INSTANS: High-Performance Event Processing with Standard RDF and SPARQL

Mikko Rinne, Esko Nuutila, and Seppo Törmä

Department of Computer Science and Engineering,
Aalto University, School of Science, Finland
`firstname.lastname@aalto.fi`

Abstract. Smart environments require collaboration of multi-platform sensors operated by multiple parties. Proprietary event processing solutions lack interoperation flexibility, leading to overlapping functions that can waste hardware and communication resources. Our goal is to show the applicability of standard RDF and SPARQL – including SPARQL 1.1 Update – for complex event processing tasks. If found feasible, event processing would enjoy the benefits of semantic web technologies: cross-domain interoperability, flexible representation and query capabilities, interrelating disjoint vocabularies, reasoning over event content, and enriching events with linked data. To enable event processing with standard RDF/SPARQL we have created Instans, a high-performance Rete-based platform for continuous execution of interconnected SPARQL queries.

Keywords: Rete, SPARQL, RDF, Complex event processing

1 Introduction

Complex event processing is currently more dominated by proprietary systems and vertical products than open technologies. In the future, however, internet-connected people and things moving between smart spaces in smart cities will create a huge volume of events in a multi-actor, multi-platform environment.

Semantic web technologies enable flexible representation of events in RDF and advanced specification of event patterns with SPARQL. They provide possibilities to reason about event content and to enrich events with linked open data available in the web. Semantic web standards have clear potential to improve the interoperability and offer new capabilities in complex event processing.

A major event processing application can hardly be created out of a single SPARQL query. The INSERT operation in SPARQL 1.1 Update introduced a critical new property: By inserting data into a graph, collaborating SPARQL queries can store intermediate results and communicate with each other. On an environment supporting simultaneous, continuous evaluation of multiple queries, SPARQL can be used to create entire event processing applications [6].

After finding no other platform for incremental processing of multiple SPARQL 1.1 queries, we created Instans. Based on the tried and tested Rete-algorithm [3], Instans shares equivalent parts of queries, caches intermediate matches and provides results immediately, when all the conditions of a query have been matched. In addition to being competitive in SPARQL query processing [1], our studies show qualitative and quantitative benefits compared to SPARQL-based systems using repeated execution of queries over windows on event streams [6].

Here we extend the discussion in [5] by adding further information on the INSTANS implementation of continuous incremental SPARQL query processing.

