Efficient and Expressive Stream Reasoning with Object-Oriented Complex Event Processing

Riccardo Tommasini

Politecnico di Milano
riccardo.tommasini@polimi.it

Abstract. RDF Stream Processing (RSP) engines - systems able to continuously answer queries upon semantically annotated information flows - empirically proved that Stream Reasoning (SR) is feasible. However, existing RSP engines do not investigate the trade-off between the reasoning expressiveness and the performance typical of information flow processing (IFP) systems: either an high throughputs with a low expressiveness (e.g. ρDF) or an high expressiveness (e.g., $\mathcal{EL}$) with a low throughputs are provided. *Can the systematic exploration of this trade-off lead SR to continuously execute expressive reasoning without loosing the efficiency typical of IFP systems?* In this paper, we propose a Systematic Comparative Research Approach (SCRA) to investigate the RSP solution space. Moreover, in contrast with the state-of-the-art trend of adding IFP capabilities to reasoners, we discuss how to realize an Efficient and Expressive Stream Reasoning by adding reasoning capabilities into IFP systems (in particular to Object-Oriented Complex Event Processors).

1 Scene Setting

Stream Reasoning (SR) is a novel research trend that aims at enabling reasoning on rapidly changing information flows [12]. So far, many RDF Stream Processing (RSP) engines - systems able to cope with semantically annotated data flows - were developed as proof-of-concepts [1, 5, 6, 8]. However, due to the complexity of the reasoning task strongly impacts real-time processing, existing solutions either focus on keeping either Information Flow Processing (IFP [11]) comparable performances [16] offering low expressive reasoning (i.e., ρDF [18] with limited extensions) or to optimize expressive reasoning algorithms to the streaming scenario loosing the typical IFP performances [20].

The most of the state-of-the-art solutions pipeline IFP and Semantic Web reasoning modules into *black box* (BB) architectures.

White box (WB) architectural approaches, which redesign all the underlying modules into an integrated solution, can better investigate the performances and reasoning expressiveness trade-off [22]. WB attempts like [1, 7, 16] try to add IFP capabilities to reasoners, while the opposite approach, adding reasoning capabilities to IFP systems, is not attempted yet.

Research Question: *Can the systematic exploration of the performances and reasoning expressiveness trade-off lead SR to continuously execute expressive reasoning without loosing the efficiency typical of IFP systems?*

Exploring the RSP solutions space requires to analyze the RSP engine while is processing. But, due to the complexity of the RSP engine, it might be hard. We need to enable a systematic comparative research approach (SCRA) [9] that simplifies the analysis through a strategy for cross-case studies. Moreover, realizing an efficient and expressing stream reasoning (E_2SR) demands both to rethink the execution semantic model and to expand the solution space with new implementations. Our approach aim at adding reasoning capabilities into IFP systems, in particular Object Orient Complex Event Processor.

Outline - the remainder of this paper is organized as follows: Section 2 summarizes state-of-the-art RSP engines with an IFP background and Section 3 presents a brief overview on RSP Benchmarking. Section 4 presents the proposed research approach. Section 5 describes the approach implementation and the current stage of development. Section 6 shows the evaluation methodology and summarizes the obtained results. Section 7 comes to conclusion presenting the work already done, the lessons learned, and our future directions.

2 RSP Engines State of the Art

Semantic Web (SW) and IFP technologies like Data Stream Management Systems (DSMS) or Complex Event Processing (CEP) played a crucial role to demonstrate that SR is possible. Indeed, they foster the definition of SR requirements [17]: (R.1) Real-Time processing (DSMS); (R.2) pattern-matching on incoming information (CEP); (R.3) Data Integration (SW); (R.4) Rich Ontology Languages (SW). And finally, (R.5) expressive query languages and (R.6) systems scalability (IFP & SW).

Table 1 summarizes and extends a recent survey [17]. It highlights some relevant characteristics of the state-of-the-art RSP engines with a IFP background:

- *Continuous Query Answering* [3] - it is needed to satisfy (R.1);
- *Background Data* - supporting static data access is needed to satisfy (R.3);
- *Time Model* - the system temporal model: one or more timestamps (R.1);
- *Reasoning* - the reasoning expressiveness, when reasoning is available (R.4);
- *Time-Aware* - time-related operators are crucial to satisfy (R.2);
- *Data Transformation* - presence of abstraction functions/aggregates (R.5);
- *Historical Data* - availability of historical data storages (R.3);
- *System Design* - w.r.t IFP or SW: DSMS/CEP/rule-based (R.6);
- *Architectural Approach* - the adopted architectural approach: white box (WB) or a black box (BB) (R.6).

