# Taming velocity and variety simultaneously in big data with stream reasoning tutorial

Della Valle, Emanuele ; Dell' Aglio, Daniele ; Margara, Alessandro

DOI: https://doi.org/10.1145/2933267.2933539

# Tutorial: Taming Velocity and Variety Simultaneously in Big Data with Stream Reasoning

Emanuele Della Valle, Daniele Dell'Aglio, Alessandro Margara
DEIB – Politecnico di Milano
[emanuele.dellavalle, daniele.dellaglio, alessandro.margara]@polimi.it

## ABSTRACT

Many "big data" applications must tame velocity (processing data *in-motion*) and variety (processing many different types of data) simultaneously.

The research on knowledge representation and reasoning has focused on the variety of data, devising data representation and processing techniques that promote integration and *reasoning* on available data to extract implicit information. On the other hand, the event and stream processing community has focused on the velocity of data, producing systems that efficiently operate on streams of data on-the-fly according to pre-deployed processing rules or queries. Several recent works explore the synergy between stream processing and reasoning to fully capture the requirements of modern data intensive applications, thus giving birth to the research domain of *stream reasoning*.

This tutorial paper offers an overview of the theoretical and technological achievements in stream reasoning, highlighting the key benefits and limitations of existing approaches, and discussing the open challenges and the opportunities for future research. The paper mainly targets researchers and practitioners in the area of event and stream processing. The paper aims to stimulate the discussion on stream reasoning and to further promote the integration of reasoning techniques within event and stream processing systems in three ways: (i) by presenting an active research domain, where researchers on event and stream processing can apply their expertise; (ii) by discussing techniques and technologies that can help advancing the state of the art in event and stream processing; (iii) by identifying the open problems in the field of stream reasoning, and drawing attention to promising research directions.

## 1. INTRODUCTION

Many "big data" applications must tame velocity (processing data *in-motion* in real-time or near real-time) and variety (processing many different types of data) simultaneously. The goal is to timely provide access to implicit and explicit knowledge that can be extracted from the data in-motion, possibly joined with static data *at-rest*. Examples come from many scenarios, like the Internet of Things, social media analytics and smart cities.

The researchers on knowledge representation and reasoning, in particular the fields of Semantic Web [57] and Ontology-based Data Integration [43], focused on the heterogeneity of data (the *variety* aspect of Big Data). They devised data models (RDF [24]), query languages (SPARQL [34]), modelling languages (OWL [36]), and methodologies (Ontology Based Data Access (OBDA) [20]) that ease data integration and enable access to (potentially implicit) knowledge. Current solutions scale on the size of data, but assume changes to occur at low frequencies, and this clashes with the requirement of real-time processing.

On the other hand, the research on event and stream processing has mainly focused on the *velocity* of data, producing software systems that efficiently operate on streams of data on the fly according to some pre-deployed processing rules or queries [23]. This led to the development of various data stream processing systems [9] and Complex Event Processing (CEP) systems [46, 33] that effectively deal with the transient nature of data streams, providing low delay processing even in the presence of large volumes of input data generated at a high rate.

All these systems are based on data models, like for example the well known relational model, which allow the implementation of ad-hoc optimizations to improve the processing. However, these models also limit the processing to a predefined set of operations on streams with a fixed structure.

A number of recent works explore the synergy between stream processing and reasoning to capture both the real-time requirements of modern applications and the heterogeneity of the data they consider. This gave birth to the research field of *stream reasoning* [27]. After a few years of research with interesting investigations in the field [47], we believe that the full potential of the stream reasoning research still remains vastly unexplored.

This paper offers a detailed presentation of the theoretical and technological achievements in stream reasoning, highlighting the key ideas and benefits of existing approaches, and discussing the open issues and limitations.

The paper aims to draw the attention of the experts on event and stream processing to the field of stream reasoning, with the goal of promoting further advancements in the area.

In particular, the paper provides the following contributions: (1) An overview of event and stream processing systems; (2) An introduction to the Semantic Web and to the technologies for reasoning on static data; (3) An overview of some use cases and scenarios that motivate and guide the research on stream reasoning; (4) A detailed presentation of the state of the art techniques and tools for stream reasoning; (5) A critical discussion of the strengths and limitations of current approaches and tools, with focus on the open problems and potential research directions.

We believe that the content of this tutorial paper is relevant for researchers on event and stream processing for two reasons: first, the paper provides an overview of an active research domain where the researchers can apply their expertise; second, the tutorial presents reasoning techniques and technologies that they can use to advance the research on event and stream processing.

The paper is organized as follows. Section 2 presents background information on data stream and event processing, and on Semantic Web technologies and reasoning. Section 3 discusses some use cases that motivate the need for reasoning on streams of dynamic data. Section 4 presents the state of the art approaches for stream rea-

soning, highlighting their focus, their benefits, and their limitations. Section 5 presents a critical discussion of the of the open problems and presents potential research directions. Finally, Section 6 concludes the paper.

# 2. BACKGROUND

This section presents backgound information on the technologies that mostly influenced the development of stream reasoning. In particular, Stream reasoning builds on the results of two main research areas: data and event stream processing (Section 2.1) and reasoning (Section 2.2).

## 2.1 Data and Event Stream Processing

The research on data and event stream processing aims to analyze streams of data on the fly to timely produce new results or detect situations of interest based on a set of pre-deployed rules or queries [23].

Solutions for event and stream processing typically target two main requirements: high throughput, to manage large volumes of input data, and low latency, to provide new results with minimum delay. In other words, event and stream processing systems address the *velocity* dimension of Big Data.

Two main models and approaches for stream processing have emerged, namely data stream processing and complex event processing.

### *2.1.1 Data Stream Processing*

Data stream processing systems [9] provide processing abstractions to *transform* one or more input data streams into one or more output data streams.

