Towards Efficient Schema-Enhanced Pattern Matching over RDF Data Streams

Srdjan Komazec and Davide Cerri

Semantic Technology Institute (STI) Innsbruck, University of Innsbruck
Technikerstraße 21a, 6020 Innsbruck, Austria
{srdjan.komazec,davide.cerri}@sti2.at

Abstract. Data streams, often seen as sources of events, have appeared on the Web. Event processing on the Web needs however to cope with the typical openness and heterogeneity of the Web environment. Semantic Web technology, meant to facilitate data integration in an open environment, can help to address heterogeneities across multiple streams. In this paper we discuss an approach towards efficient pattern matching over RDF data streams based on the Rete algorithm, which can be considered as a first building block for event processing on the Web. Our approach focuses on enhancing Rete with knowledge from the RDF schema associated with data streams, so that implicit knowledge can contribute to pattern matching. Moreover, we cover Rete extensions that cope with the streaming nature of the processed data, such as support for temporal operators, time windows, consumption strategies and garbage collection.

Keywords: RDF, data streams, pattern matching, event processing, semantic Web, inference, Rete.

1 Introduction

Data streams are becoming more and more common on the Web. Many streams regarding e.g. stock exchange movements, weather information, sensor readings, or social networking activity notifications are already present, and platforms to collect and share these streams, such as Patchube,[1] have appeared. These data streams can often be seen as sources of events, and the opportunity to efficiently detect event occurrences at the Internet scale while correlating different sources on this *"Web of Events"* could open the doors for new powerful applications.

Event processing is an established computing paradigm that provides approaches and techniques to process event streams and to respond in a timely fashion. The combination of event processing techniques with data streams distributed across the Web comes as a natural fit; event processing on the Web needs however to cope with the typical openness and heterogeneity of the Web environment. Semantic Web technologies are meant to facilitate data integration in an open environment, thus can help to overcome these problems by using machine processable descriptions to resolve heterogeneities across multiple streams.

[1] `http://www.patchube.com/`

For example, Semantic Sensor Web [11] represents an attempt to collect and process avalanches of data about the world using semantic Web technologies.

While there are a few systems bridging semantic Web technologies and event processing, bringing together these two areas and providing an efficient solution for event processing on the Web is an open research topic. In this paper we present a work-in-progress solution towards efficient pattern matching over RDF data streams, which can be considered as a first building block in this vision.

2 Problem Statement and Related Work

A solution for pattern matching over RDF data streams in the context of event processing on the Web needs to address issues along two different dimensions. First, the presence of RDF schema entailments can materialise RDF statements at runtime, which is not the usual case in event processing systems. Entailed statements, even if they are not explicitly present in the input streams, can contribute to pattern completion. Second, the streaming nature of the data calls for support to express and match patterns taking into account the temporal dimension, and thus for operators that are common for event processing systems but are not part of the usual notion of RDF pattern matching (e.g. in SPARQL).

ETALIS [2] is a Complex Event Processing system providing a number of features such as evaluation of out-of-order events, computation of aggregate functions, dynamic insertion and retraction of patterns, and several garbage collection policies and consumption strategies. To operate on RDF streams ETALIS provides EP-SPARQL, an extension of SPARQL that introduces a number of temporal operators (compliant to ETALIS ones) used to combine RDF graph patterns along time-related dependencies. The presence of background knowledge, in the form of an RDFS ontology, is also supported. EP-SPARQL patterns are evaluated on top of ETALIS, and thus can benefit from all ETALIS features.

C-SPARQL [4] is a SPARQL-based stream reasoning system and language extended with the notions of RDF streams, time windows, handling of multiple streams and aggregation functions. It is suited for cases in which a significant amount of static knowledge needs to be combined with data streams (e.g., coming from heterogeneous and noisy data sensors) in order to enable time-constrained reasoning. The RDF statements from the streams are fed into pre-reasoners performing incremental RDF schema-based materialisation of RDF snapshots, which are further fed into reasoners to which SPARQL queries are submitted [3].

