



HAL
open science

Grounding Stream Reasoning Research

Pieter Bonte, Jean-Paul Calbimonte, Daniel de Leng, Daniele Dell’Aglia, Emanuele Della Valle, Thomas Eiter, Federico Giannini, Fredrik Heintz, Konstantin Schekotihin, Danh Le-Phuoc, et al.

► **To cite this version:**

Pieter Bonte, Jean-Paul Calbimonte, Daniel de Leng, Daniele Dell’Aglia, Emanuele Della Valle, et al.. Grounding Stream Reasoning Research. Transactions on Graph Data and Knowledge, 2024, 10.4230/TGDK.2.1.2 . hal-04792478

HAL Id: hal-04792478

<https://hal.science/hal-04792478v1>

Submitted on 20 Nov 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Distributed under a Creative Commons Attribution 4.0 International License

Grounding Stream Reasoning Research

Pieter Bonte ✉ 🏠 

Department of Computer Science, KU Leuven Campus Kulak, Belgium

Daniel de Leng ✉ 

Linköping University, Sweden

Emanuele Della Valle ✉ 

DEIB - Politecnico di Milano, Italy

Federico Giannini ✉ 

DEIB - Politecnico di Milano, Italy

Konstantin Schekotihin ✉ 

Alpen-Adria-Universität Klagenfurt, Austria


Alessandra Mileo ✉ 🏠 

Insight Centre for Data Analytics, Dublin City University, Ireland

Riccardo Tommasini ✉ 🏠 

INSA Lyon, CNRS LIRIS, France


University of Tartu, Estonia

Giacomo Ziffer ✉ 

DEIB - Politecnico di Milano, Italy

Jean-Paul Calbimonte ✉ 🏠 

University of Applied Sciences and Arts Western Switzerland HES-SO, Sierre, Switzerland

Daniele Dell’Aglio ✉ 🏠 

Aalborg University, Denmark

Thomas Eiter ✉ 

Technische Universität Wien, Austria

Fredrik Heintz ✉ 

Linköping University, Sweden

Danh Le-Phuoc ✉ 🏠 

Technical University Berlin, Germany

Patrik Schneider ✉ 

Technische Universität Wien, Austria

Siemens AG, Chemnitz, Germany

Jacopo Urbani ✉ 

Vrije Universiteit Amsterdam, The Netherlands

Abstract

In the last decade, there has been a growing interest in applying AI technologies to implement complex data analytics over data streams. To this end, researchers in various fields have been organising a yearly event called the “Stream Reasoning Workshop” to share perspectives, challenges, and experiences around this topic.

In this paper, the previous organisers of the workshops and other community members provide a summary of the main research results that have been discussed during the first six editions of the event. These results can be categorised into four main research areas: The first is concerned with the technological challenges related to handling large

data streams. The second area aims at adapting and extending existing semantic technologies to data streams. The third and fourth areas focus on how to implement reasoning techniques, either considering deductive or inductive techniques, to extract new and valuable knowledge from the data in the stream.

This summary is written not only to provide a crystallisation of the field, but also to point out distinctive traits of the stream reasoning community. Moreover, it also provides a foundation for future research by enumerating a list of use cases and open challenges, to stimulate others to join this exciting research area.

2012 ACM Subject Classification Information systems → Data streams; Information systems → Stream management; Information systems → Graph-based database models; Information systems → Query languages for non-relational engines; Computing methodologies → Temporal reasoning; Computing methodologies → Description logics; Information systems → Semantic web description languages

Keywords and phrases Stream Reasoning, Stream Processing, RDF streams, Streaming Linked Data, Continuous query processing, Temporal Logics, High-performance computing, Databases

Digital Object Identifier 10.4230/TGDK.2.1.2

Category Position

Funding J.-P. Calbimonte acknowledges support from the Swiss National Science Foundation under grant No. 213369 (*StreamKG*), and from the EU Horizon Europe program under grant No. 101092908 (*SmartEdge*). D. Le-Phuoc is supported by the Deutsche Forschungsgemeinschaft, German Research



