Towards a Benchmark for Expressive Stream Reasoning

Riccardo Tommasini, Marco Balduini, Emanuele Della Valle

DEIB, Politecnico of Milano, Milano, Italy
riccardo.tommasini@polimi.it, emanuele.dellavalle@polimi.it,
marco.balduini@polimi.it

Abstract. The stream reasoning community is conducting a good amount of empirical research. It created benchmarks like LSBench, (C)SRBench, CityBench. They fostered the research in RDF Stream Processing (RSP). However, they do not stress much the reasoning task. Indeed, they are limited to RDFS. At the same time, the existing OWL benchmarks do not consider streaming tasks. There is a need to define, design and evaluate a domain-specific benchmark for stream reasoners that go beyond RDFS, namely Expressive Stream Reasoners (ESR). In this paper, we address this need, and we present LASS 1.0, a first attempt to realize a benchmark for ESR. LASS 1.0 comprises an ontology that models influence in social media, a set of reasoning tasks to stress ESR and an instance generation algorithm that creates streaming and static workloads.

1 Introduction

The Stream Reasoning community is conducting a good amount of empirical research to show that it is possible to make sense in real-time of heterogeneous and vast information flows [14]. As in the more mature database community, domain specific benchmarks [8] are emerging as a dominant approach to foster technological progress via fair assessments.

Unfortunately, existing stream reasoning benchmarks [21, 18, 1, 17, 19], focusing on RDF Stream Processing (RSP), only supports continuous query answering under RDFS entailment regime. Indeed, those who conducted research on stream reasoning [2, 3, 12, 15, 20, 22, 23] evaluated their prototypes using traditional (static) OWL benchmarks [4, 10, 13] rather than RSP ones (see Table 1).

We perceive the need for a new stream reasoning benchmark that supports OWL2 stream reasoning tasks, namely Expressive Stream Reasoning (ESR). Therefore, we formulate the research question investigated in this paper as: *How can we define, design and evaluate a domain specific benchmark for ESR?*

In this paper, we answer it by the means of LASS 1.0. This is a first attempt to realize a benchmark for ESR that consists of:

- an OWL 2 RL ontology (named L1O) that extends SIOC[1] with stream reasoning tasks;

[1] https://www.w3.org/Submission/sioc-spec/

		Tested With					
	Entailment	Berlin [4]	LUBM [10]	UOBM [13]	DBpedia	Spire	Galen
streamingKB [23]	RDFS					✓	
EP-SPARQL [2]	RDFS					✓	
TrOWL [20]	EL+						✓
DynamiTE [22]	RDFS		✓				
SparkWave [12]	RDFS	✓					
RDFox [15]	OWL2 RL		✓	✓	✓		

Table 1: Stream reasoners and their entailments vs the evaluation benchmarks.

- a data generation algorithm and its implementation (named L1G); and
- a set of continuous reasoning tasks (named L1C).

The reminder of the paper is organized as follows. Section 2 discusses the design principles. Section 3 details the design process used for LASS 1.0. Section 4 presents the evaluation of LASS 1.0. It shows that L1O adheres to Tom Gruber's principles for ontology design [9]. It shows that L1C supports the design of continuous query using RSPQL syntax. It presents a prototype implementation of L1G algorithm. Moreover, it shows that the whole benchmark is compliant to Jim Grays's critera for benchmark design [8]. Section 5 concludes the paper discussing its limitations and future work.

2 Desing Principles

We analyzed the requirements for stream reasoning benchmarks presented in [18] and the criticism [24] to LUBM benchmark for OWL reasoning. [10, 24]. In the following list, we propose the list of design principles that we elicit. Notably, the connection between each principle and the there papers cited above is traced between brackets using the notation [citation].<pointer in the paper>.

[P.1] *Static TBox of moderate size yet of scalable complexity.* ([18].S7, [10].(3), [24].R3). Assuming static and moderate size TBoxes is a common assumption for SR approaches. For benchmarking purposes, it should be possible to scale the complexity of the TBox by including more axioms and combining them in more complex expressions.