A classical processing model adopted in many data stream processing systems is the CQL model [6], first defined in the Stanford STREAM system [5]. CQL defines the processing tasks in terms of the well known relational operators, and introduces additional primitives to deal with the dynamic nature of the input data. CQL comprises stream-to-relation primitives (windows) to isolate relevant portions of each stream, traditional relational operators to operate on the content of each window, and relation-to-stream operator to convert back the results of the processing into a data stream. For instance, a window can be used to isolate the last 10 minutes in two incoming stream, a join operator can be used within the window to merge the two streams, and a relation-to-stream operators can stream only newly generated data.

Recent proposals substitute the relational model with functional programming abstractions to process streaming data. This approach has received great attention with the advent of several open source solutions designed for cluster environment, such as Spark Streaming [71] and Flink [2].

Other stream processing systems specify the processing task as a graph of operators [1]. Each operator consumes the input streams and produces one or more output streams for other operators. Operators can be either standard or custom, that is to say, defined programmatically by the developer. Recently, this model has been adopted to build scalable solutions that process large volumes of data in cluster environment. For instance, Storm and Heron [39] have been used as part of the Twitter infrastructure.

### *2.1.2 Complex Event Processing*

Complex Event Processing (CEP) systems [46, 33] consider data elements inside streams as timestamped event notifications, and provide processing abstractions to capture patterns of interest in the input event streams. Patterns predicate on the content and timing relations among events.

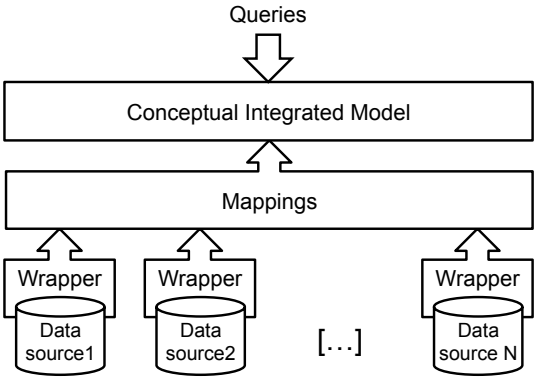


**Figure 1: Data Integration Systems.**

Complex Event Processing systems differ in the way they represent time: some systems provide point semantics that assumes each event to occur at a precise point in time, while others use interval semantics that assumes each event to be valid in a time interval [70].

Different systems also offer different tradeoffs between expressivity and efficiency. Some systems adopt simple patterns that can be translated into automata for efficient processing [18], while others adopt more complex patterns, for instance based on logics [22, 3].

## 2.2 Reasoning for Data Integration

The problem of data integration has been studied for decades. In separate data sources the same type of information can appear with different syntaxes, data structures and conceptual models, raising the data variety problem at syntactical, structural and semantic level.

Most data integration systems adopt the architecture outlined in Figure 1. A *Conceptual Integrated Model* (CIM) offers a vocabulary of terms to use when issuing queries on the data sources as if they were a single integrated database. *Wrappers* hide the differences among data sources and logically expose them in a single data model. *Mappings* couple[1] the terms in the CIM to those used in the wrappers.

In the '90s, the relational database schema languages were investigated as modelling languages for the CIM. Nowadays, all major database vendors offer them to tame variety at syntactic and structural level. Starting from the 2000's, Description Logics [8] were studied as modelling languages for the CIM, and Ontology Based Data Access (OBDA) became an accepted method for taming variety at semantic level (see [20] for a recent special issue on the topic). In OBDA systems, when a query is issued against the CIM, a reasoner rewrites it to a query that can be mapped into queries to the underlying data source and wrapped into the query languages locally used. The answers to those wrapped queries can be integrated to provide the answer. Some reasoning can be run also on the results of the rewritten query before returning the answers to the users. Recent works shown that DL-Lite [7] is able to express queries to be rewritten over relational databases and directly encoded in SQL, proving that the complexity of the conjunctive query answering task is $AC^0$.

Nowadays, when building an OBDA system, we can count on Semantic Web standards. Resource Description Format (RDF) [24] can be used as the logical data model to represent information in the wrapped data sources. Ontology Web Language (OWL) [36] can be use as ontological language for modelling the CIM. R2RML [25] can be used as mapping language to describe how to map a OWL

[1]Mappings can assume multiple forms, i.e. Local-as-View, Global-as-View and both [43]

**Table 1: How stream processing (SP), Complex Event Processing (CEP) and reasoning (OBDA) tame Big Data dimensions**

| Dimensions | SP/CEP | OBDA |
|---|---|---|
| Data Volume | ✓ | ✓ |
| Data Velocity | ✓ | ✗ |
| Data Variety | ✗ | ✓ |
| Data Veracity | ✓ | ✗ |

ontology into a relational data schema. Dialects of R2RML exist to map between RDF and tree- and graph-based data models. Finally, SPARQL [34] can be used as query language for RDF.

# 3. THE NEED FOR STREAM REASONING

Several modern data intensive applications need to deal with large volumes of heterogeneous and dynamic data. Such applications need to cope with the *volume* of input data, with the *variety* of data and sources that expose it; and with the *velocity*, as data remains valid for a limited amount of time and needs to be processed as soon as possible to produce relevant results.

An example is represented by the domain of smart cities that aims to process and understand the information relevant for the life of a city and use it to make the city run better, faster, and cheaper [41, 62].

Smart grids [68] represent another scenario that requires data monitoring and integration, situation detection, and (partially or completely) automated decision making. The goal of smart grids is to make current energy grids more efficient and sustainable by collecting and interpreting information coming from different stake holders, such as energy producers, grid operators, or appliance manufacturers.

Finally, semantic analysis of social media [32] extends classic connection analysis by enriching the relations between people and concepts with semantic annotations. One of the goals of the analysis of social media is to capture hidden relations between people and concepts. In this scenario, it is interesting to detect not only the current situation or context, but also the historical evolution of relations over time.

These scenarios pose some challenging requirements that cannot be easily satisfied with the classic solutions for data stream and complex event processing and with reasoning engines for static data, as presented in Section 2.