2 INSTANS Event Processing Platform

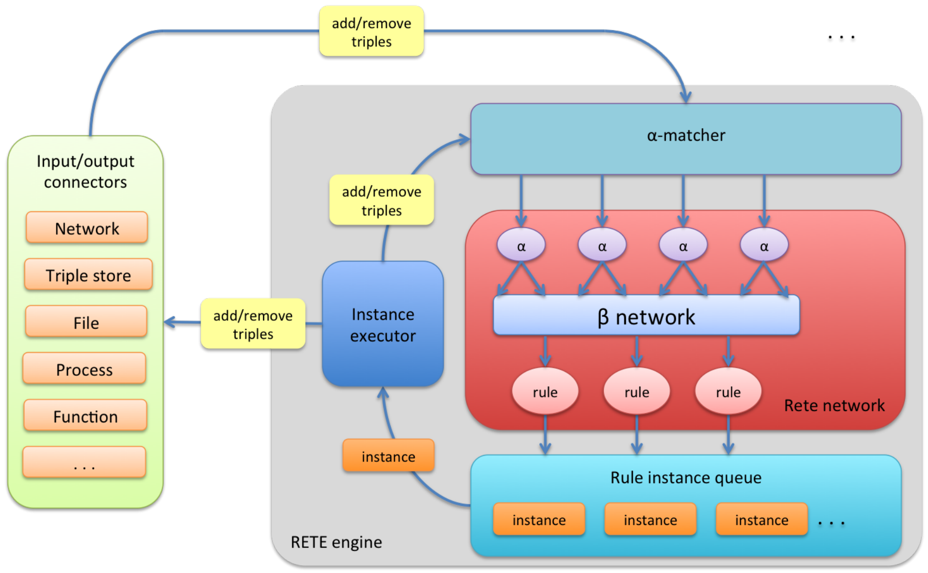


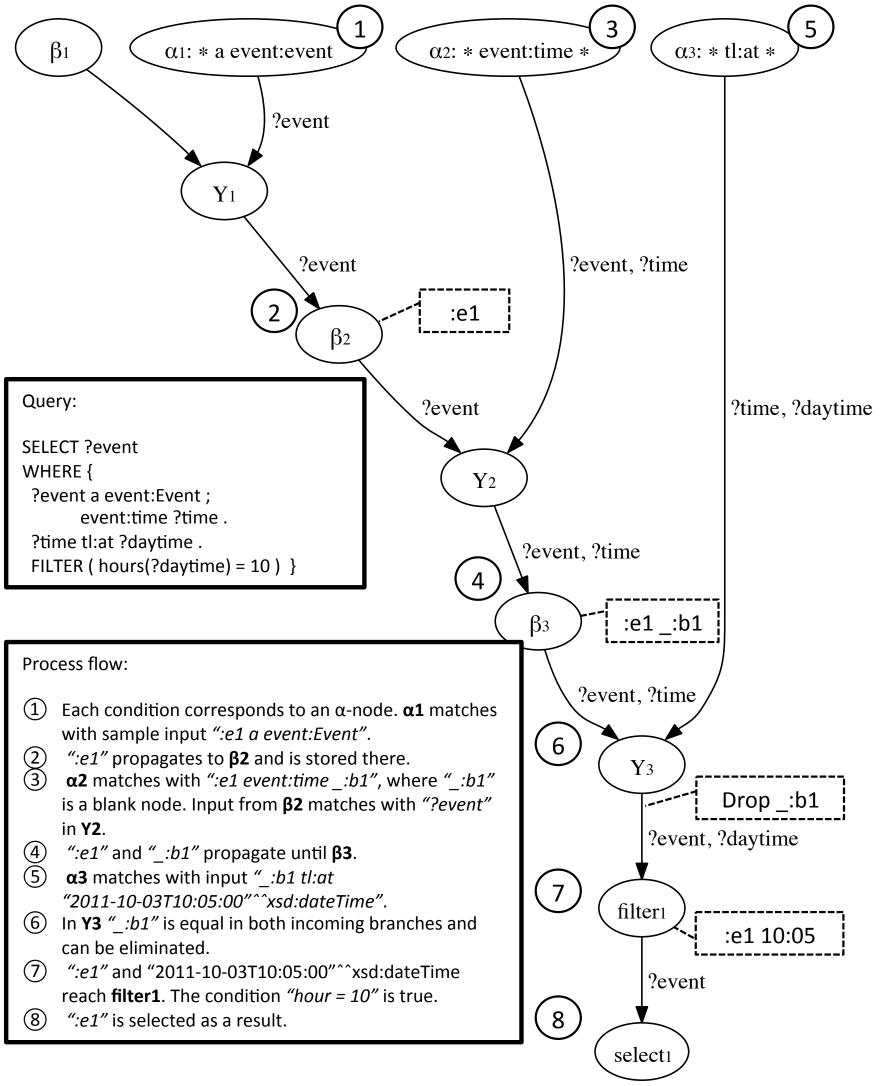
Fig. 1: INSTANS Structure

INSTANS[1] [6] is an incremental engine for near-real-time processing of complex, layered, heterogeneous events. Based on the Rete-algorithm [3], INSTANS performs continuous evaluation of incoming RDF data against multiple SPARQL queries. Intermediate results are stored into a β-node network. When all the conditions of a query are matched, the result is instantly available.

The structure of INSTANS is illustrated in Fig. 1. The system consists of the Rete engine and the input and output connectors, which can interface with the network, triple stores, files or other processes. The Rete engine has four components: 1. Rete network, 2. α-matcher, 3. Rule instance queue, 4. Instance executor. The α-matcher and the Rete network are capable of finding all SPARQL rule conditions satisfying the current set of triples. During runtime the α-matcher receives commands to add and remove triples. The matcher finds the α-nodes of the Rete that match the triples and calls the add or remove methods of those nodes. The changes propagate through the β-network and eventually fully satisfied rule conditions enter the rule nodes, which add new rule instances (with