These approaches fall in our problem space, but neither of them provides a complete solution. Although it covers RDF data stream processing, ETALIS does not provide native support for entailments based on RDF schema.[2] On the other hand, C-SPARQL only provides a simple `timestamp()` function to express temporal relationships between RDF patterns, but it does not provide more expressive temporal operators like ETALIS, which can hinder its application in the area of event processing. Since queries are periodically evaluated, C-SPARQL does

[2] ETALIS translates an RDFS ontology into a set of Prolog rules via an external library.

not provide truly continuous querying, and in case of short time windows, high-frequency streams and rich background knowledge, the overhead to compute the incremental materialisation of RDF snapshots becomes significant. ETALIS suffers from performance issues in case of complex pattern expressions.

The goal of our system is to provide efficient pattern matching functionalities on RDF streams, enabling the expression of temporal dependencies and taking into account RDF schema entailments. Unlike C-SPARQL, our system operates over a fixed RDF schema, does not support static background knowledge (besides the schema), and does not provide generic reasoning capabilities. We believe that a fixed schema does not significantly limit the applicability of our solution, since in most applications only data statements come through the streams (e.g., sensor readings). This makes the inclusion of schema-driven entailments simpler, as it can be realised in a pre-running step. The exclusion of background knowledge is due to performance reasons. Our system can be used as the entry block of a bigger system and has performance as a primary concern, whereas more complex operations and semantics are left to subsequent blocks (which can have less stringent performance requirements, as they operate after the first "filtering").

3 Solution

Unlike the aforementioned solutions, our system takes as foundation an efficient pattern matching algorithm: *Rete* [8]. The Rete algorithm, originally designed as a solution for production rule systems, represents a general approach to deal with many pattern/many object situations. The algorithm emphasises on trading memory for performance by building comprehensive memory structures, called α- and β-networks, designated to check respectively intra- and inter-pattern conditions. The algorithm internally preserves intermediate matches in the form of tokens which point to all contributing objects. The intrinsic dataflow nature of Rete goes in favour of using it also over data streams. However, in order to cope with our requirements, the basic Rete algorithm needs to be extended to properly address the issues of RDF schema entailments and data stream processing.

3.1 Support for Schema Entailments in Rete

The RDF semantics specification [10] includes a set of entailment rules to derive new statements from known ones (Table 1[3]). Since in our system the schema is fixed, some rules have no impact at runtime: in particular, rules rdfs5, rdfs6, rdfs8, rdfs10, rdfs11, rdfs12 and rdfs13 have in their bodies only T-box statements, thus cannot be activated by statements in the streams. Rule rdf1 is also not relevant at runtime because, if the property is not in the schema, nothing else (i.e., domain, range, sub/super-properties) can be stated about it, thus no further entailments are possible. Finally, unless we look for resources, rules rdfs4a,

[3] Rules are named as in the specification. We omit rules dealing with blank nodes and literals, since they are not relevant to our discussion.

Table 1. RDF/RDFS entailment rules

Rule name	*If*	*Then add*
rdf1	(x p y)	(p rdf:type rdf:Property)
rdfs2	(p rdfs:domain c),(x p y)	(x rdf:type c)
rdfs3	(p rdfs:range c),(x p y)	(y rdf:type c)
rdfs4a	(x p y)	(x rdf:type rdfs:Resource)
rdfs4b	(x p y)	(y rdf:type rdfs:Resource)
rdfs5	(p rdfs:subPropertyOf q),(q rdfs:subPropertyOf r)	(p rdfs:subPropertyOf r)
rdfs6	(p rdf:type rdf:Property)	(p rdfs:subPropertyOf p)
rdfs7	(p rdfs:subPropertyOf q),(x p y)	(x q y)
rdfs8	(c rdf:type rdfs:Class)	(c rdfs:subClassOf rdfs:Resource)
rdfs9	(c rdfs:subClassOf d),(x rdf:type c)	(x rdf:type d)
rdfs10	(c rdf:type rdfs:Class)	(c rdfs:subClassOf c)
rdfs11	(c rdfs:subClassOf d),(d rdfs:subClassOf e)	(c rdfs:subClassOf e)
rdfs12	(p rdf:type rdfs:ContainerMembershipProperty)	(p rdfs:subPropertyOf rdfs:member)
rdfs13	(x rdf:type rdfs:Datatype)	(x rdfs:subClassOf rdfs:Literal)