First, the above examples need to cope with large volumes of data. For instance, in the context of smart cities, the sensors in the city of Dublin currently produce every day about four to six GB of data about the public transport [41], and in the future more sensors will be deployed which will produce more complex data (e.g., HD cameras). In the context of social media analysis, Facebook, at the end of 2015, had 1.59 billion monthly active users.

Second, the scenarios require to tame velocity, i.e., on the fly processing of data streams. On the one hand, the volume of data is too large to be stored before processing. On the other hand, applications demand for new results with small delay, to enable informed decision making. For instance, Facebook users produce on average 4.5 billions "like" daily and Twitter produces more than 7000 tweets per second.

Third, all scenarios aim to derive high level knowledge from low level information. The presence of a suitable processing model to express the processing tasks is one of the main challenges and needs.

Forth, producing valuable results require the integration of heterogeneous datasets, both static and dynamic (taming the variety dimension of Big Data). For instance, data produced by different social media have different data formats. In smart cities, the number and types of deployed sensors continuously increases, and each of them produces data with different content and format.

Finally, data can be incomplete or noisy. This is known in Big Data as the veracity dimension. For instance, the sensors deployed in a city can experience malfunctioning or provide incorrect or inaccurate results. Thus, the processing model should be able to verify the consistency of data and limit the production of incorrect results.

# 4. THE STATE OF THE ART

This section reviews the state of the art approaches to stream reasoning. First, we illustrate the intuition that makes stream reasoning feasible in Section 4.1. Then, given the heterogeneous nature of the proposals in the field, we organize the remainder of the section in four parts. Section 4.2 presents solution to apply event and stream processing techniques to streams of RDF data. Section 4.3 presents approaches that explicitly target the reasoning process in presence of streaming data. Section 4.4 presents approaches that define formal models to process, query and reason over streams of RDF data. Section 4.5 presents approaches that deal with the veracity dimension of data streams.

## 4.1 The intuition

A fundamental problem of stream reasoning is the fact that many relevant reasoning methods, for example for description logics, are not able to deal with high frequency data streams. While they try to derive entailments of the goal predicate, newly incoming data will pile up. However, a trade-off exists between the complexity of the reasoning method and the frequency of the data stream the reasoner is able to handle.

The intuition [61] to solve this problem is straightforward. It stems from the observation of a similar trade-off between memory size and access time in computer systems, which is solved using a memory hierarchy. stream reasoning can be optimised to provide reactive answers by using a hierarchy of processing steps of increasing complexity. Figure 2 illustrates this idea of *cascading stream reasoners* for processing streaming data. Technically, this intuition is supported by the possibility to push processing steps down in the hierarchy to speed up reasoning and the possibility to complete the reasoning process at each layer by only processing the results coming up from the layer underneath.

The lower levels are designed to cope with the volume and the velocity of streaming data. Those layers plays two roles: they logically wrap the raw data stream into an adequate data model (as we will explain below, RDF Stream) and they provide the possibility to query those RDF streams using a continuous extension of SPARQL query language under OWL2QL entailment regime applying the OBDA methods. Only those parts of the raw stream that match the registered queries are passed on to the higher levels, where they arrive with a lower volume/frequency. On the next higher level, relatively simple but efficient reasoning methods, e.g., OWL2RL based reasoning, can be used to further process the result stream. Only at the top of the hierarchy where the frequency of change has been reduced significantly, we can expect to be able to use expressive reasoners. Following this intuition, only inferences that cannot be carried out on the lower layers of the hierarchy are actually carried out using more expressive reasoning methods.

## 4.2 RDF Stream Processing

Being the RDF stream data model an extension of RDF, SPARQL is the perfect candidate to build event and stream processing languages for it. This section overviews the main approaches for the

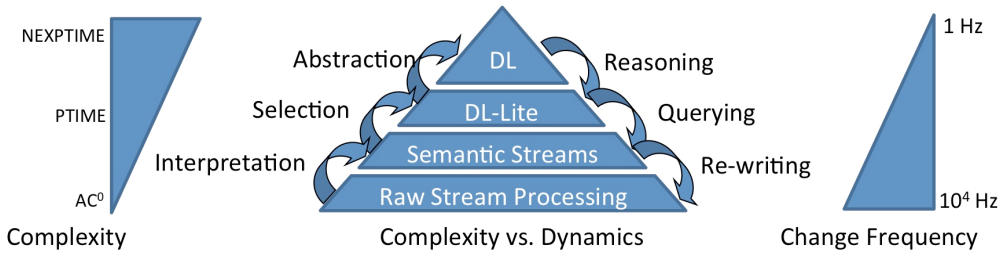


**Figure 2: The intuition of the feasibility of stream reasoning.**

continuous evaluation of processing rules or queries on RDF data streams.

Most approaches inherit the query model of data stream processing systems, and in particular of the CQL language: they offer relations-to-stream (e.g., window) operators to isolate the portions of the input streams that are relevant for processing, relation-to-relation operators to process the content of the streams, and relation-to-stream operators to select some results of the processing and append them to some output streams.

The main difference with respect to data stream processing systems is in the relation-to-relation operators. Data stream processing systems adopt relational operators whereas RDF adopts variants of the SPARQL query language, designed to work with the RDF data model.

A few approaches adopt the model of Complex Event Processing systems, and define the processing tasks in terms of rules that define patterns of interest to be detected in the incoming RDF data streams.

C-SPARQL [15] is a language for continuous queries over streams of RDF data that extends SPARQL by adding operators inspired by the data stream processing model of CQL. The language is implemented in the C-SPARQL engine that builds on top of the Esper and Jena systems. Esper is responsible of executing continuous queries over RDF streams, producing a sequence of RDF graphs over time. Jena executes a standard SPARQL query against each RDF graph in the sequence, producing a continuous result. C-SPARQL offers a limited support to Complex Event Processing temporal operators.

CQELS [40] extends SPARQL with stream-to-relation and relation-to-stream operators. Differently from the C-SPARQL engine that delegates the processing to existing stream and SPARQL engines, CQELS implements the query evaluation engine natively to reduce the processing overhead. The CQELS engine dynamically adapts to the changes in the input data by recompiling the query plan to reduce the processing delay.