[P.2] *Continuous reasoning tasks.* ([18].S3/4, [24].R4). The benchmark should support reasoning tasks that are meaningful for the steaming domain (i.e., continuous ones) and relevant to stress the stream reasoner.

[P.3] *Arbitrary scaling of static and streaming data.* ([18].S1, [10].(2), [24].R2/8). The benchmark should allow to tune: the size of static ABox, the number of streams and their rates rates.

[P.4] *Usage of continuous queries.* ([18].S2/3/4, [10].(1)). For compatibility with the existing RSP works, the benchmark should embed stream reasoning tasks in continuous queries.

Building on these principles, we now introduce what an ESR benchmark is. We do so building on the definition of RSP benchmark that we introduced in [21].

Definition 1. *An expressive stream reasoning benchmark consist of a set of experiments of the form* $< \mathcal{R}, \mathcal{E}, \mathcal{T}, \mathcal{D}, \mathcal{S}, \mathcal{Q}, \mathcal{K} >$*, where*

$\mathcal{R}$ *is a stream reasoner;*
$\mathcal{E}$ *is an entailment regime to test;*
$\mathcal{T}$ *is a static TBox expressive enough to enable* $\mathcal{E}$*;*
$\mathcal{D}$ *is a static ABox;*
$\mathcal{S}$ *is streaming ABox;*
$\mathcal{Q}$ *is set of continuous queries involving reasoning tasks under* $\mathcal{E}$*;, and;*
$\mathcal{K}$ *is a set of key performance indicators (KPIs) to measure.*

In accordance with **[P.1]**, an *SR Experiment* separates the entailment regime $\mathcal{E}$ from the static TBox $\mathcal{T}$ allowing to scale the domain complexity. Adhering to **[P.2]**, the domain complexity can be scaled independently from the involved reasoning tasks. It separates static and streaming ABoxes definitions, supporting their independent scaling as required by **[P.3]**. As recommended by **[P.4]**, it allows to select a set of queries $\mathcal{Q}$ that stimulates the stream reasoner $\mathcal{R}$ against specific reasoning tasks **[P.2]**. Last but not least, it allows to specify the KPIs set for each experiment, e.g. completeness and soundness with the selected entailment regime, and reactiveness.

3 LASS 1.0

In this section, we describe how we design LASS 1.0. We tell how we chose a domain relevant for stream reasoning (Section 3.1). We discuss how we chose the maximum entailment regime and how we developed a TBox that fully captures the domain complexity (Section 3.2). We present how we wrote a query set that targets continuous reasoning tasks relevant for the domain (Section 3.3). And, we report on the design of an approach to generate ABox streaming and static data according with our TBox (Section 3.4).

3.1 Domain Selection

We selected the Social Media domain for our benchmark because it includes information that is naturally represented as streams, e.g. users continuously generate content. It is well-known to the Semantic Web community that proposed the popular Semantically-Interlinked Online Communities ontology (SIOC) [5]. And, it allows to encode meaningful (stream) reasoning tasks, e.g. users' interactions or user reputation [16].

The validity of this choice is confirmed by the recent adoption of this domain also by the Linked Data Benchmark Council (LDBC) for one of its benchmarks[2].

[2] http://ldbcouncil.org/benchmarks/snbs

3.2 Ontology Engineering

SIOC [5] is a vocabulary to annotate data from social media. Its core module includes 17 classes and 61 properties. Its *Types* module defines user generated contents, and its *Services* module defines the web services interfaces. Considering all the modules, its DL expressiveness is $\mathcal{SHI}(\mathcal{D})$. This entailment includes interesting features, e.g. inverse property.

$$\exists MicroPost \sqcap Post \quad (1)$$
$$\exists contains.\top \sqcap Post \quad (2)$$
$$\exists leads \sqcap has\_influence\_on \quad (3)$$
$$\top \sqcap \forall contains.Tag \quad (4)$$
$$writes \equiv performed\_by^{-} \quad (5)$$
$$is\_sharing \; o \; is\_sharing \equiv is\_sharing \quad (6)$$
$$SingleAuthorPost \equiv Post \wedge = 1 performed\_by.UserAccount \quad (7)$$

Listing. 1: Example of DL axioms showing L1O's language features.