RSP engines like Streaming Knowledge Base [24] and C-SPARQL Engine [5] are examples of in-memory, window-based, RSP engines that adopt a black box approach pipelining a DSMS and a naïve reasoner. They allow continuous query answering on RDF streams or graphs w.r.t background knowledge by the means

System	Cfr.	Cont. Queries	BG Data	Time Model	Reasoning	Time. Aware	Data Trans	Hist. Data	Arch. Appr.	Design
C-SPARQL E.	[5]	✓(p)	✓	TS	RIF*	✓***	✓		BB	DSMS
IMaRS	[4]	✓(p)	✓	TS	Transitive		✓		BB	DSMS
TrOWL	[20]	✓	✓	TS	EL+/SHIQ**				WB	Rules
CQELS	[7]	✓	✓	TS			✓		WB	DSMS
SKB	[24]	✓	✓	TS	OWL sub				BB	DSMS
SparkWave	[16]	✓	✓	TS	RDFS subset		✓		WB	Rules
ETALIS	[1]	✓	✓	2xTS	RDFS subset	✓	✓	✓	BB	CEP
Morph$_{stream}$	[7]	X	✓	TS	ELIO				BB	DSMS

Table 1. State Of The Art of RSP engines related to IFP - p:(periodic) WB: white box BB: black box; 2TS:interval; *:supports Jena Rule-Based Reasoning; **TBox/ABox; *** Subset of Allen Algebra by TS function

of queries expressed with extensions of SPARQL 1.1 (e.g. C-SPARQL) that include the time semantics.

Morph$_{stream}$ [7] ports some reasoning capabilities into existing DSMS system and allows to query virtual RDF streams with SPARQL$_{stream}$. A conjunctive query is translates into the union of multiple conjunctive queries thanks to a reasoner that performs query rewriting and an R2RML mappings extension with time semantics (windows). Queries are executed by an underlying DSMS.

ETALIS [2] engine is a WB solution that processes queries written in ETALIS languages converting them to Prolog rules and executes them on a Prolog engine at run-time. EP-SPARQL [1] is a SPARQL extensions for Event Processing that enables black box Stream Reasoning on ETALIS [2]. EP-SPARQL queries are translated in logic expressions of the ETALIS Language. [1] is the only solution that allows to write complex patterns with time constraints on incoming events, that provides streaming and historical data integration and that is natively time-aware. However, its performance are not satisfying at all [19].

CQELS [8] implements a WB approach porting DSMS concepts (e.g. physical operators, data structures and query executor) into an SPARQL engine with no reasoning capabilities. It can operate queries optimization, because each phase of the processing is available.

SparkWave [16] is a white box ruled-based RSP engine designed for high RDFS performance reasoning over RDF Streams by extending RETE, a reasoning system algorithm, to process incoming information flows. IMaRS [4] optimizes incremental reasoning by relying on a fixed time window to predict expiration times. TrOWL [20] is an engine for efficient incremental ontology maintenance when updates are frequent (but not streaming). It does not rely on fixed time windows to predict the expiration time of streaming information, but it reduces reasoning complexity exploiting syntactic approximation. Despite big performance limitations, TrOWL is still relevant for our research. Indeed, it supports TBox stream reasoning of EL+ and approximate ABox stream reasoning of SHIQ expressiveness.

Notice that ASP-based solutions are out of the scope of this research because their reasoning capabilities and performances are non-comparable with the systems in the solution space we target to investigate.

3 RSP Benchmarking State of the Art

The SR community focuses on RSP engine evaluation. So far, challenges and requirements were formulated [21] and many attempts to address them were developed [14, 19, 25]. Preliminary evaluations on the state-of-the-art confirmed that none of existing RSP engines provides IFP-comparable performances and expressive reasoning at the same time. However, RSP benchmarking still presents some limitations and it is not applied systematically yet.

[25, 14, 19] neither face all the challenges nor satisfy all the requirements proposed in [21]. They provide ontologies, datasets and queries for the evaluation. The metrics set comprises query language coverage, throughput and recently [14] query results mismatch and correctness, but does not consider memory consumption and query execution latency. A minimal testing facility is provided by [19, 14], but without a method to lead the investigation.

