle streams of data to timely produce new results but they lack the complex reasoning capacities that are required to go from data to actionable knowledge. Conversely, engines that can perform such complex reasoning tasks, are mostly designed to work on static data. The main aim of my research proposal is to provide a solution to perform complex reasoning on dynamic semantic information in a scalable way. At its core, this requires a solution which combines advantages of both stream processing and reasoning research areas, and has flexible heuristics for adaptation of the stream reasoning processes in order to enhance scalability.

Keywords: stream reasoning, stream processing, non-monotonic reasoning, multi-context systems

1 Introduction

The ever growing advance of the Internet and Sensor technology has brought new challenges evoked by the explosion of highly dynamic data. Large volumes of data are continuously produced from various sources, and published at a speed which exceeds by far our current methods and infrastructure for processing it. An infographic from analytics software provider Domo [1], attempts to quantify just how much data is generated in one minute online. It has been estimated that in 2013 every minute on the Internet 200 million emails were sent, 4 million queries were submitted in Google, and 2.5 million pieces of content were posted on Facebook.These numbers do not include the volumes of data coming from sensors and Internet of Things (IoT) devices. We refer to each of these dynamic data flows as a data stream.

More specifically, data streams are defined as sequences of time-varying data elements [4]. They occur in various modern applications such as environment monitoring, traffic management, space situational awareness, and so on. These real applications face several challenges because the data they need to process

[1] http://www.domo.com/learn/data-never-sleeps-2

is massive, ordered, can be incomplete, heterogeneous, and noisy. In addition to that, they have to provide timely response, therefore time delay becomes a key evaluation metric. Advances on Semantic Web & Linked Data research and standards have already provided formats and technologies for representing and sharing knowledge on the Web. In the last few years, Semantic Web technologies such as RDF, OWL, SPARQL have provided mechanisms for processing semantic data streams. However these solutions can not exhibit complex reasoning capabilities such as the ability of managing defaults, common-sense, preferences, recursion, and non-determinism. Conversely, logic-based non-monotonic reasoners can perform such tasks but are suitable for data that changes in low volumes at low frequency. Therefore, there is a clear need for design and implementation of new approaches to enable complex reasoning for web data streams.

The concept of ”stream reasoning”, as defined in [13], is considered as the application of reasoning techniques to data streams. Stream reasoning is described as “an unexplored yet high- impact research area and a new multidisciplinary approach that can provide the abstractions, foundations, methods, and tools required to integrate data streams, the Semantic Web, and reasoning systems” [13]. A variety of concrete applications highlight clearly the important need for stream reasoning technologies, such as Urban Computing [4] (i.e., the application of pervasive computing to urban environment), Smart Cities (i.e., the application of processing and understanding the information relevant for the life of a city and use it to make the city run better, faster, and cheaper) [11], and so on. Stream reasoning is definitely considered as a research area that can have a huge impact on quality of life.

From the analysis of several application scenarios, the authors in [11] extracted the key challenges for stream reasoning systems. These are challenges that we will consider in our approach on stream reasoning for the Semantic Web:

- Integration: data in most scenarios comes from multiple sources with various data types. This raises issues of representing and combining heterogeneous data under processing. Moreover, stream reasoning systems also use domain knowledge in reasoning. This background knowledge is mainly static and time-independent. This integration is challenging because retrieving and analysing large volumes of dynamic data and static knowledge during stream reasoning can be particularly expensive with current technologies.
- Scalability: the scalability is typically evaluated on two aspects. They are computational complexity (i.e., the ability to perform more complex tasks) and input size (i.e., the ability to process a larger input). It is essential that the reasoning process is scalable regarding both aspects.
- Expressivity: all scenarios aim at deriving high level knowledge from large volumes of low level knowledge. Expressivity of a reasoner is known to be inversely related to its performance - the more expressive reasoner is, the longer it takes to perform reasoning.

The overall purpose of this PhD proposal is to critically investigate how to perform complex reasoning on data streams maintaining scalability. We refer to scalability as to the ability to provide answers in an acceptable time when the throughput increases and the reasoning gets computationally intensive. We will explore a heuristic-based stream reasoning approach for the (dynamic) web of data, where query processing and non-monotonic reasoning can be adapted to continuously improve the expressivity versus scalability trade-off.