$\text{SPARQL}_{\text{stream}}$ [19] is another extension of SPARQL that supports a larger set of streaming operators with respect to C-SPARQL and CQELS. The language is implemented in a query processor that adopts OBDA. It rewrites $\text{SPARQL}_{\text{stream}}$ queries in relational algebra expressions extended with time window constructs, optimizes them, and converts them in the language of a target stream processing engine, such as the Event Processing Language of Esper.

INSTANS [56] models a processing task as a set of interconnected SPARQL queries. INSTANS performs continuous evaluation of incoming RDF data against the compiled set of queries, stores the results into intermediate data structures, and outputs new results when all the conditions in the queries are satisfied. In this sense, INSTANS does not require continuous operators to extend RDF and SPARQL.

## 4.3 Reasoning on RDF streams

In this section, we introduce the state of the art approaches that provide reasoning functionalities for RDF streams. Section 4.3.1 presents techniques for efficient materlization, which is the computation of the complete implicit knowledge that can be derived from the explicit information present in RDF data streams. Section 4.3.2 presents works that extend the continuous evaluation of processing rules or queries with reasoning capabilities.

### *4.3.1 Materialization and Incremental Maintenance*

The term *materialization* refers to the problem of computing all the implicit knowledge that can be derived from some given data according to some ontology. In presence of streaming data that changes frequently techniques that maintain the materialization *incrementally* are required for efficiency.

The origin of incremental maintenance approach can be found in maintenance of materialized views in active databases [21, 60]. This work considers the problem of generating a materialized view and maintain it incrementally through set of updates. When the number of modification in the database is under a threshold, the incremental maintenance techniques perform orders of magnitude faster than the whole re-computation of the view.

Volz et al. [67] propose a declarative variant of the DRed algorithm [60] to incrementally maintain an ontological materialization. The algorithm works in three steps: (i) It overestimates the deletions: starting from the facts that should be deleted, compute the facts that are deducted by them; (ii) It prunes the over-estimated deletions: determine which facts can be rederived by other facts; (iii) It inserts the new deducted facts: derive facts that are consequences of added facts and insert them in the materialization.

Streaming Knowledge Bases [69] is one of the earliest stream reasoning engines. Its approach is to combine a stream processor with a reasoner: in fact, it relies on the TelegraphCQ to efficiently process data streams, and on the Jena rule engine to incrementally materialize the knowledge base.

DynamiTE [66] is a framework to compute the materialization of a knowledge base and update it upon changes. The key novelty of the approach is the introduction of parallelization techniques to improve the performance. In the case of additions, DynamiTE updates the materialization through a parallel evaluation of available axioms. In the case of removal, DynamiTE deletes the explicit concepts and all the derived concepts that are no longer valid. To quickly identify concepts to be removed, authors propose a novel approximate *counting* algorithm that exploits the idea of counting the number of possible ways in which a concept can be derived.

RDFox [49] is an in-memory RDF store characterized by high scalability and performance. Inference is performed through a parallel datalog engine implementing an incremental reasoning algorithm extending DReD. The idea behind this extension is to reduce the number of overestimated deletions, by using backward and forward reasoning to avoid the deletion of axioms that are going to be re-introduced in the rederivation step.

Ren and Pan [54] investigate the possibility to optimize Truth Maintenance Systems to perform expressive incremental reasoning in presence of frequent changes, in the form of updates and deletes. Differently from DRed variations introduced in the aforementioned approaches, they adopt a graph to track dependencies between concepts – the nodes of the graph. Addition operations generate new nodes and edges in the graph, while removal operations are performed by traversing the graph and recursively removing nodes that become unreachable.

StreamRule [48] is similar in spirit, but it uses a different reasoning technique, i.e., Answer Set Programming (ASP) [44] declarative problem solving. StreamRule implements a two-layer approach: the first layer is a stream processing engine that acts as a filter to reduce the amount of data to be considered in the inference process. The second layer is the logic program based on incremental ASP that computes the answer set.

### *4.3.2 Continuous Querying with Reasoning*

The continuous inference task of (continuous) query answering is the point of conjunction between RDF Stream Processing engines, which perform continuous queries over streams of RDF data, and reasoners over RDF streams. This area is only partially explored. The works, presented hereafter, start to move in this direction.

IMaRS [30] is a variation of DRed for the incremental maintenance of the materialization of all the knowledge that is valid in a given window of time. It is one of the first application of DReD to stream reasoning. IMaRS optimizes the computation of incremental maintenance in presence of deletions by exploiting the semantics of windows to determine when a statement is going to expire and thus should be deleted. In this way, it is possible to manage the deletions that, in the general DReD case, cannot be foreseen and are very expensive because they require to determine which consequences become invalid. This allows IMaRS to work out a new correct materialization when a new window is computed by dropping explicit and implicit knowledge that is no longer valid.

Sparkwave [38] performs materialization based on pattern matching over RDF data streams and RDF schema entailment. Sparkwave implements IMaRS on the top of the well known Rete algorithm to compute pattern matching and augments it with RDF schema entailment under the assumption that the ontology does not change over time. Under this assumption, RDF schema axioms can be encoded as rules that are activated by individual RDF triples from the stream. Therefore, each triple from the stream can be treated independently and in a stateless way, which guarantees good performance.

DyKnow [35] is a middleware for autonomous agents that sense and act in a dynamic and changing environment. Such embedded agents take in input row data from the sensors and have to create on the fly qualitative knowledge structures representing aspects of the dynamic environment where they are. Those structures are at the basis of the qualitative reactive reasoning to perform symbol grounding, signal to symbol transformations, information fusion, contextual reasoning, and focus of attention. DyKnow uses real-time CORBA as a communication infrastructure among its distributed components.

ETALIS [4] is built on the top of a Prolog engine and captures event patterns as deductive rules to be evaluated against the streaming data. In particular, ETALIS performs RDF schema entailment that is a relatively simple form of reasoning with good computation complexity.