The stage of analysis is also limited. Indeed, it consists into an average result after a predefined testing period, while the dynamics of the RSP engine during the entire test is not considered.

RSP benchmarking still misses both an infrastructure to design and test RSP engines performances and a methodology to investigate systematically the trade off. In summary, a SCRA that allows to design and execute comparable, reproducible and repeatable experiments in a controlled environment and, thus, provide a picture of the solution space.

4 Proposed Approach

In Section 3 we stated that both the black box (BB) and white box (WB) state-of-the-art RSP engines show performance limitations [25, 14, 19]. The former cannot perform cross-module optimization, while the latter is realized by adding IFP-capabilities to reasoning systems and, thus, it is not possible to exploit the typical order-based optimization that guarantee IFP performances.

Building on these lessons learned, it would be possible to develop an efficient and expressive SR (E_2SR). Indeed, a common step to all the mature research areas is focusing on improving the systems performances [15]. Two approaches are possible, eliminating the lacks or reinventing the technology pillars. Both require to enable a systematic comparative research approach (SCRA) for RSP engines. Why should the investigation be comparative? SCRA is popular in those research fields where the complexity of the subject goes beyond the possible observable models (e.g. social science). Single-case studies help to deeply understand the subject, but do not foster any generalization. On the other hand, cross-case studies allow general thinking, but with and high final complexity.

Comparing RSP engine dynamics during the entire test execution will clarify how the actual execution semantic of the RSP engine influences the performances. SR needs a strategy to reduce the analysis complexity without losing the relevance of each involved system. SCRA consists into comparing RSP engine dynamics under a given experimental condition.

Enabling SCRA is crucial at this stage of development. A specific investigation methodology is required to contrast the performance measurements, state which solution is better, if any, and possibly drill down or raise up the analysis at different levels. We need to describe normality and stressing conditions for RSP engines, so it is necessary to understand which variables involve into the evaluation. Finally, it would be possible to investigate how the system actually works and to position it in the solution space. To this final extent, we need an infrastructure to design and systematically execute repeatable and reproducible experiments on a given RSP engine under comparable conditions.

The E_2SR goal are the high entailment regime of [20], (i.e. $\mathcal{EL}+$) and the IFP-compatible performances performances of [16, 7]. This requires a WB approach for intra-modules optimization. In Section 2 we presented the limitations of porting IFP-capabilities into reasoners. Moreover, [1] already covers all the featured characteristic that Table 1 highlights, but it is not optimized neither in performance nor for reasoning expressiveness. Thus, we propose to introduce reasoning capabilities in IFP systems, by extending rule-based Object Orient Complex Event Processor engines into a WB approach.

[1] - and in general CEP-based RSP engines - presents many technical opportunities towards E_2SR: (i) event processing languages like Tesla [10] or EPL[1] can perform the reasoning tasks that can be encoded as rules; (ii) they are natively order-aware, more specifically time-aware since data are ordered by recency [13, 22]. (iii) the information flows are usually represented with objects, as in object-oriented languages or databases. Object-Oriented programming languages natively allow some reasoning task which can improve the final entailment regime. (iv) last but not least, CEP systems are usually well-engineered, because the IFP research focused on bandwidth and latency performance optimization as well as scaling by the means of system distribution.

Finally, how to evaluate the obtained results? E_2SR performance evaluation explicitly needs to enable SCRA; SCRA itself demands instead to prove that: (i) the testing infrastructure does not influence the systems results; (ii) a base measurements-set and an investigation method are defined and accepted by the SR community; (iii) some baselines and analysis guidelines are available.

5 Approach Implementation

The first research phase focused on the following research sub-questions:

- SP.1 *Can a test-stand*[2] *enable a SCRA for RSP engines?*

[1] http://bit.ly/1GdhUFC

[2] an aerospace engineering facility to design and execute experiments over engines and to collect performance measurements

– SP.2 *It is possible to implement simple, ruled-based reasoning upon a CEP?*

SP.1) My Master Thesis had the goal to develop Heaven [23], an open source[3] framework that consists into an RSP engine test stand, four naïve implementations of black box DSMS-based RSP engines called baselines and a evaluation methodology. Heaven targets window-based, in-memory RSP engines implemented in Java, like C-SPARQL engine [5] or CQELS [8] or the baselines themselves. The test-stand makes no assumption about the tested RSP engine internal process, treating it as a black box. Thanks to Heaven, it is finally possible to design comparable, repeatable and reproducible experiments by providing: an RSP engine $\mathcal{E}$, an ontology $\mathcal{T}$, a query-set $\mathcal{Q}$ and an dataset $\mathcal{D}$ to stream.