The combination is based on the principle of having a 2-tier approach (Fig. 1) where:

– Query processing is used to filter semantic data elements. We plan to use RDF stream query processing engines such as C-SPARQL [2], CQELS [10].
– Non-monotonic reasoning is used for computationally intensive tasks. In this proposal, we use Answer Set Programming (ASP) [9] over non-ground programs for the reasoning component.

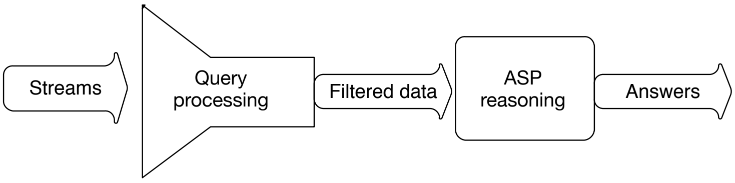

Fig. 1. 2-tier approach

2 State of the Art

There are various existing approaches aiming to perform reasoning over data streams [5]. In stream processing, the existing solutions are divided into two categories: (1) Data Stream Management Systems and (2) Complex Event Processing [11]. The former approach has some well-known engines such as CQELS and C-SPARQL that have ability to process continuously low-level data streams at high rate. The later approach considers observable raw data as primitive events and expresses composite events by some specific operators. These approaches do not manage incomplete information and do not perform complex reasoning tasks.

In the knowledge representation and reasoning community, recent works have been proposed, which attempt toward scalable reasoning using the MapReduce framework. The authors in [1] focus on distributed methods for non-monotonic rule-based reasoning. Their current works perform parallel defeasible reasoning under the assumption of stratification which imposed a severe limitation considering the range of allowed rule set and ASP is still beyond this work. Other attempts focus on extending the well established declarative complex reasoning framework of ASP with dynamic data. M. Gebser et al [9] proposed modeling approaches for continuous stream reasoning based on reactive ASP, utilizing time-decaying logic programs to capture sliding window data in a natural way. This is a first step towards gearing ASP to continuous reasoning tasks. However, these approaches still mainly process on low changing data and relatively smaller data sizes. Do et al [6] also utilize ASP in their stream reasoning system and the approach is based on the DLV system [8], which does not deal with continuous and window-based reasoning over data stream within the reasoner. A similar approach is proposed in [12], where the authors present the StreamRule framework, which combines a stream processing engine and a non-monotonic reasoner. Despite some preliminary investigations, no detailed evaluation is currently available to assess the performance of StreamRule.

3 Problem Statement and Contributions

Most of the real-time applications mentioned in Section 1 require dealing with incomplete and noisy input streams, inconsistency, defaults, qualitative preferences, and non-determinism. These forms of reasoning are computationally intensive. ASP over non-ground programs makes it possible to address these cases in offline scenarios. However, state-of-the-art ASP reasoners can not cope with huge and very dynamic input data in streaming scenarios. In this research, we intend to focus on enriching the ability of reasoning over data streams while still keeping the solution scalable by leveraging existing engines from both stream processing and non-monotonic reasoning research areas.

We will extend the approach in [12], which combines CQELS in stream query processing for data on the Web with ASP-based engines. We rely on the following assumptions: i) that not all dynamic data streams are relevant for complex reasoning tasks, ii) we consider semantically annotated RDF streams as input, iii) the dynamic stream is dynamically changing in size, rate, and accuracy. The query processing engine will be used for filtering and aggregating input data in order to provide less amount of higher-level data for the reasoner. However, we want to have a better way to integrate these two components than as a pipeline. Therefore, the questions we want to target are the followings:

a. *Is there a correlation between streaming rate, reasoning complexity, and window size which can help designing heuristics to increase the performance of a 2-tier stream reasoning framework?*
We observed that current implementation of StreamRule as a pipeline can cause a bottleneck for the reasoning component. Therefore, the non-monotonic reasoner needs to return results faster than the inputs arrive from the stream query component. We want to study the relationship between streaming rate, reasoning complexity, and window size which can be used to design heuristics that improve the performance of the stream reasoning system.

b. *How can we integrate the semantic of stream processing with the semantic of answer set programming?*