rdfs4b and rdfs8 are not relevant, since their output is not used as input by other rules. Therefore, only rules rdfs2, rdfs3, rdfs7 and rdfs9, i.e. inference based on the hierarchies of classes and properties together with their domain and range, need to be considered at runtime. The other rules are relevant only when the extended Rete network is built, based on the schema and the patterns.

A similar discussion can be found in [12], which distinguishes between rules for online and offline computation (the latter used to compute schema closure). Similarly, in our system we can pre-compute schema closure, and use it in building our extended Rete network. Our approach extends the Rete architecture with an additional network of entailment nodes, called *ε-network*, responsible for generating network objects following schema entailments. Given a schema and a set of patterns, the ε-network consists of an optimised set of nodes arranged according to the dataflow paradigm. In contrast to the typical usage of Rete in forward-chaining reasoners, i.e. detecting patterns corresponding to the bodies of entailment rules, our approach encodes in the ε-network schema-driven property hierarchies with specified domain and range definitions connected to class hierarchies. The triples that constitute the schema are not part of the data streams, therefore they are not present as data items in the network and are not used as such in the pattern-matching process. The outputs of the ε-network are connected to the appropriate inputs of the α-network. This approach retains the efficiency of Rete, and fits appropriately with the support for data stream issues.

The ε-network is built as follows. Each property and each class in the schema correspond to a node:[4] for each property p in the schema a node with the triple pattern (? p ?) is created, and for each class c in the schema a node with the triple pattern (? rdf:type c) is created. Nodes are connected by three different types of links: *S-links*, which transfer the subject of the triple to the following class node, *O-links*, which transfer the object of the triple to the following class node (where it becomes the subject), and *SO-links*, which transfer both the subject and the object of the triple to the following property node. For each subclass statement in the schema, an S-link between the corresponding class nodes is

[4] Actually, only the subset of the schema that is relevant for the patterns is considered.

created. For each subproperty statement in the schema, an SO-link between the corresponding property nodes is created. For each domain statement in the schema, an S-link from the corresponding property node to the corresponding class node is created. For each range statement in the schema, an O-link from the corresponding property node to the corresponding class node is created. A simple example of ε-network and the corresponding ontology fragment are shown in Figure 1.

```
@prefix rdf:
  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix rdfs:
  <http://www.w3.org/2000/01/rdf-schema#>.
@prefix :
  <http://example.org/ns#>

:Post          rdf:type            rdfs:Class.

:Tweet         rdf:type            rdfs:Class;
               rdfs:subClassOf     :Post.

:FBPost        rdf:type            rdfs:Class;
               rdfs:subClassOf     :Post.

:Place         rdf:type            rdfs:Class.

:TalksAbout    rdf:type            rdf:Property;
               rdfs:domain         :Post;
               rdfs:range          :Place.

:TAPositively  rdf:type            rdf:Property;
               rdfs:subPropertyOf  :TalksAbout.

:TANeutrally   rdf:type            rdf:Property;
               rdfs:subPropertyOf  :TalksAbout.

:TANegatively  rdf:type            rdf:Property;
               rdfs:subPropertyOf  :TalksAbout.
```