LASS 1.0 Ontology, shortly L1O, is a medium-size ontology that comprises 43 Classes, 24 Object Properties and 7 Datatype Properties[3]. L1O adds to SIOC Core users influence roles, competences and responsibilities, e.g. technical member, discussion leaders, moderators. It adds complex interactions, e.g. shared posts, likes, comments and their popularity within a discussion. It adds post content, e.g. topics and tags. As a result, L1O includes the following DL features: class inclusion (Listing 1.1); role inclusion (Listing 1.3); domain/range restriction (Listing 1.2, Listing 1.4); inverse properties (Listing 1.5); transitive properties (Listing 1.6), and; qualified cardinality restrictions (Listing 1.7).

3.3 Reasoning & Querying

To stimulates reasoning, we considered the following reasoning tasks :

[RT.1] *Continuously monitor the posts that contain a given tag.* "SingleAuthorPost" has several subclasses and requires hierarchical reasoning.
[RT.2] *Continuously monitor the users that are sharing a given post.* The property "is_sharing" is transitive and requires to compute its transitive closure to retrieve all the share posts.
[RT.3] *Continuously monitor the relevant posts.* The Axiom RelevantPost $\sqsubseteq$ $\exists$ contains.TrendingTopic requires realization reasoning.
[RT.4] *Continuously monitor a given user's posts.* The property "performed_by" is the inverse of "writes" and requires to inferred.
[RT.5] *Continuously monitor the participants roles to a given discussion.* The property "participates" has two subproperties and involves hierarchical role reasoning. Moreover, the axiom DiscussionLeader $\sqsubseteq$ $\exists$ leads.Discussion requires realization reasoning.

[3] `http://streamreasoning.org/ontologies/lass.owl`

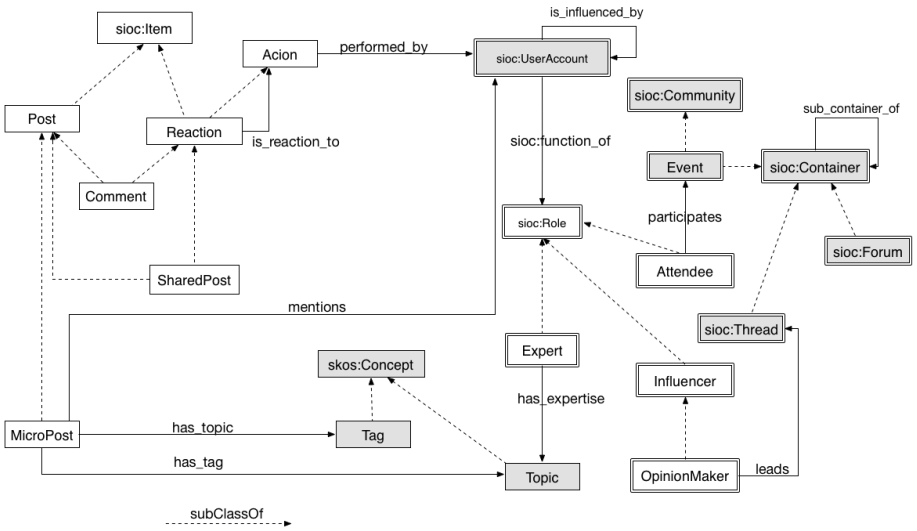

Fig. 2: A sub-portion of LASS 1.0 ontology L1O. The figures highlights with different strokes and background colors the logical modules of L1O.