We can bridge the gap between stream processing and reasoning by integrating latest advances from both these research areas. Combining them as a pipeline is a simple way to have a stream reasoning system which can deal with complex reasoning tasks on top of query processing. However, this method can not help in managing the information flow between two different semantics. Therefore, it requires an expressive framework which can help to combine them in a better way.

c. *How can we resolve inconsistency raised in a heterogeneous distributed system?*
One of the issues which arises easily in heterogeneous and distributed systems is inconsistency. The heterogeneous and distributed data coming from noisy streams can cause conflicts within a knowledge domain. Moreover, inconsistency may happen when the system exchanges information across different knowledge domains. This makes it necessary to develop a method for handling inconsistency.

4 Research Methodology and Approach

In order to answer the above research questions, our approach unfolds in the following phases:

a. Correlation between streaming rate, reasoning complexity, and window size: Steaming rate, reasoning complexity, and window size are among the main features which can affect the performance of our stream reasoning system. This step deals with the identification of relationships between these features. For example, a correlation exists between logical window size and streaming rate: faster streams are more likely to produce query matches, so they require smaller window sizes, unless the speed of the reasoning process is increased by faster hardware. In order to discover such correlations, we intend to follow these steps:
   - Identify classes of reasoning tasks and their complexity, including a qualitative and quantitative analysis where possible.
   - For each reasoning task, we conduct experiments on StreamRule to observe the behaviour of each component in the system. This observation can be help to discover the correlation which then can be translated into a heuristic.
   - Study how different combinations of heuristics can affect the performance of the system.

   The contribution of this work in stream reasoning is to help designing an adaptation mechanism for enhancing the scalability of the system. In other words, it can help to address the trade-off between complexity and scalability in dynamic environment.

b. Integration of the semantics of stream processing with the semantics of Answer Set Programming:
   Given the latest advances of both stream processing and reasoning research areas, this step aims to find an expressive representation which can capture different logics (standard RDF/SPARQL semantics for stream processing and Stable Model semantics for ASP reasoning) in a system. We intend to consider an instance of Multi-Context System (MSC) [3] for this task. MCS is a powerful method for many application scenarios where heterogeneity of logics and inter-contextual information exchange are essential. The basic idea is to leave the diverse logics and knowledge bases (called *contexts* or *nodes*) untouched, and to equip each context with a collection of so-called bridge rules in order to model the necessary information flow among contexts. The contexts themselves may be heterogeneous in the sense that they can use different logical languages and different inference systems. Moreover, MCS are capable of integrating "typical" monotonic knowledge representation logics like description logics or temporal logics, and non-monotonic logics like default logic and ASP. From this view point, we consider our state-of-the-art stream processing engine and reasoner as contexts, namely:
   - A query processing context: this context connects the whole system to the real world by receiving data streams and reduces the enormous volume of data via stream query pattern matching.
   - A non-monotonic reasoning context: this context analyses information obtained from the query processing context, extracts high level knowledge, and performs complex reasoning.

   These two contexts can exchange information via a set of bridge rules. The efficient query processing context can reduce the irrelevant data from input streams and the bridge rules can control the useful information flow from the stream processing context to the reasoning context. Moreover, in order to enable the ability of adaptation of the system, we will add a context which is called "control context" to the framework (see (Fig. 2)). This element contains meta-knowledge about query processing and reasoning context and controls their behaviour. The heuristics designed in step (a) can help to develop this control component.

c. A mechanism for managing consistency in reasoning:
   MCS can enable integration at a general level between different formalisms. However, due to its distributed nature, information exchange can have unforeseen effects, and in particular cause a system to be inconsistent. To tackle this issue, we aim to analyse inconsistencies in our system, in order to understand where and why such inconsistencies occur, and how they can be managed. This will allow to specify how to handle inconsistencies and to extend the system with a consistency management mechanism. This mechanism can be improved by:
   - Extending the definition of equilibrium of MCS for capturing the dynamic property of data streams.
   - Exploiting the relationships between SPARQL 1.1. and ASP.

– Imposing different kinds of preferences on the notions of diagnosis and explanation introduced in [7].
– Establishing concrete consistency management procedures for analysis.

