Stream Reasoning: Where We Got So Far

Davide Barbieri, Daniele Braga, Stefano Ceri, Emanuele Della Valle, and Michael Grossniklaus

Dip. di Elettronica e Informazione, Politecnico di Milano, Milano, Italy
email: {dbarbieri, braga, ceri, dellavalle, grossniklaus } @elet.polimi.it

Abstract. Data Streams - unbounded sequences of time-varying data elements - are pervasive. They occur in a variety of modern applications including the Web where blogs, feeds, and microblogs are increasingly adopted to distribute and present information in real-time streams. We foresee the need for languages, tools and methodologies for representing, managing and reasoning on data streams for the Semantic Web. We collectively name those research chapters Stream Reasoning. In this extended abstract, we motivate the need for investigating Steam Reasoning; we characterize the notion of Stream Reasoning; we report the results obtained by Politecnico di Milano in studying Stream Reasoning from 2008 to 2010; and we close the paper with a short review of the related works and some outlooks.

1 Motivation

The use of the Internet as a major source of information has created new challenges for computer science and has let to significant innovation in areas such as databases, information retrieval and semantic technologies. Currently, we are facing another major change in the way information in provided. Traditionally information used to be mostly static with changes being the exception rather than the rule. Nowadays, more and more dynamic information, which used to be hidden inside dedicated systems, is getting available to decision makers. A large part of this dynamic information is an (almost) "continuous" flow of information with the recent information being more relevant as it describes the current state of a dynamic system.

Continuous processing of these flows of information (namely data streams) has been largely investigated in the database community [1]. Specialized Data Stream Management Systems (DSMS) are available on the market and features of DSMS are appearing also in major database products, such as Oracle and DB2.

On the contrary, continuous processing of data streams *together with rich background knowledge* requires specialized reasoners, but work on semantic technologies is still focusing on rather static data. In existing work on logical reasoning, the knowledge base is always assumed to be static (or slowly evolving). There is work on changing beliefs on the basis of new observations [2], but the solutions proposed in this area are far too complex to be applicable to gigantic data streams of the kind illustrated in the oil production example above.

As argued in [3], we strongly believe that there is a need to close this gap between existing solutions for belief update and the actual needs of supporting decision process based on data streams and rich background knowledge. We named this little explored, yet high-impact research area Stream Reasoning.

2 Stream Reasoning

In this section we characterize the notion of Stream Reasoning giving a definition and explaining what is peculiar to it.

Definition 1. ***Stream Reasoning****: logical reasoning in real time on gigantic and inevitably noisy data streams in order to support the decision process of extremely large numbers of concurrent users.*

Peculiar to stream processing, and thus also to Stream Reasoning, are the notions of window [4] and continuous processing [5]. In the following we characterize Stream Reasoning with regards to these two notions.

Window Traditional reasoning problems are based on the idea that all the information available should be taken in to account when solving the problem. In Stream Reasoning, we eliminate this principle and restrict reasoning to a certain window of concern which consists of a subset of statement recently observed on the stream while previous information is ignored. This is necessary for different reasons. First of all, ignoring older statements allows us to saves computing resources in terms of memory and processing time to react to important events in real time. Further, in many real-time applications there is a silent assumption that older information becomes irrelevant at some point.

Continuous Processing Traditional reasoning approaches are based on the idea that the reasoning process has a well defined beginning (when a request is posed to the reasoner) and end (when the result is delivered by the system). In Stream Reasoning, we move from this traditional model to a continuous processing model, where requests in terms of reasoning goals are registered at the reasoner and are continuously evaluated against a knowledge base that is constantly changing.

3 Current State of Development in Politecnico di Milano

Deductive Stream Reasoning has been studied at Politecnico di Milano. In [6], we specified a general and flexible architecture - based on the LarKC conceptual architecture [ICSC08] - for reasoning in the presence of data streams and rich background knowledge. This architecture leverages existing DSMS and SPARQL engines and anticipates the possibility to extend RDF graphs, by introducing

RDF streams and extending SPARQL to continuously processing RDF streams observed through windows. Both these extensions are present in Continuous SPARQL (or simply C-SPARQL) [7].