SP.2) Table 2 summarizes the current stage of development. Some E_2RS prototypes were implemented[4] with Esper, an open source CEP engine popular in the SR research field[5].

System	EPL	BG Data	Data Model	Sound/ Complete*
PLAIN	YES	Hashtable	Serialised	Yes
OO-STD	YES	OO	OO-RDF	Yes/No
GENERICS	YES	OO	OO-RDF	Yes/No

Table 2. E_2RS Prototypes * w.r.t ρDF [18]

The E_2RS prototypes development relies on the following assumptions, inspired by [16]: (i) the entailment regime is ρDF [18]; (ii) the initially ontology is small *static* and (iii) its materialization happend in pre-processing.

Plain encodes in EPL the rules for continuous query answer under ρDF entailment regime; events are serialized triples and the TBox is materialized within an hash-map. OO-std proposes a solution where both EPL rules and Java polymorphism are equally exploited to perform typical reasoning tasks of ρDF (i.e., class hierarchy subsumption). Finally, Generics tries to extend OO-std exploiting Java-generics.

6 Empirical Evaluation

SCRA and E_2SR evaluations are related. The former is evaluated by proving the effectivenesses of the testing infrastructure and of the investigation methodology. The latter requires to compare new performance results with the state-of-the-art solutions. Thus, we consider to start with SCRA evaluation, because E_2SR one strictly relies on it.

To demonstrate Heaven effectiveness we run some experiments on the test stand, which results are available[6]. As RSP engine we used some baselines, four window-based RSP engine implementations that are realized pipelining Esper

[3] https://github.com/streamreasoning/heaven

[4] https://github.com/streamreasoning/Proto-EESR

[5] Esper is written in Java and it supports sliding-windows (e.g [5] uses it to implement a black box RSP engine). Moreover, it exploits the EPL query language that has all the characteristics that we need (Section 4).

[6] http://streamreasoning.org/TR/2015/Heaven/iswc2015-appendix.pdf

and Jena ARQ. The baselines offer both RDFS naïve reasoning, materialization of the entire content of the active window at each cycle, and the incremental reasoning, maintaining the materialization over time by updating the differences between two consecutive windows. Our experiments empirically proved that Heaven influences on systems performance are stable and predictable. Thus, we can assert that it enables SCRA for RSP engine.

From the obtained insight, what is already clear is that even when an RSP engines is extremely simple (e.g., one of the baselines), hypothesis verification is hard (e.g. we cannot confirm that the incremental reasoning baselines outperform those with a naïve reasoning approach [20]).

7 Conclusion

The impact of enabling a systematic comparative research approach for RSP engine is potentially high in the Stream Reasoning research field. The initial insights we got by evaluating the baselines with Heaven showed how less we know about the RSP engine dynamics.

SCRA is a priority for all the SR community. Our first future work consist in systematically testing all the supported RSP engines, i.e only in-memory, window-based RSP engines developed in Java. To realize this we need to (i) implement an adapting facade for the RSP engines to test and (ii) define a suite of experiments, which exploits existing RDF streams, ontologies and queries available in the the state-of-the-art of RSP benchmarking [19, 25, 14] to show any aspects of the RSP engine dynamics.

About E_2SR, Esper-based prototypes (Table 2) are promising, but surely far from our goals (i.e. the [20] expressiveness and the [16, 7] performances). E_2SR research priority is defining a standard for event processing query language. E_2SR prototypes exploit a specific one: EPL. The future works should consider: (i) the definition of a minimal fragment of EPL to enable E_2SR; (ii) the prototyping of systems on alternative query languages like Tesla [10]; (iii) the proposal of a execution semantics for SR system built on CEP.

Acknowledgments. Thanks to my advisor Prof. Emanuele Della Valle (Politecnico di Milano) and my co-advisors Daniele Dell'Aglio and Marco Balduini.