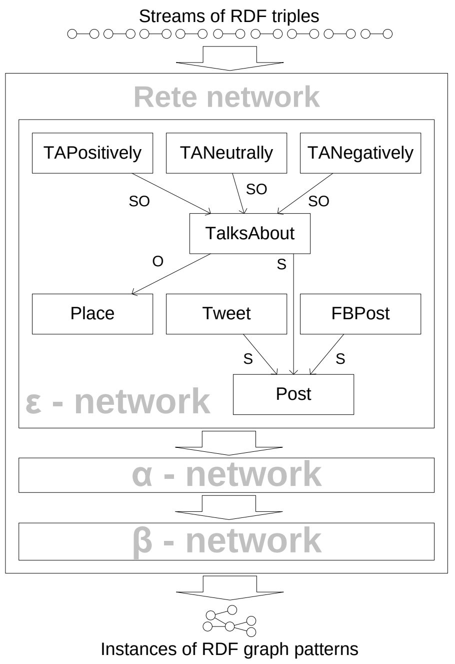

Fig. 1. Extended Rete with ε-network

When a triple enters the ε-network, it activates the node with the corresponding pattern (if it exists), in the same way as in the α-network in basic Rete. When a node is activated, it emits the corresponding triple to the α-network and it transfers the subject and/or the object of the triple (depending on the link type) to the following nodes, activating them. The activation mechanism is governed by tokens (as in basic Rete), which allows detecting multiple inference paths, in order to avoid passing the same inferred triple to the α-network more than once.

3.2 Support for Data Stream Issues in Rete

Our solution intends to incorporate and adapt the following stream processing extensions to Rete: temporal operators in RDF patterns, time windows, and data consumption strategies coupled with appropriate garbage collection approaches.

An event pattern specification language usually introduces a number of operators to build comprehensive pattern expressions. The operators can be divided into *logical operators* (conjunction, disjunction and negation), *data operators* (comparison and arithmetic), and *temporal operators*. Basic Rete provides only support for conjunctions between objects in a pattern. Support for the remaining logical and data operators has been introduced soon after the algorithm [9, 7], and our system follows the same design and implementation guidelines. Temporal operators allow to express time-dependent constraints over a set of objects in a pattern. Our system treats pattern occurrences in accordance to interval-based semantics [1], in which an occurrence spans a time interval (unlike point-based semantics in which the last contributing object defines the time of the pattern occurrence). Motivated by [5] and [14], we consider to implement the temporal operator semantics through extensions of join-nodes behaviour. The extensions allow nodes to perform interval-based comparisons to establish a temporal relationship between the joining patterns (i.e., partial graph pattern matches) by executing basic timepoint and time interval arithmetics. In order to perform the join, the node further detects if the established relationship is in accordance to the one designated by the pattern.

Rete implementations usually provide no time-window support. An initial work [13] proposes a method to automatically discard objects no longer needed by computing temporal dependency matrices over the Rete network tokens. The work in [16, 15] gives an implementation that adds new time-aware Rete nodes checking time-window boundaries during the matching process, based on temporal token dependency computations. Our system follows a similar approach by extending β-node behaviour to include time-window boundary checking. Tokens falling out of the time-window boundaries are excluded from further matching results and flagged as candidates for garbage collection. The checks are performed in the same way as for temporal operators (i.e., by establishing a relationship between the intervals, where one interval defines the time window boundaries).

Event consumption strategies (also called parameter contexts) [6, 1] resolve the issue of multiple simultaneous matches to a pattern inside of a time window. Rete implementations usually lack support for these strategies; we will consider further extensions to the join-node behaviour to enable their support. The issue of garbage collection is closely related. In [14] it is proposed to compute an object lifetime during which the object must be retained due to the participation in partial matches inside the time window. Our system will consider the same approach by implementing an incremental garbage collection process using the lifetime information to schedule deletion of objects.

4 Conclusions and Further Work

An efficient mechanism for schema-enhanced pattern detection on RDF data streams is required to address the challenges of event processing on the Web. In this paper we proposed a system based on the Rete algorithm, extended to include statements materialised through schema entailments and to support the temporal nature of data streams. We consider several dimensions as our further work. First, more precise answers to the issues of how to integrate the chosen temporal and entailment approaches into Rete will need to be defined. Second, special attention will be devoted to the analysis of mutual influence of temporal and entailment extensions. Third, performance evaluation will be conducted in order to ensure the overall system responsiveness under different conditions.