C-SPARQL adds RDF streams to the data types supported by SPARQL, much in the same way in which the stream type has been introduced to extend relations in relational data stream management systems. An RDF stream is defined as an ordered sequence of pairs, where each pair is made of an RDF triple and its timestamp τ:

$$\begin{gathered} \ldots \\ (\langle subj_i, pred_i, obj_i \rangle\ ,\ \tau_i) \\ (\langle subj_{i+1}, pred_{i+1}, obj_{i+1} \rangle\ ,\ \tau_{i+1}) \\ \ldots \end{gathered}$$

Timestamps can be considered as *annotations* of RDF triples, and are monotonically non-decreasing in the stream ($\tau_i \leq \tau_{i+1}$). More precisely, timestamps are not strictly increasing because they are not required to be unique. Any (unbounded, though finite) number of consecutive triples can have the same timestamp, meaning that they "occur" at the same time, although sequenced in the stream according to some positional order.

Consider the program querying Twitter for tweets containing "I'm reading". The result is a stream that get continuously update with new tweets that match the search criteria. Such stream can be represented as an RDF stream of triples, stating what each twitter user is reading, annotate with the posting date-time.

$$\begin{gathered} \ldots \\ (\langle :Giulia, :isReading, :Twilight \rangle, \text{2010-02-12T13:34:41}) \\ (\langle :John, :isReading, :TheLordOfTheRings \rangle, \text{2010-02-12T13:36:28}) \\ \ldots \end{gathered}$$

As a simple example of C-SPARQL, we informally explain a query that counts how many followers of Giulia have been reading a book in the last hour.

```
1  REGISTER QUERY NumberOfGiuliaFollowersWhoAreReadingBooks COMPUTE EVERY 15m
       AS
2  SELECT count(distinct ?user) as ?numberOfGiuliaFollowersReadingBooks
3  FROM <http://streamingsocialdata.org/followersNetwork>
4  FROM STREAM <http://streamingsocialdata.org/reading>
5                      [RANGE 1h STEP 15m]
6  WHERE { ?user :follows :Giulia.
7          ?user :isReading ?x .
8          ?x a :Book . }
```

At line 1, the REGISTER clause is use to tell the C-SPARQL engine that it should register a continuous query, i.e. a query that will continuously compute answers to the query. The COMPUTE EVERY clause states the frequency of every new computation, in the example every 15 minutes. The query joins background and streaming knowledge. At line 3, the standard SPARQL clause FROM is used to load in the default graph an RDF graphs describing who is following who on Twitter. At line 4, the clause FROM STREAM is used to tell the C-SPARQL engine to process the stream exemplified above. Next, line 5 defines the window of observation over the RDF stream. The window considers all the stream triples in the last hour, and is advanced every 15 minutes. The content of the window is loaded in the default graph as if it were a standard RDF graph.

However, every 15 minutes new triples enter into the window and old triples exit from the window, thus the default graph is modified accordingly. Note that the query result does not change during the slide interval, and is only updated at every slide change. Triples arriving in the stream between these points in time are queued until the next slide change and do not contribute to the result until then. The WHERE clause is standard; it includes a set of matching patterns that operates over the default graph as in a standard SPARQL query. The results are project by the SELECT clause at line 2, which also count the number of distinct bindings of the variable ?user.

In [8], we propose a formal semantics of C-SPARQL language together with a query graph model which is an intermediate representation of queries devoted to optimization. We discuss the features of an execution environment, based on [6] that leverages existing technologies and we introduce optimizations in terms of rewriting rules applied to the query graph model, so as to efficiently exploit the execution environment.

Finally in [9], we elaborate on the deductive reasoning support to C-SPARQL. The [7] version of C-SPARQL can work under entailment regimes different from RDF simple semantics, but at the cost of recomputing, when the window slides, any deduction that depends from the triples in the window. In [9], we propose a technique for incremental maintenance of materializations of ontological entailments that exploits the transient nature of streaming data. By adding expiration time information to each RDF triple, we show that it is possible to compute a new complete and correct materialization whenever the window slides, by dropping explicit statements and entailments that are expired, and then incrementally adding all the deduction that depends on the new triples that entered the window and tagging them with a correct expiration time.