EP-SPARQL combines event processing operators and SPARQL queries. It is built on the top of ETALIS, and translates EP-SPARQL queries in rules for ETALIS. It follows that, while most of the systems we presented in Section 4.4 evaluate SPARQL queries without using any form of reasoning, EP-SPARQL represents an exception, since it also derives implicit knowledge before performing pattern matching to answer a query. To the best of our knowledge, EP-SPARQL is the only RDF stream processing language to support interval time semantics.

STARQL [51] defines the semantics of stream reasoning in two layers: the first layer denotes an Ontology Language to model the data and its schema, and the second layer is an Embedded Constraint Language for query composition. STARQL offers window operators, clauses to express event matching and a layer to integrate static and streaming data.

## 4.4 Formal Models

The emerging number of approaches, such as the ones presented above, briefly raised the need of compare and contrast them. This introduced the problem of the lack of foundations, key to model and formally define the behaviours of the developed solutions.

To tackle this issue, two works recently emerged, namely RSP-QL [31] and LARS [17]. They follow two different approaches: the former starts by a query processing model, SPARQL, and extends it in order to capture the behaviour of RSP and SR engines; the latter introduce window as operator in modal logics.

In the direction of finding agreements on the approaches, it is worth mentioning the ongoing effort of the W3C RSP community group[2]. The group is working in the direction of defining common models for RDF streams and relative processing, and protocols for their exchange across the Web.

RSP-QL [31] is a formal model to describe the evaluation semantics of stream reasoning systems in the context of continuous query answering. RSP-QL moves the evaluation semantics of the model from one time (as in SPARQL) to continuous. That means, RSP-QL produces streams of answers, computed at different time instants, to cope with the fact that the data on the stream is dynamic and changes over time. It is used as basis do introduce event and data stream processing inspired operators, such as sliding windows and event patterns. RSP-QL captures the evaluation semantics of most of the RSP engines, e.g., C-SPARQL, CQELS and SPARQL$_{stream}$.

LARS [17] defines a logic to precisely define the data and processing model for a stream reasoning engine. Concerning the data, LARS models the notion of stream as sequence of time-annotated formulas. In addition to the usual logic operators —conjunction, disjunction, implication, negation— the authors define four temporal logic operators: (i) $\diamond$ indicates that a formula holds at some time in the past; (ii) $\Box$ indicated that a formula always holds in the past; (iii) $@t$ indicates that a formula holds at the specific point in time $t$; (iv) $\boxplus$ indicates that a formula holds in a given time interval, and is used to express the semantics of time windows. The authors prove that LARS captures the semantics of the CQL and Etalis languages.

## 4.5 Dealing with Data Veracity

Streaming information is often incomplete and noisy. Some recent work [10, 11, 16] demonstrated the possibility to effectively deal with noisy and incomplete social media streams by coupling deductive stream reasoning with relational learning.

[65] proposes a new approach to OBDA for data streams to handle fuzzy and temporal information. The system can answer (temporal) fuzzy conjunctive queries over fuzzy data streams with respect to a (crisp) DL-Lite ontology. This enables the use of standard query rewriting engines while dealing with noisy data.

In [50] a framework is proposed for dealing with inconsistencies, noisy data, and probabilistic processing rules in RDF data streams

[2]Cf. https://www.w3.org/community/rsp/.

and Linked Data. The framework reasons about dynamic Web data using probabilistic Answer Set Programming (ASP) [14].

Probabilistic Event Calculus [59] proposes to deal with uncertainty in logic-based event recognition by extending the Event Calculus [58] with Markov logic networks [55].

In [42], statistical learning and stream reasoning are combined: the former is used to build an ontology that is used by the latter to perform reasoning. The final goal is to predict the upcoming content of the stream, e.g., the traffic conditions of cities [62].

# 5. DISCUSSION AND FUTURE WORKS

The research conducted so far has shown that stream reasoning is indeed possible: it is no longer something that needs to be proved, but rather something that needs to be improved.

Stream reasoning provides the technology stack to tame variety in data streams by means of (*i*) a data model to represent heterogenous data streams, i.e., RDF streams, (*ii*) continuous queries languages (see Section 4.2), and (*iii*) continuous reasoning techniques (see Section 4.3).

Moreover, Sections 4.3.2 and 4.5 respectively show that it is possible to tame velocity and variety simultaneously by optimizing the continuous querying and continuous reasoning tasks so to provide reactive answers, and also to tame veracity by combining deductive stream reasoning with other techniques robust to noise.

Some real-world applications were built on top of stream reasoning technologies. This happened in the areas of Social Media Analytics [10, 13, 11, 16] and Smart Cities [12, 53, 63].

However, existing approaches have some important limitations that require future work and that make stream reasoning an open field of research.

First, the work on expressive deductive stream reasoning is at an intermediate stage. Existing stream reasoning approaches are fragmented: some focus on temporal reasoning, some on rule base reasoning, some on expressive Description Logics, some on even more complex reasoning techniques like ASP. A unified approach to stream reasoning has not been elaborated yet, despite the promising work presented in Section 4.4. The same applies for the ability of the stream reasoning systems to deal with the variety dimension; approaches exists, but they lack a common theory.

Second, the streams are parallel and distributed in nature, so far only [52] has reported on successful investigation on distributed and parallel RDF Stream Processing, while on parallel stream reasoning some work in progress was reported in [37] and [45]. Similarly, the processing might involve distributed static data that is too large to be fetched locally at one node (e.g., on the Web). Techniques like [26, 72] are therefore needed to identify and retrieve this data on-demand, in order to enable more sophisticated inference and control at the same time the impact on engine responsiveness.

Third, the long term success of stream reasoning requires a framework for comparative evaluation [64, 29]. The community needs a comprehensive and widely accepted benchmark that can be used to provide concrete evidence that stream reasoning is the best solution in some domain. It is of paramount importance to include in the comparison also state-of-the-art solutions like DSMS and CEP.

Finally, data streams are only an example of ordered dataset. The presence of effective order-sensitive processing techniques[30, 38, 49] open the opportunity to harness other types of ordering relations among items [28], for instance to investigate top-k query answering.