[1] Incremental eNgine for STANding Sparql, http://cse.aalto.fi/instans/

variable bindings) to the rule instance queue. The instance executor executes the rule instances, which causes add and remove triple commands to be fed into the output connectors. The rule instance execution also feeds add and remove triple commands to the α-matcher, resulting in new rule instances. INSTANS operation over an example query is illustrated in Fig. 2. The query selects events occurring between 10 and 11 am. The asynchronous nature of INSTANS means that all input



Query:

```
SELECT ?event
WHERE {
  ?event a event:Event ;
         event:time ?time .
  ?time tl:at ?daytime .
  FILTER ( hours(?daytime) = 10 )  }
```

Process flow:

① Each condition corresponds to an α-node. **α1** matches with sample input *":e1 a event:Event"*.
② *":e1"* propagates to **β2** and is stored there.
③ **α2** matches with *":e1 event:time _:b1"*, where *"_:b1"* is a blank node. Input from **β2** matches with *"?event"* in **Y2**.
④ *":e1"* and *"_:b1"* propagate until **β3**.
⑤ **α3** matches with input *"_:b1 tl:at "2011-10-03T10:05:00"^^xsd:dateTime"*.
⑥ In **Y3** *"_:b1"* is equal in both incoming branches and can be eliminated.
⑦ *":e1"* and "2011-10-03T10:05:00"^^xsd:dateTime reach **filter1**. The condition *"hour = 10"* is true.
⑧ *":e1"* is selected as a result.

Fig. 2: Example of SPARQL query processing in a Rete-net

is processed when it arrives. To manage periodic actions and missing events, the concept of *timed events* is introduced [7]. When a new timer is started, an *actor* is used to schedule wakeup, at which time a predicate of the timer is changed. A

SPARQL query matching such a triple reacts to the change and carries out the defined actions. No extensions to SPARQL are needed to support timed events.

Performance of INSTANS in terms of notification delay was compared to C-SPARQL [2] using an example application described in [6]. INSTANS yielded average notification delays of 12 ms on a 2.26 GHz Intel Core 2 Duo Mac. In C-SPARQL average query processing delay varied between 12 - 253 ms for window sizes of 5-60 events, respectively, resulting in the window repetition rate being the dominant component of the notification delay for any window repetition rate longer than a second. Using repetition rates of 5-60 seconds with 1 event per second inter-arrival time C-SPARQL notification delay was measured at 1.34-25.90 seconds. Further details are available on the INSTANS project website. Comparison with CQELS [4] is waiting for the availability of a generic version.

3 Conclusions

The feasibility of the central paradigm of INSTANS – continuous incremental matching of multiple SPARQL queries supporting inter-query communication – has so far been supported by empirical tests. When complemented with support for timed events, we have found no showstopper problems which would render the approach unusable for any complex event processing task.

The performance of INSTANS is higher compared to systems based on repeated execution of queries at fixed time intervals (or triple counts); they cannot practically compete with INSTANS whose notification delays are in the order of milliseconds. INSTANS avoids redundant computation: each event is processed immediately on arrival and only once through the Rete network, network structures are shared across similar queries, and intermediate results are memorized.

References

1. Abdullah, H., Rinne, M., Törmä, S., Nuutila, E.: Efficient matching of SPARQL subscriptions using Rete. In: Proceedings of the 27th Symposium On Applied Computing (Mar 2012)
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 - EDBT '10. p. 441. Lausanne, Switzerland (2010)
3. Forgy, C.L.: Rete: A fast algorithm for the many pattern/many object pattern match problem. Artificial Intelligence 19(1), 17–37 (Sep 1982)
4. 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: ISWC'11. pp. 370–388. Springer-Verlag Berlin (Oct 2011)
5. Rinne, M.: DC Short Paper: SPARQL Update for Complex Event Processing. In: ISWC 2012. Springer-Verlag, Boston, MA (2012)
6. Rinne, M., Abdullah, H., Törmä, S., Nuutila, E.: Processing Heterogeneous RDF Events with Standing SPARQL Update Rules. In: Meersman, R., Dillon, T. (eds.) OTM 2012 Conferences, Part II. pp. 793–802. Springer-Verlag (2012)
7. Rinne, M., Törmä, S., Nuutila, E.: SPARQL-Based Applications for RDF-Encoded Sensor Data. In: 5th International Workshop on Semantic Sensor Networks (2012)