3.4 Data Generation

To generate instances, we need to distinguish between static ABox $\mathcal{D}$ and streaming ABox $\mathcal{S}$. The former scales in size, while the latter scales in numbers of streams and rates of change. $\mathcal{S}$ instance generation relies on ordering relations that are naturally present in data and captured in the TBox. In L1O, we distinguish four logical modules that we highlighted in Figure 2:

- The *Community* module is identified with a double stroke with gray background. It models users and groups generating content. SIOC Community and UserAccount are the module's root; we included classes as Social Network, e.g. Facebook or Twitter; Argumentative Discussion, e.g. users posting about a Topic; Event, e.g. a concert where users are.
- The *Influence* module is identified with a double stroke with white background. It models users' influence roles within a community. SIOC Role is the module's root; we included with classes as Expert, i.e. someone who knows about a Topic; Influencer, i.e. someone who can spread opinions.
- The *Content* module is identified with a single stroke with gray background. It models what characterizes posts and discussions. SKOS Concept is the module's root that we extended it with Topic, e.g. a TV-Show or a football team; Tag, i.e. a label attached to a post that relates it to other posts or to an event.
- The *Action* module is identified with a single stroke with white background. It models users' interaction within a community. SIOC Item and Post are the module's roots; we included Action, i.e. everything that a user can perform

and Reaction, e.g. comment and likes; MicroPost, e.g. tweets; RelevantPost characterizes the relevance w.r.t. the presence of some Topic.

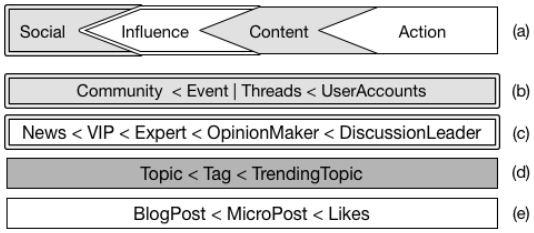

Fig. 3: Class changing rates to generate L1O instances. $A < B$= A slower than B.

L1O's modules have different velocities, i.e. class instances change at different rates. Properties link instances across and within modules, but do not influence directly the ordering. Figure 3 visualize the different velocity between the instances of L1O's classes. Communities and User Accounts change daily, i.e., less frequently than influence roles that change hourly. Content, i.e. topics, tags, pictures change faster (in minutes), but less than user actions, e.g posts, likes, shared-posts, etc., which change in seconds. Within class modules, further differences in changing rates can be found, e.g. a VIP role changes less frequently than a discussion leader or Blog Posts are slower than tweets. Notably, this ordering relationships are specific for L1O, but similar ones are present in SR domains.

L1G is a data generation algorithm that exploits these ordering relations to instantiates individuals. It starts from the "slowest" classes that in L1O belong to the *Community* Module: Discussion, Event, Thread and UserAccount (Figure 3.b). Then, it continues with "faster" classes from the *Content* Module like Topic, Tag and TrendingTopic (Figure 3.c). Finally, it generates the "fastest" classes, e.g. Post, Like, SharedPost, that belong to the *Actions* Module (Figure 3.e).

L1G does not generate classes from the Influence Module, but it ensures that they can be deducted. For instance, it randomly assigns a *leads* relation that triggers the inference of lass:DiscussionLeader (Figure 3.d). L1G does not assign a time.stamp to instances, but maintains the ordering relations[4]. Thus, timestamps can be assigned post-hoc according to flow rates that better represent the domain of choice [21].

4 Evaluation

In this section, we report on our experience in evaluating an ESR benchmark. To this extent, we evaluate LASS 1.0 soundness. First, we evaluate L1O and L1C independently. Then, we evaluate LASS 1.0 as a whole showing that it complies to Jim Gray's principles for domain-specific benchmarks. We leave L1G's evaluation to future empirical studies that involve LASS 1.0.

[4] For instance, Posts are generated first then a user is selected as author. Tags, Topics and Mentions are assigned to each Post just after. Reactions are assigned in the end.