# 6. CONCLUSIONS

This tutorial paper presents an overview of stream reasoning, a recent and growing research area that aims to combine the benefits of stream processing systems and reasoning engines to better capture the requirement of modern data intensive applications, specifically concerning the velocity and variety of input data.

The paper offers an overview of the research on data and event stream processing and reasoning, which represent the foundational elements for stream reasoning. The paper then motivates the need for stream reasoning by analyzing the requirements of some scenarios, and presents the current state of the art of the research on stream reasoning.

The goal of the tutorial paper is to highlight the open questions and challenges in the domain, to stimulate the discussion and promote further developments.

# 7. REFERENCES


[1] D. J. Abadi, D. Carney, U. Çetintemel, M. Cherniack, C. Convey, S. Lee, M. Stonebraker, N. Tatbul, and S. Zdonik. Aurora: A new model and architecture for data stream management. *The VLDB Journal*, 12(2):120–139, 2003.

[2] A. Alexandrov, R. Bergmann, S. Ewen, J.-C. Freytag, F. Hueske, A. Heise, O. Kao, M. Leich, U. Leser, V. Markl, F. Naumann, M. Peters, A. Rheinländer, M. J. Sax, S. Schelter, M. Höger, K. Tzoumas, and D. Warneke. The stratosphere platform for big data analytics. *The VLDB Journal*, 23(6):939–964, 2014.

[3] D. Anicic, P. Fodor, N. Stojanovic, and R. Stühmer. An approach for data-driven and logic-based complex event processing. In *Proceedings of the International Conference on Distributed Event-Based Systems*, DEBS '09, pages 26:1–26:2. ACM, 2009.

[4] D. Anicic, S. Rudolph, P. Fodor, and N. Stojanovic. Stream reasoning and complex event processing in etalis. *Semantic Web*, 3(4):397–407, 2012.

[5] A. Arasu, B. Babcock, S. Babu, M. Datar, K. Ito, I. Nishizawa, J. Rosenstein, and J. Widom. Stream: The stanford stream data manager (demonstration description). In *Proceedings of the International Conference on Management of Data*, SIGMOD '03, pages 665–665. ACM, 2003.

[6] A. Arasu, S. Babu, and J. Widom. The cql continuous query language: Semantic foundations and query execution. *The VLDB Journal*, 15(2):121–142, 2006.

[7] A. Artale, D. Calvanese, R. Kontchakov, and M. Zakharyaschev. The dl-lite family and relations. *J. Artif. Intell. Res. (JAIR)*, 36:1–69, 2009.

[8] F. Baader and W. Nutt. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[9] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom. Models and issues in data stream systems. In *Proceedings Symposium on Principles of Database Systems*, PODS '02, pages 1–16. ACM, 2002.

[10] M. Balduini, A. Bozzon, E. Della Valle, Y. Huang, and G. Houben. Recommending venues using continuous predictive social media analytics. *IEEE Internet Computing*, 18(5):28–35, 2014.

[11] M. Balduini, I. Celino, D. Dell'Aglio, E. D. Valle, Y. Huang, T. K. Lee, S. Kim, and V. Tresp. BOTTARI: an augmented reality mobile application to deliver personalized and location-based recommendations by continuous analysis of social media streams. *J. Web Sem.*, 16:33–41, 2012.

[12] M. Balduini, E. Della Valle, M. Azzi, R. Larcher, F. Antonelli, and P. Ciuccarelli. Citysensing: Fusing city data for visual storytelling. *IEEE MultiMedia*, 22(3):44–53, 2015.

[13] M. Balduini, E. Della Valle, D. Dell'Aglio, M. Tsytsarau, T. Palpanas, and C. Confalonieri. Social listening of city scale events using the streaming linked data framework. In *International Semantic Web Conference (2)*, volume 8219 of *LNCS*, pages 1–16. Springer, 2013.

[14] C. Baral, M. Gelfond, and J. N. Rushton. Probabilistic reasoning with answer sets. *TPLP*, 9(1):57–144, 2009.

[15] D. F. Barbieri, D. Braga, S. Ceri, E. Della Valle, and M. Grossniklaus. C-sparql: a continuous query language for rdf data streams. *International Journal of Semantic Computing*, 04(01):3–25, 2010.

[16] D. F. Barbieri, D. Braga, S. Ceri, E. Della Valle, Y. Huang, V. Tresp, A. Rettinger, and H. Wermser. Deductive and inductive stream reasoning for semantic social media analytics. *IEEE Intelligent Systems*, 25(6):32–41, 2010.

[17] H. Beck, M. Dao-Tran, T. Eiter, and M. Fink. Lars: A logic-based framework for analyzing reasoning over streams. In *Proceedings of the AAAI Conference on Artificial Intelligence*, AAAI '15, pages 1431–1438. AAAI Press, 2015.

[18] L. Brenna, A. Demers, J. Gehrke, M. Hong, J. Ossher, B. Panda, M. Riedewald, M. Thatte, and W. White. Cayuga: A high-performance event processing engine. In *Proceedings of the International Conference on Management of Data*, SIGMOD '07, pages 1100–1102. ACM, 2007.

[19] J. Calbimonte, Ó. Corcho, and A. J. G. Gray. Enabling ontology-based access to streaming data sources. In *International Semantic Web Conference (1)*, LNCS, pages 96–111. Springer, 2010.

[20] D. Calvanese, M. Koubarakis, and D. Toman. Special issue of the journal of web semantics on ontology-based data access. *J. Web Sem.*, 33:1–2, 2015.

[21] S. Ceri and J. Widom. Deriving production rules for incremental view maintenance. In *Proceedings of the International Conference on Very Large Data Bases*, VLDB '91, pages 577–589. Morgan Kaufmann, 1991.

[22] G. Cugola and A. Margara. Tesla: A formally defined event specification language. In *Proceedings of the International Conference on Distributed Event-Based Systems*, DEBS '10, pages 50–61. ACM, 2010.