The following example extends the C-SPARQL query above by requesting to count only the followers of Giulia who were only twittering about books. The ontological definition of user, who only twittered about books, is given in OWL-RL as follows:

```
:UserOnlyTwitteringAboutBooks rdfs:subClassOf :User;
  rdfs:subClassOf [
          a owl:Restriction;
          owl:onProperty :tweets;
          owl:allValuesFrom :Book;
  ] .
```

An example of C-SPARQL query that leverages such ontological definition can be the following one:

```
REGISTER QUERY NumberOfGiuliaFollowersWhoAreReadingBooks COMPUTE EVERY 15m
     AS
SELECT count(distinct ?user) as ?numberOfGiuliaFollowersOnlyReadingBooks
FROM <http://streamingsocialdata.org/followersNetwork>
FROM STREAM <http://streamingsocialdata.org/reading>
                    [RANGE 1h STEP 15m]
WHERE { ?user :follows :Giulia.
        ?user a :UserOnlyTwitteringAboutBooks .
        ?user :isReading ?x .
        ?x a :Book . }
```

In order to be evaluated, this C-SPARQL query requires the engine to reason about the triple in the window and incrementally evaluate which users satisfy the ontological definition of UserOnlyTwitteringAboutBooks.

4 Related Work

Two approaches, alternative C-SPARQL exists: Streaming SPARQL [10] and Time-Annotated SPARQL (or simply TA-SPARQL) [11]. Both languages introduce the concept of window over stream, but only C-SPARQL brings the notion of continuous processing, typical of stream processing, into the language; all the other proposal still rely on permanent storing the stream before processing it using one-shot queries. Moreover, only C-SPARQL proposes an extension to SPARQL to support aggregate functions[1]. Such a support for aggregates is designed specifically to exploit optimization techniques [8] that push, whenever possible, aggregates computation as close as possible to the raw data streams.

We believe that C-SPARQL is an important piece of the Stream Reasoning puzzle, it supports OWL RL entailment [9] is only a first step toward Stream Reasoning.

There are many ideas of how to support incremental reasoning on the different levels of complexity. In particular, there are approaches for the incremental maintenance of materialized views in logic [13], object-oriented [14] and graph databases [15], for extensions of the well known RETE algorithm for incremental rule-based reasoning [16, 17] and even some initial ideas of how to support incremental reasoning in description logics [18, 19]. All of these methods operate incrementally, thus they are appropriate to treat information that changes, but none of them is explicitly dedicated to process an (almost) continuous flow of information with the recent information being more relevant.

5 Outlook

In these two years of work on Stream Reasoning, we have been mainly working on C-SPARQL and its corresponding infrastructure. We believe they provide an excellent starting point for Stream Reasoning research, because a) RDF streams provide an RDF-based representation of heterogeneous streams and b) C-SPARQL construct query can create RDF snapshots that can feed information into existing reasoning mechanisms. We already made a first step in this direction, investigating the incremental maintenance of ontological entailment materializations [9]. This approach needs to be generalized to more expressive ontological languages.

Moreover, the extraction of patterns from data streams is subject of ongoing research in machine learning. For instance, results from statistical relational learning are able to derive classification rules from example data in very effective

[1] For a comparison between C-SPARQL and SPARQL 1.1 support for aggregates see [12]

ways. In our future work, we intend to link relational learning methods with C-SPARQL to facilitate pattern extraction on top of RDF streams.

Finally, we envision the possibility to leverage recent developments in distributed and parallel reasoning [20, 21] as well as the compilation of reasoning tasks into queries [22] for scaling up to gigantic data streams and to extremely large numbers of concurrent reasoning tasks.

Acknowledgements

The work described in this paper is has been partially supported by the European project LarKC (FP7-215535).