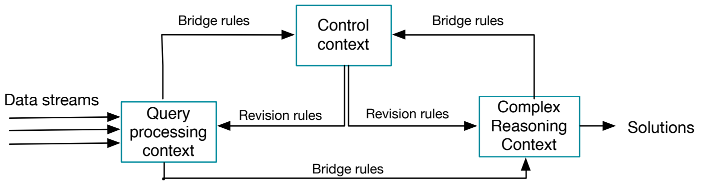

Fig. 2. Conceptual Framework

5 Initial Investigation

In our initial investigation, we have conducted an experiment for better understanding the nature of the correlation between (event-based) window size and streaming rate (research question a). This experiment mainly focused on the performance of ASP reasoning with different streaming rates. We used the state-of-the-art ASP reasoner clingo 4.3.0 [2] and Java 7. The experiment were conducted on a machine running Debian GNU/Linux 6.0.10, containing 8-cores of 2.13 GHz processor and 64 GB RAM.

The ASP rule set we used for this experiment includes 10 rules which have 2 negation-as-failure rules. We executed the ASP reasoner with various amount of input data (events) from 100 to 50000. The trend line of the processing time is illustrated in (Fig. 3). This type of trend line shows that given a unit of time, for some of certain streaming rates, there is a corresponding (event-based) window size which can help to reduce the processing time of the system to less than the unit of time. For example, given a unit of time is 1 second and a streaming rate 20000 events/second, if we feed all 20000 events to the reasoner, it will take 1232 milliseconds for processing. However, if we divide 20000 events into 10 groups of 2000 events and stream these groups to the reasoner, it will take 720 milliseconds for processing whole 20000 events.

Based on the above experiment, we can find an optimal window size for a given streaming rate for a particular ASP program for reducing the processing time of the system. However, this conclusion holds iff there is no dependency between input events for the reasoning component. This assumption can not be applied for many real scenarios and move investigation is required to understand how we can relax this assumption. Therefore, the next steps will be: i) study how

[2] http://sourceforge.net/projects/potassco/files/clingo/4.3. 0/

to relax this assumption to still find an optimal window for a given streaming rate, ii) investigate correlation with different windows and complexity levels of reasoning.

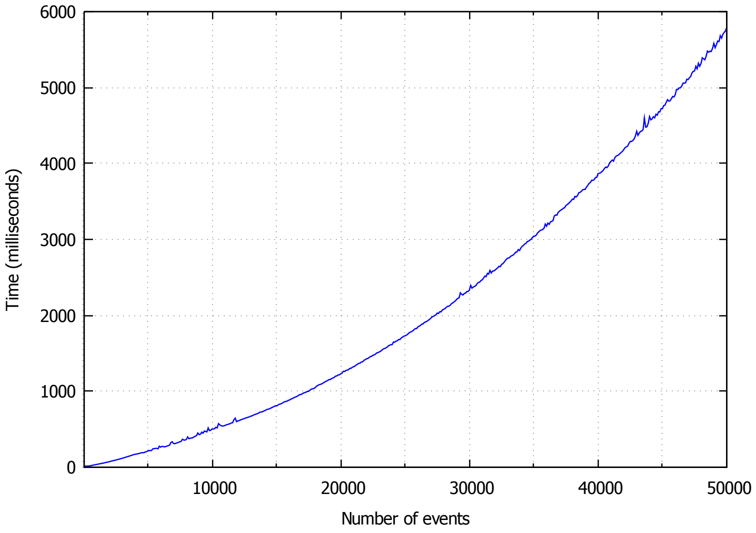

Fig. 3. Offline processing time

6 Evaluation Plan

An evaluation plan is an important step to observe the efficiency of our stream reasoning approach and compare it with similar solutions. At this very initial stage of my PhD, I foresee to conduct two evaluations:

- **System Evaluation**: In order to evaluate our system, we will provide the formal proof of soundness and completeness of the formalism using MCS, including the consistency check for the theoretical evaluation. In the experimental evaluation, we consider these following metrics:
  - Complexity, expressed in the number of rules and types of rules within a logic program.
  - Dataset size, expressed in the number of facts in the input.
  - Latency, as the time required from receiving input data in the stream query processor to providing the output as answer sets.