[23] G. Cugola and A. Margara. Processing flows of information: From data stream to complex event processing. *ACM Computing Surveys*, 44(3):15:1–15:62, 2012.

[24] R. Cyganiak, D. Wood, and M. Lanthaler. Rdf 1.1 concepts and abstract syntax. Technical report, 2014.

[25] S. Das, S. Sundara, and R. Cyganiak. R2RML: RDB to RDF Mapping Language. Technical report, W3C Recommendation, 2012.

[26] S. Dehghanzadeh, D. Dell'Aglio, S. Gao, E. Della Valle, A. Mileo, and A. Bernstein. Approximate continuous query answering over streams and dynamic linked data sets. In *ICWE*, volume 9114 of *LNCS*, pages 307–325. Springer, 2015.

[27] E. Della Valle, S. Ceri, F. v. Harmelen, and D. Fensel. It's a streaming world! reasoning upon rapidly changing information. *IEEE Intelligent Systems*, 24(6):83–89, 2009.

[28] E. Della Valle, S. Schlobach, M. Krötzsch, A. Bozzon, S. Ceri, and I. Horrocks. Order matters! harnessing a world of orderings for reasoning over massive data. *Semantic Web*, 4(2):219–231, 2013.

[29] D. Dell'Aglio, J. Calbimonte, M. Balduini, Ó. Corcho, and E. D. Valle. On correctness in RDF stream processor benchmarking. In *International Semantic Web Conference (2)*, volume 8219 of *Lecture Notes in Computer Science*, pages 326–342. Springer, 2013.

[30] D. Dell'Aglio and E. Della Valle. Incremental reasoning on RDF streams. In *Linked Data Management*, pages 413–435. Chapman and Hall/CRC, 2014.

[31] D. Dell'Aglio, E. Della Valle, J. Calbimonte, and Ó. Corcho. RSP-QL semantics: A unifying query model to explain heterogeneity of RDF stream processing systems. *Int. J. Semantic Web Inf. Syst.*, 10(4):17–44, 2014.

[32] G. Erétéo, M. Buffa, F. Gandon, and O. Corby. Analysis of a real online social network using semantic web frameworks. In *In Proceedings of the International The Semantic Web Conference*, ISWC '09, pages 180–195. Springer, 2009.

[33] O. Etzion and P. Niblett. *Event Processing in Action*. Manning Publications Co., 2010.

[34] S. Harris and A. Seaborne. Sparql 1.1 query language. Technical report, W3C Recommendation, 2013.

[35] F. Heintz and P. Doherty. Dyknow: An approach to middleware for knowledge processing. *Journal of Intelligent and Fuzzy Systems*, 15(1):3–13, 2004.

[36] P. Hitzler, M. Krotzsch, B. Parsia, P. F. Patel-Schneider, and S. Rudolph. OWL 2 Web Ontology Language Primer. Technical report, W3C Recommendation, 2012.

[37] J. Hoeksema and S. Kotoulas. High-performance distributed stream reasoning using s4. In *First International Workshop on Ordering and Reasoning, Bonn, Germany*, 2011.

[38] S. Komazec, D. Cerri, and D. Fensel. Sparkwave: Continuous schema-enhanced pattern matching over rdf data streams. In *Proceedings of the International Conference on Distributed Event-Based Systems*, DEBS '12, pages 58–68. ACM, 2012.

[39] S. Kulkarni, N. Bhagat, M. Fu, V. Kedigehalli, C. Kellogg, S. Mittal, J. M. Patel, K. Ramasamy, and S. Taneja. Twitter heron: Stream processing at scale. In *Proceedings of the International Conference on Management of Data*, SIGMOD '15, pages 239–250. ACM, 2015.

[40] D. Le-Phuoc, M. Dao-Tran, J. X. Parreira, and M. Hauswirth. A native and adaptive approach for unified processing of linked streams and linked data. In *Proceedings of the International Semantic Web Conference*, ISWC '11, pages 370–388. Springer-Verlag, 2011.

[41] F. Lecue, S. Kotoulas, and P. M. Aonghusa. Capturing the pulse of cities: Opportunity and research challenges for robust stream data reasoning. In *AAAI Workshops*, AAAI '12, 2012.

[42] F. Lécué and J. Z. Pan. Predicting knowledge in an ontology stream. In *IJCAI*. IJCAI/AAAI, 2013.

[43] M. Lenzerini. Data integration: A theoretical perspective. In L. Popa, S. Abiteboul, and P. G. Kolaitis, editors, *Proceedings of the Twenty-first ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 3-5, Madison, Wisconsin, USA*, pages 233–246. ACM, 2002.

[44] V. Lifschitz. What is answer set programming? In D. Fox and C. P. Gomes, editors, *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008, Chicago, Illinois, USA, July 13-17, 2008*, pages 1594–1597. AAAI Press, 2008.

[45] C. Liu, J. Urbani, and G. Qi. Efficient RDF stream reasoning with graphics processingunits (GPUs). In C.-W. Chung, A. Z. Broder, K. Shim, and T. Suel, editors, *WWW (Companion Volume)*, pages 343–344. ACM, 2014.

[46] D. C. Luckham. *The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems*.

Addison-Wesley, 2001.

[47] A. Margara, J. Urbani, F. van Harmelen, and H. Bal. Streaming the web: Reasoning over dynamic data. *Web Semantics: Science, Services and Agents on the World Wide Web*, 25(C):24–44, 2014.

[48] A. Mileo, A. Abdelrahman, S. Policarpio, and M. Hauswirth. Streamrule: A nonmonotonic stream reasoning system for the semantic web. In *Proceedings of the International Conference on Web Reasoning and Rule Systems*, RR '13, pages 247–252. Springer-Verlag, 2013.

[49] Y. Nenov, R. Piro, B. Motik, I. Horrocks, Z. Wu, and J. Banerjee. Rdfox: A highly-scalable RDF store. In *Proceedings of the International Semantic Web Conference*, pages 3–20. Springer, 2015.