4.1 L1O

To evaluate L1O, we consider Tom Gruber's principles for ontology design [9]. According with Gruber, an ontology should effectively communicate the intended meaning (*Clarity*). It should be formally consistent to its specifications (*Coherence*). It should welcome the definition of new terms for special uses (*Extendibility*). It should be specified at knowledge-level, without depending on the symbolic representation (*Minimal Encoding Bias*). It should make as few claims as possible to capture the intended knowledge (*Minimal Ontological Commitment*). To evaluate L1O, we interpret these principles as follow:

- *Clarity* is satisfied, all properties and classes are paired with a natural language descriptions and aligned with existing and well-known ontologies and vocabularies (e.g. SIOC).
- *Coherency* is satisfied, since no inconsistency or meaningless information is inferred in the ABox w.r.t. all TBox inferences.
- *Extendibility* is satisfied, since L1O is based on SIOC and it does not contrasts with SIOC modules.
- *Minimal Encoding Bias* is satisfied because L1O is released using W3C standards (e.g. OWL2), and, thus, it uses only the allowed axioms.
- *Minimal Ontological Commitment* is not satisfied intentionally, since L1O is designed for benchmarking purposes and, thus, it intentionally includes expressive language constructs that stress the stream reasoner.

4.2 L1C

To evaluate L1C, we consider its adherence to the practice of micro-benchmarking that in the SPARQL benchmarking context has been already applied through the idea of choke points [7]. A choke point is an aspect of the query execution which is known to be problematical for the present generation of engines [7]. In accordance with micro-benchmarking practice L1C isolates stream reasoners choke points as reasoning tasks. We can use them to design continuous queries that stress engines. Listing 1.1 presents an example using RSP-QL syntax that captures RT 2 and RT 3 reasoning tasks.

```
PREFIX lass: <streamreasoning.org/ontologies/lass.owl>
SELECT ?u ?p
FROM NAMED WINDOW :win1 ON :facebook [RANGE 30 minutes STEP 30 minute]
FROM NAMED WINDOW :win2 ON :twitter  [RANGE 30 minutes STEP 30 minute]
WHERE {
        WINDOW :win1 {
        ?p ; lass:performed_by ?u .
        ?u lass:SingleAuthorPost;
           lass:isSharing ?p1 .
           }
        WINDOW :win2 {
        ?p1 a lass:PopularPost;
            lass:performed_by <@realDonaldTrump> .
           }}
```

Listing 1.1: Example RSP-QL continuous query that captures RT 2 and RT 3.

4.3 LASS 1.0

. The Linked Data Benchmarking Council[5] (LDBC) recently adopted criteria for good benchmark design from the Database community [8, 11] to evaluate Linked Data benchmarks. Jim Gray's design principles [8] are:

[G.1] *Simplicity.* A benchmark must be understandable.
[G.2] *Portability.* It must be applied to many different systems.
[G.3] *Scalability.* It must scale up to small or larger systems.
[G.4] *Relevance.* It must measure performance/price of typical system tasks according with a cost model.

Moreover, we consider Karl Huppler's *Verifiability* principle:

[H.1] a benchmark must represent the system performance in a formally verifiable way [11][6].

LASS 1.0 satisfies **[G.1]** because it is set to a natively streaming domain that is known to the SR community. It satisfies **[G.3]** because in its experiment definition it allows to variate both data and expressiveness. It satisfies **[G.2]** because we used standard W3C semantic technologies to develop it. It satisfies [**H.1**] because of the formal experiment definition that is compliant to the experimental environment defined in [21] and results into formal reports that can be automatically compared.

5 Discussion & Conclusion

In this paper, we investigate how to define (Section 2), design (Section 3) and evaluate (Section 4) a domain specific benchmark for expressive stream reasoning. We presented LASS 1.0, a first attempts to realize a benchmark for expressive stream reasoners, that contributes to the state of the art with:

L1O an ontology for the social media domain of expressiveness $\mathcal{SRIQ}(\mathcal{D})$;
L1C a set of continuous reasoning tasks to stress the stream reasoner; and
L1G a instance generator for L1O that exploits the implicit ordering relation between the classes that is given by their different variance over time.

LASS 1.0's domain of application is adequate for stream reasoning. It contains complex concepts that demand for expressive reasoning, comparable to traditional OWL benchmark. It involves information flows that naturally provide streaming workloads (Table 2). It is scalable and verifiable thanks to an experiment-driven methodology inspired by [21].