References

1. Anicic, D., Fodor, P., Rudolph, S., Stojanovic, N.: EP-SPARQL: a unified language for event processing and stream reasoning. In: WWW 2011. pp. 635–644 (2011)
2. Anicic, D., Rudolph, S., Fodor, P., Stojanovic, N.: Stream reasoning and complex event processing in ETALIS. Semantic Web 3(4), 397–407 (2012)
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., Braga, D., Ceri, S., Della Valle, E., Grossniklaus, M.: Incremental reasoning on streams and rich background knowledge. In: ESWC 2010. pp. 1–15 (2010)

5. Barbieri, D.F., Braga, D., Ceri, S., Della Valle, E., Grossniklaus, M.: Querying RDF streams with C-SPARQL. SIGMOD Record 39(1), 20–26 (2010)
6. Bolles, A., Grawunder, M., Jacobi, J.: Streaming SPARQL - extending SPARQL to process data streams. In: The Semantic Web: Research and Applications, pp. 448–462. Springer Berlin Heidelberg (2008)
7. Calbimonte, J., Corcho, Ó., Gray, A.J.G.: Enabling ontology-based access to streaming data sources. In: The Semantic Web - ISWC 2010. pp. 96–111 (2010)
8. Calbimonte, J., Jeung, H., Corcho, Ó., Aberer, K.: Enabling query technologies for the semantic sensor web. Int. J. Semantic Web Inf. Syst. 8(1), 43–63 (2012)
9. Creswell, J.W.: Research Design: Qualitative, Quantitative, and Mixed Methods Approaches. Sage Publications Ltd., 3 edn. (2008)
10. Cugola, G., Margara, A.: Tesla: A formally defined event specification language. In: ACM International Conference on DEBS. pp. 50–61. ACM (2010)
11. Cugola, G., Margara, A.: Processing flows of information: From data stream to complex event processing. ACM Comput. Surv. 44(3), 15 (2012)
12. Della Valle, E., Ceri, S., van Harmelen, F., Fensel, D.: It's a streaming world! reasoning upon rapidly changing information. IEEE Intelligent Systems (2009)
13. 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)
14. Dell'Aglio, D., Calbimonte, J., Balduini, M., Corcho, Ó., Della Valle, E.: On correctness in RDF stream processor benchmarking. In: The Semantic Web - ISWC 2013. pp. 326–342 (2013)
15. Gray, J. (ed.): The Benchmark Handbook for Database and Transaction Systems (2nd Edition). Morgan Kaufmann (1993)
16. Komazec, S., Cerri, D., Fensel, D.: Sparkwave: continuous schema-enhanced pattern matching over RDF data streams. In: 6th ACM International Conference on Distributed Event-Based Systems, DEBS 2012. pp. 58–68 (2012)
17. Margara, A., Urbani, J., van Harmelen, F., Bal, H.E.: Streaming the web: Reasoning over dynamic data. J. Web Sem. 25, 24–44 (2014)
18. Muoz, S., Prez, J., Gutierrez, C.: Minimal deductive systems for RDF. In: Springer-Verlag. pp. 53–67. ESWC '07, Springer-Verlag, Berlin, Heidelberg (2007)
19. Phuoc, D.L., Dao-Tran, M., Pham, M., Boncz, P.A., Eiter, T., Fink, M.: Linked stream data processing engines: Facts and figures. In: The Semantic Web - ISWC 2012. pp. 300–312 (2012)
20. Ren, Y., Pan, J.Z., Zhao, Y.: Ontological stream reasoning via syntactic approximation. In: Proceedings of the 4th International Workshop on Ontology Dynamics (IWOD 2010). vol. 651. Citeseer (2010)
21. Scharrenbach, T., Urbani, J., Margara, A., Della Valle, E., Bernstein, A.: Seven commandments for benchmarking semantic flow processing systems. In: The Semantic Web - ESWC 2013. pp. 305–319 (2013)
22. Stuckenschmidt, H., Ceri, S., Della Valle, E., van Harmelen, F.: Towards expressive stream reasoning. In: Semantic Challenges in Sensor Networks (2010)
23. Tommasini, R., Della Valle, E., Balduini, M., Dell'Aglio, D.: Heaven test stand: towards comparative research on RSP engines. In: OrdRing 2015-5rd International Workshop on Ordering and Reasoning. p. 6p (2015)
24. Walavalkar, O., Joshi, A., Finin, T., Yesha, Y.: Streaming knowledge bases. In: International Workshop on Scalable Semantic Web Knowledge Base Systems (2008)
25. Zhang, Y., Pham, M., Corcho, Ó., Calbimonte, J.: Srbench: A streaming RDF/SPARQL benchmark. In: The Semantic Web - ISWC 2012. pp. 641–657