[50] M. Nickles and A. Mileo. Web stream reasoning using probabilistic answer set programming. In R. Kontchakov and M. Mugnier, editors, *Web Reasoning and Rule Systems - 8th International Conference, RR 2014, Athens, Greece, September 15-17, 2014. Proceedings*, volume 8741 of *LNCS*, pages 197–205. Springer, 2014.

[51] Ö. L. Özçep, R. Möller, and C. Neuenstadt. A stream-temporal query language for ontology based data access. In *Proceedings of the International Workshop on Description Logics*, pages 696–708, 2014.

[52] D. L. Phuoc, H. N. M. Quoc, C. L. Van, and M. Hauswirth. Elastic and scalable processing of linked stream data in the cloud. In *International Semantic Web Conference (1)*, volume 8218 of *LNCS*, pages 280–297. Springer, 2013.

[53] D. Puiu, P. M. Barnaghi, R. Toenjes, D. Kuemper, M. I. Ali, A. Mileo, J. X. Parreira, M. Fischer, S. Kolozali, N. FarajiDavar, F. Gao, T. Iggena, T. Pham, C. Nechifor, D. Puschmann, and J. Fernandes. Citypulse: Large scale data analytics framework for smart cities. *IEEE Access*, 4:1086–1108, 2016.

[54] Y. Ren and J. Z. Pan. Optimising ontology stream reasoning with truth maintenance system. In *Proceedings of the International Conference on Information and Knowledge Management*, CIKM '11, pages 831–836. ACM, 2011.

[55] M. Richardson and P. M. Domingos. Markov logic networks. *Machine Learning*, 62(1-2):107–136, 2006.

[56] M. Rinne, E. Nuutila, and S. Törmä. INSTANS: high-performance event processing with standard RDF and SPARQL. In *Proceedings of the ISWC Posters & Demonstrations Track*, CEUR Workshop Proceedings. CEUR-WS.org, 2012.

[57] N. Shadbolt, T. Berners-Lee, and W. Hall. The semantic web revisited. *IEEE Intelligent Systems*, 21(3):96–101, 2006.

[58] M. Shanahan. The event calculus explained. In *Artificial Intelligence Today*, pages 409–430. 1999.

[59] A. Skarlatidis, G. Paliouras, A. Artikis, and G. A. Vouros. Probabilistic event calculus for event recognition. *ACM Trans. Comput. Log.*, 16(2):11:1–11:37, 2015.

[60] M. Staudt and M. Jarke. Incremental maintenance of externally materialized views. In *Proceedings of the International Conference on Very Large Data Bases*, VLDB '96, pages 75–86. Morgan Kaufmann, 1996.

[61] H. Stuckenschmidt, S. Ceri, E. Della Valle, and F. van Harmelen. Towards expressive stream reasoning. In K. Aberer, A. Gal, M. Hauswirth, K. Sattler, and A. P. Sheth, editors, *Semantic Challenges in Sensor Networks, 24.01. - 29.01.2010*, volume 10042 of *Dagstuhl Seminar Proceedings*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Germany, 2010.

[62] S. Tallevi-Diotallevi, S. Kotoulas, L. Foschini, F. Lécué, and A. Corradi. Real-time urban monitoring in dublin using semantic and stream technologies. In *In Proceedings of the International The Semantic Web Conference*, ISWC '13, pages 178–194. Springer, 2013.

[63] S. Tallevi-Diotallevi, S. Kotoulas, L. Foschini, F. Lécué, and A. Corradi. Real-time urban monitoring in dublin using semantic and stream technologies. In *International Semantic Web Conference (2)*, volume 8219 of *LNCS*, pages 178–194. Springer, 2013.

[64] R. Tommasini, E. Della Valle, M. Balduini, and D. Dell'Aglio. Heaven: a framework for systematic comparative research approach for rsp engines. In *The Semantic Web: Semantics and Big Data, 13th International Conference, ESWC 2016, Heraklion, Crete, Greece, May 29 - June 2, 2016. Proceedings*, 2016.

[65] A. Turhan and E. Zenker. Towards temporal fuzzy query answering on stream-based data. In D. Nicklas and Ö. L. Özçep, editors, *Proceedings of the 1st Workshop on High-Level Declarative Stream Processing co-located with the 38th German AI conference (KI 2015), Dresden, Germany, September 22, 2015.*, volume 1447 of *CEUR Workshop Proceedings*, pages 56–69. CEUR-WS.org, 2015.

[66] J. Urbani, A. Margara, C. Jacobs, F. Harmelen, and H. Bal. Dynamite: Parallel materialization of dynamic rdf data. In *Proceedings of the International Semantic Web Conference*, ISWC '13, pages 657–672. Springer, 2013.

[67] R. Volz, S. Staab, and B. Motik. Journal on data semantics. chapter Incrementally Maintaining Materializations of Ontologies Stored in Logic Databases, pages 1–34. Springer-Verlag, 2005.

[68] A. Wagner, S. Speiser, and A. Harth. Semantic web technologies for a smart energy grid: Requirements and challenges. In *In proceedings of the International Semantic Web Conference*, ISWC '10, pages 33–37. Springer, 2010.

[69] O. Walavalkar, A. Joshi, T. Finin, and Y. Yesha. Streaming knowledge bases. In *Proceedings of the International Workshop on Scalable Semantic Web Knowledge Base Systems*, SSWS '08, 2008.

[70] W. White, M. Riedewald, J. Gehrke, and A. Demers. What is "next" in event processing? In *Proceedings of the Symposium on Principles of Database Systems*, PODS '07, pages 263–272. ACM, 2007.

[71] M. Zaharia, T. Das, H. Li, T. Hunter, S. Shenker, and I. Stoica. Discretized streams: Fault-tolerant streaming computation at scale. In *Proceedings of the Symposium on Operating Systems Principles*, SOSP '13, pages 423–438. ACM, 2013.

[72] S. Zahmatkesh, E. Della Valle, and D. Dell'Aglio. When a filter makes the difference in continuously answering sparql queries on streaming and quasi-static linked data. In *ICWE*, LNCS. Springer, 2016.