The main limitation of this work is that it lacks of an empirical study that proves LASS 1.0 effectiveness. This choice is motivated by the absence of a

[5] `http://ldbcouncil.org`
[6] As argued in [11], for a proper evaluation it is not necessary to be compliant to all the principles, but only to those that reflects the benchmark purpose.

Benchmark	Domain	W3C	Data Scalability	Entailment	Streaming	Verifiability
CityBench [1]	IoT	✓	ds	rdfs6	✓	Correctness
BSBM [4]	e-Commerce	✓	✓^{g}	RDFS		✓
CSRBench [6]	IoT	✓	ds	rdfs6	✓	Oracle
LUBM [10]	Academic	✓	✓^{g}	OWL Lite		✓
UOBM [13]	Academic	✓	✓^{g}	OWL DL		✓
LSBench [17]	SN & IoT	✓	✓^{g}	None	✓	
SRBench [19]	IoT	✓	✓^{g}	rdfs6	✓	
Spire[7]	Botanic	✓	✓^{g}	RDFS		✓
Galen[8]	Medical	✓	ds	EL++		✓
LASS 1.0	Social media	✓	✓^{g}	$\mathcal{SRIQ}(\mathcal{D})$	✓	Definition 2

Table 2: Benchmarks used for stream reasoner evaluation. IoT= InternetOfThings; SN=SocialMedia; ✓^{g}=generator; ds=dataset; rdfs6=SubClassOf.

method to ensure comparability for stream reasoners evaluation. In [21] authors investigate it for RSP engines, but the question is still open when reasoning is involved. Moreover, we consider the benchmark as a scientific effort that aims at positioning an approach in the solution space. Inevitably, the benchmark influences the evaluation, but this bias can be predictable if the logical path that sustains the benchmark specification is clear. However, we understand its value and we consider it an immediate future work.

A second limitation of this work, is that it does not discuss in detail the metrics that the benchmark should consider. Key performance indicators (KPIs) selection has been discussed in the state of the art [1, 17, 19]. We consider soundness, completeness and throughput as relevant KPIs for expressive stream reasoning benchmarking. However, a broader study has still to be conducted.

Future Work. As previously mentioned, we are planning to empirically test LASS 1.0 to evaluate existing stream reasoners. However, this requires to investigate the problem of correctness. Moreover, we aim at introducing more language features in LASS 1.0, defining modules that cover different OWL 2 profiles. Last but, not least, we plan to explore further reasoning types, e.g. Event Calculus or temporal reasoning.

References

1. Ali, M.I., Gao, F., Mileo, A.: Citybench: A configurable benchmark to evaluate RSP engines using smart city datasets. In: The Semantic Web - ISWC 2015 - 14th ISWC, Bethlehem, PA, USA, October 11-15, 2015, Proceedings, Part II. pp. 374–389
2. Anicic, D., Fodor, P., Rudolph, S., Stojanovic, N.: EP-SPARQL: a unified language for event processing and stream reasoning. In: WWW. pp. 635–644
3. Barbieri, D.F., Braga, D., Ceri, S., Della Valle, E., Grossniklaus, M.: Incremental reasoning on streams and rich background knowledge. In: The Semantic Web: Research and Applications, 7th Extended Semantic Web Conference, ESWC 2010, Heraklion, Crete, Greece, May 30 - June 3, 2010, Proceedings, Part I. pp. 1–15
4. Bizer, C., Schultz, A.: The berlin SPARQL benchmark. Int. J. Semantic Web Inf. Syst. 5(2), 1–24