References

1. Garofalakis, M., Gehrke, J., Rastogi, R.: Data Stream Management: Processing High-Speed Data Streams (Data-Centric Systems and Applications). Springer-Verlag New York, Inc., Secaucus, NJ, USA (2007)
2. Gaerdenfors, P., ed.: Belief Revision. Cambridge University Press (2003)
3. Della Valle, E., Ceri, S., van Harmelen, F., Fensel, D.: It's a Streaming World! Reasoning upon Rapidly Changing Information. IEEE Intelligent Systems **24**(6) (2009) 83–89
4. Arasu, A., Babcock, B., Babu, S., Datar, M., Ito, K., Nishizawa, I., Rosenstein, J., Widom, J.: STREAM: The Stanford Stream Data Manager (Demonstration Description). In: Proc. ACM Intl. Conf. on Management of data (SIGMOD 2003). (2003) 665
5. Babu, S., Widom, J.: Continuous Queries over Data Streams. SIGMOD Rec. **30**(3) (2001) 109–120
6. Della Valle, E., Ceri, S., Barbieri, D.F., Braga, D., Campi, A.: A First Step Towards Stream Reasoning. In: Proc. Future Internet Symposium (FIS). (2008) 72–81
7. Barbieri, D.F., Braga, D., Ceri, S., Valle, E.D., Grossniklaus, M.: C-sparql: Sparql for continuous querying. In: WWW. (2009) 1061–1062
8. Barbieri, D.F., Braga, D., Ceri, S., Grossniklaus, M.: An Execution Environment for C-SPARQL Queries. In: Proc. Intl. Conf. on Extending Database Technology (EDBT). (2010)
9. 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., Hyvonen, E., eds.: ESWC. Lecture Notes in Computer Science, Springer (2010)
10. Bolles, A., Grawunder, M., Jacobi, J.: Streaming SPARQL – Extending SPARQL to Process Data Streams. In: Proc. Europ. Semantic Web Conf. (ESWC). (2008) 448–462
11. Rodriguez, A., McGrath, R., Liu, Y., Myers, J.: Semantic Management of Streaming Data. In: Proc. Intl. Workshop on Semantic Sensor Networks (SSN). (2009)
12. Barbieri, D.F., Braga, D., Ceri, S., Della Valle, E., Grossniklaus, M.: Feedbacks on sparql 1.1 support for aggregates. Technical report, LarKC Project. (February 2010) Available on line at http://wiki.larkc.eu/c-sparql/sparql11-feedback.
13. Volz, R., Staab, S., Motik, B.: Incrementally maintaining materializations of ontologies stored in logic databases. J. Data Semantics **2** (2005) 1–34

14. Kuno, H., Rundensteiner, E.: Incremental maintenance of materialized object-oriented views in multiview: Strategies and performance evaluation. IEEE Transactions on Data and Knowledge Engineering **10**(5) (september 1998) 768–792
15. Zhuge, Y., Garcia-Molina, H.: Graph structured views and their incremental maintenance. In: Proceedings of the Fourteenth International Conference on Data Engineering. (1998) 116 – 125
16. Fabret, F., Regnier, M., Simon, E.: An adaptive algorithm for incremental evaluation of production rules in databases. In: Proceedings of the 19th International Conference on Very Large Data Bases. (1993) 455 – 466
17. Berster, B.: Extending the rete algorithm for event management. In: Proceedings of the Ninth International Symposium on Temporal Representation and Reasoning (TIME'02). (2002)
18. Cuenca-Grau, B., Halaschek-Wiener, C., Kazakov, Y.: History matters: Incremental ontology reasoning using modules. In: Proceedings of the 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007. Volume 4825 of Lecture Notes in Computer Science. (2007) 183–196
19. Parsia, B., Halaschek-Wiener, C., Sirin, E.: Towards incremental reasoning through updates in owl-dl. In: Proceedings of the Reasoning on the Web Workshop at WWW2006, Edinburgh, UK. (2006)
20. Urbani, J., Kotoulas, S., Maassen, J., van Harmelen, F., Bal, H.: Owl reasoning with webpie: calculating the closure of 100 billion triples. In: Proceedings of ESWC 2010, Heraklion, Crete. (2010)
21. Schlicht, A., Stuckenschmidt, H.: Distributed resolution for expressive ontology networks. In: Web Reasoning and Rule Systems. (2009)
22. Lutz, C., Toman, D., Wolter, F.: Conjunctive query answering in the description logic el using a relational database system. In: IJCAI'09: Proceedings of the 21st international jont conference on Artifical intelligence, San Francisco, CA, USA, Morgan Kaufmann Publishers Inc. (2009) 2070–2075