  We will analyse a list of reasoning tasks which are in different complexity levels. This step can be based on real scenarios from the EU project CityPulse

[3]. Moreover, we can collect real data streams from this project for designing evaluation. We will conduct experiments with different combinations of heuristics for testing the performance of our system.

- **Comparison**: It is important to provide the baseline for comparing our system with existing systems. We want to set up a benchmark which should be sufficiently generic for a fair comparison. Three mentioned metrics and the benchmark can be used to enable the comparison with other methods.

7 Conclusion

In this paper, we have described the emerging challenges of stream reasoning for the web of data, identified questions in this area we want to tackle, and also proposed a methodology and an approach to target them. We also have presented the initial investigation of the correlation between streaming rate and window-size, and a tentative evaluation plan for testing our approach. The goal of my PhD is to enable complex reasoning on data streams so that we can bridge the gap between stream processing and stream reasoning and enable a new market of applications to be built on Semantic Web streams. In relation to this work, we are aware of relate activities on RDF stream processing standards [4] and ASP-based stream reasoning [5,6]. This work is partially supported by "CityPulse: real-time IoT stream processing and large-scale analytics for smart city applications".

References

1. Antoniou, G., Batsakis, S., Tachmazidis, I.: Large-scale reasoning with (semantic) data. In: Proceedings of the 4th International Conference on Web Intelligence, Mining and Semantics (WIMS14). p. 1. ACM (2014)
2. Barbieri, D.F., Braga, D., Ceri, S., Grossniklaus, M.: An execution environment for c-sparql queries. In: Proceedings of the 13th International Conference on Extending Database Technology. pp. 441–452. ACM (2010)
3. Brewka, G., Ellmauthaler, S., Pührer, J.: Multi-context systems for reactive reasoning in dynamic environments. In: Proceedings of the International Workshop on Reactive Concepts in Knowledge Representation (ReactKnow). pp. 23–30 (2014)
4. Della Valle, E., Ceri, S., Barbieri, D., Braga, D., Campi, A.: A first step towards stream reasoning. Future Internet–FIS 2008 pp. 72–81 (2009)
5. Della Valle, E., Schlobach, S., Krötzsch, M., Bozzon, A., Ceri, S., Horrocks, I.: Order matters! harnessing a world of orderings for reasoning over massive data. Semantic Web 4(2), 219–231 (2013)
6. Do, T.M., Loke, S.W., Liu, F.: Answer set programming for stream reasoning. In: Advances in Artificial Intelligence, pp. 104–109. Springer (2011)
7. Eiter, T., Fink, M., Schüller, P., Weinzierl, A.: Finding explanations of inconsistency in multi-context systems. Artificial Intelligence 216, 233–274 (2014)

[3] http://www.ict-citypulse.eu/

[4] https://www.w3.org/

[5] http://www.kr.tuwien.ac.at/research/projects/dhsr/

[6] http://potassco.sourceforge.net

8. Eiter, T., Ianni, G., Schindlauer, R., Tompits, H.: Dlv-hex: Dealing with semantic web under answer-set programming. In: Proc. of ISWC (2005)
9. Gebser, M., Grote, T., Kaminski, R., Obermeier, P., Sabuncu, O., Schaub, T.: Answer set programming for stream reasoning. CoRR, abs/1301.1392 (2013)
10. Le-Phuoc, D., Dao-Tran, M., Parreira, J.X., Hauswirth, M.: A native and adaptive approach for unified processing of linked streams and linked data. In: The Semantic Web–ISWC 2011, pp. 370–388. Springer (2011)
11. Margara, A., Urbani, J., van Harmelen, F., Bal, H.: Streaming the web: Reasoning over dynamic data. Web Semantics: Science, Services and Agents on the World Wide Web 25, 24–44 (2014)
12. Mileo, A., Abdelrahman, A., Policarpio, S., Hauswirth, M.: Streamrule: a non-monotonic stream reasoning system for the semantic web. In: Web Reasoning and Rule Systems, pp. 247–252. Springer (2013)
13. Valle, E.D., Ceri, S., Harmelen, F.v., Fensel, D.: It's a streaming world! reasoning upon rapidly changing information. IEEE Intelligent Systems 24(6), 83–89 (2009)