Acknowledgement. This work has been supported by the European Commission through the SUPPORT project (EU FP7 Project 242112).

References

1. Adaikkalavan, R., Chakravarthy, S.: SnoopIB: Interval-Based Event Specification and Detection for Active Databases. Data Knowl. Eng. 59, 139–165 (October 2006)
2. Anicic, D., Fodor, P., Rudolph, S., Stühmer, R., Stojanovic, N., Studer, R.: ETALIS: Rule-Based Reasoning in Event Processing. In: Reasoning in Event-Based Distributed Systems, Studies in Computational Intelligence, vol. 347, pp. 99–124. Springer (2011)
3. Barbieri, D., Braga, D., Ceri, S., Della Valle, E., Grossniklaus, M.: Incremental Reasoning on Streams and Rich Background Knowledge. In: The Semantic Web: Research and Applications, Proc. of 7th Extended Semantic Web Conference (ESWC 2010). LNCS, vol. 6088, pp. 1–15. Springer (2010)
4. Barbieri, D.F., Braga, D., Ceri, S., Della Valle, E., Grossniklaus, M.: C-SPARQL: a Continuous Query Language for RDF Data Streams. Int. J. Semantic Computing 4(1), 3–25 (2010)
5. Berstel, B.: Extending the RETE Algorithm for Event Management. In: Proc. of 9th Int. Symp. on Temporal Representation and Reasoning (TIME'02). pp. 49–51. IEEE Computer Society (2002)
6. Chakravarthy, S., Krishnaprasad, V., Anwar, E., Kim, S.K.: Composite Events for Active Databases: Semantics, Contexts and Detection. In: Proc. of 20th Int. Conf. on Very Large Data Bases. pp. 606–617. VLDB '94, Morgan Kaufmann (1994)
7. Doorenbos, R.J.: Production Matching for Large Learning Systems. Ph.D. thesis, Carnegie Mellon University, Pittsbrurg, PA (1995)
8. Forgy, C.L.: Rete: A fast algorithm for the many pattern/many object pattern match problem. Artificial Intelligence 19, 17–37 (1982)
9. Grady, C., Highland, F., Iwaskiw, C., Pfeifer, M.: System and Method For Building a Computer-Based RETE Pattern Matching Network. Tech. rep., IBM Corp., Armonk, N.Y. (1994)
10. Hayes, P.: RDF Semantics. W3C recommendation, W3C (Feb 2004)
11. Sheth, A., Henson, C., Sahoo, S.S.: Semantic Sensor Web. IEEE Internet Computing 12(4), 78–83 (2008)
12. Stuckenschmidt, H., Broekstra, J.: Time-Space Trade-offs in Scaling up RDF Schema Reasoning. In: WISE Workshops. LNCS, vol. 3807, pp. 172–181. Springer (2005)
13. Teodosiu, D., Pollak, G., Souvenirs, A.M., Piaf, E.: Discarding Unused Temporal Information in a Production System. In: Proc. of ISMM Int. Conf. on Information and Knowledge Management. pp. 177–184. CIKM-92 (1992)

14. Walzer, K., Breddin, T., Groch, M.: Relative temporal constraints in the Rete algorithm for complex event detection. In: Proc. of 2nd Int. Conf. on Distributed Event-Based Systems. pp. 147–155. DEBS '08, ACM (2008)
15. Walzer, K., Groch, M., Breddin, T.: Time to the Rescue – Supporting Temporal Reasoning in the Rete Algorithm for Complex Event Processing. In: Proc. of 19th Int. Conf. on Database and Expert Systems Applications. pp. 635–642. DEXA '08, Springer-Verlag (2008)
16. Walzer, K., Schill, A., Löser, A.: Temporal constraints for rule-based event processing. In: Proc. of ACM 1st Ph.D. Workshop in CIKM. pp. 93–100. PIKM '07, ACM (2007)