5. Bojars, U., Breslin, J.G., Finn, A., Decker, S.: Using the semantic web for linking and reusing data across web 2.0 communities. J. Web Sem. 6(1), 21–28
6. 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 - 12th ISWC, Sydney, NSW, Australia, October 21-25, 2013, Proceedings, Part II. pp. 326–342
7. Erling, O., Averbuch, A., Larriba-Pey, J., Chafi, H., Gubichev, A., Prat-Pérez, A., Pham, M., Boncz, P.A.: The LDBC social network benchmark: Interactive workload. In: Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, Melbourne, Victoria, Australia, May 31 - June 4, 2015. pp. 619–630
8. Gray, J. (ed.): The Benchmark Handbook for Database and Transaction Systems (2nd Edition). Morgan Kaufmann
9. Gruber, T.R.: Toward principles for the design of ontologies used for knowledge sharing? Int. J. Hum.-Comput. Stud. 43(5-6), 907–928
10. Guo, Y., Pan, Z., Heflin, J.: the: A benchmark for OWL knowledge base systems. J. Web Sem. 3(2-3), 158–182
11. Huppler, K.: The art of building a good benchmark. In: Performance Evaluation and Benchmarking, First TPC Technology Conference, TPCTC 2009, Lyon, France, August 24-28, 2009, Revised Selected Papers. pp. 18–30
12. Komazec, S., Cerri, D., Fensel, D.: Sparkwave: continuous schema-enhanced pattern matching over RDF data streams. In: Proceedings of the Sixth ACM International Conference on Distributed Event-Based Systems, DEBS 2012, Berlin, Germany, July 16-20, 2012. pp. 58–68
13. Ma, L., Yang, Y., Qiu, Z., Xie, G.T., Pan, Y., Liu, S.: Towards a complete OWL ontology benchmark. In: The Semantic Web: Research and Applications, 3rd European Semantic Web Conference, ESWC 2006, Budva, Montenegro, June 11-14, 2006, Proceedings. pp. 125–139
14. Margara, A., Urbani, J., van Harmelen, F., Bal, H.E.: Streaming the web: Reasoning over dynamic data. J. Web Sem. 25, 24–44
15. Nenov, Y., Piro, R., Motik, B., Horrocks, I., Wu, Z., Banerjee, J.: Rdfox: A highly-scalable RDF store. In: The Semantic Web - ISWC 2015 - 14th ISWC, Bethlehem, PA, USA, October 11-15, 2015, Proceedings, Part II. pp. 3–20
16. Ouirdi, M.E., El Ouirdi, A., Segers, J., Henderickx, E.: Social media conceptualization and taxonomy: A lasswellian framework. Journal of Creative Communications 9(2), 107–126
17. 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 - 11th ISWC, Boston, MA, USA, November 11-15, 2012, Proceedings, Part II. pp. 300–312
18. Scharrenbach, T., Urbani, J., Margara, A., Della Valle, E., Bernstein, A.: Seven commandments for benchmarking semantic flow processing systems. In: The Semantic Web: Semantics and Big Data, 10th International Conference, ESWC 2013, Montpellier, France, May 26-30, 2013. Proceedings. pp. 305–319
19. Stupar, A., Michel, S.: Srbench-a benchmark for soundtrack recommendation systems. In: 22nd ACM International Conference on Information and Knowledge Management, CIKM'13, San Francisco, CA, USA, October 27 - November 1, 2013. pp. 2285–2290
20. Thomas, E., Pan, J.Z., Ren, Y.: Trowl: Tractable OWL 2 reasoning infrastructure. In: The Semantic Web: Research and Applications, 7th Extended Semantic

Web Conference, ESWC 2010, Heraklion, Crete, Greece, May 30 - June 3, 2010, Proceedings, Part II. pp. 431–435
21. Tommasini, R., Della Valle, E., Balduini, M., Dell'Aglio, D.: Heaven: a framework for systematic comparative research approach for rsp engines. In: 13th Extended Semantic Web Conference, ESWC 2016, Heraklion, Crete, Greece. pp. 87–92
22. Urbani, J., Margara, A., Jacobs, C.J.H., van Harmelen, F., Bal, H.E.: Dynamite: Parallel materialization of dynamic RDF data. In: The Semantic Web - ISWC 2013 - 12th ISWC, Sydney, NSW, Australia, October 21-25, 2013, Proceedings, Part I. pp. 657–672
23. Walavalkar, O., Joshi, A., Finin, T., Yesha, Y.: Streaming knowledge bases. In: International Workshop on Scalable Semantic Web Knowledge Base Systems (2008)
24. Weithöner, T., Liebig, T., Luther, M., Böhm, S.: Whats wrong with owl benchmarks. Citeseer