© Pieter Bonte, Jean-Paul Calbimonte, Daniel de Leng, Daniele Dell’Aglio, Emanuele Della Valle, Thomas Eiter, Federico Giannini, Fredrik Heintz, Konstantin Schekotihin, Danh Le-Phuoc, Alessandra Mileo, Patrik Schneider, Riccardo Tommasini, Jacopo Urbani, and Giacomo Ziffer; licensed under Creative Commons License CC-BY 4.0

Transactions on Graph Data and Knowledge, Vol. 2, Issue 1, Article No. 2, pp. 2:1–2:47



Transactions on Graph Data and Knowledge

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Foundation under grant number 453130567 (*COSMO*) and by the Horizon Europe Research and Innovation Actions under grant number 101092908 (*SmartEdge*). A. Mileo acknowledges support from Science Foundation Ireland (SFI) Grant Number SFI/12/RC/2289_P2, co-funded by the European Regional Development Fund. T. Eiter acknowledges support from the Vienna Science and Technology Fund (WWTF) project ICT22-023 (*TAIGER*) and the EU Horizon Europe program under grant No. 820437 (*Humane AI Net*).

Received 2023-09-18 Accepted 2023-11-17 Published 2024-05-03

Editors Aidan Hogan, Ian Horrocks, Andreas Hotho, and Lalana Kagal

Special Issue Trends in Graph Data and Knowledge – Part 2

1 Introduction

Stream Reasoning (SR) has emerged as a branch of artificial intelligence that draws attention to the need to make decisions incrementally, as soon as possible, and before they are no longer helpful. Such an ambitious and broad goal requires many competencies as it entails different research problems. As a result, Stream Reasoning bridges several research communities, such as Knowledge Representation, Robotics, Data Management, and Semantic Web, and it has found applications in various application domains, including traffic management, social media analytics, and robotics.

For more than a decade, the stream reasoning community has proceeded with a shared vision and provided many independent contributions. In this paper, a few community members, some active since the beginning and some recently welcomed provide an overview of the leading research contributions within the Stream Reasoning field discussed during these events. Moreover, this article aims to crystallise the notion of stream reasoning, examining how these different communities contributed to various aspects of its research vision and highlighting the overlaps and peculiarities. The inputs of this crystallisation process were the programs and discussions of the past workshops and the results of a questionnaire prepared specifically for this article. Some authors prepared the questionnaire, starting from one of the initial research questions [75], called **Q** henceforth, that contributed to the fundamental vision of Stream Reasoning:

(**Q**) Can we make sense in near real-time of vast, rapidly evolving, constantly varying, inevitably noisy, incomplete and heterogeneous data streams coming from complex domains?

This question touches upon the various research dimensions related to SR: *Near real-time* pertains to the urgency of obtaining an answer; it is essential to secure a response as swiftly as possible and definitely before the information loses its value, a notion referred to as *velocity*. A unique challenge is given by *heterogeneous data*, emphasising the variety of data, where data is not uniformly formatted. The term *noisy* refers to the inherent uncertainty about and in the data. When one mentions data being *vast*, they point to the immense volume of data generated in a given time frame, signifying scalability challenges. *Incomplete* data suggests an absence of specific data or information in the stream. The description *rapidly evolving* underscores the unpredictable nature of the stream's ingestion rate, while *constantly varying* speaks to the unpredictability of the content of the stream and its potential constituents. In conclusion, the term *Complex domains*, which perhaps distinguishes Stream Reasoning from the related topic of Stream Processing, is reserved for those application areas where merely validating data is not sufficient; these domains necessitate capturing and integrating semantics, complex relations between parts, and context through a more expressive language.

Question **Q** can also be used to define a macro-level perspective on decision-making. In social sciences, macro-level questions correspond to the collective investigation of a research field, i.e., they are used to define the research context [131]. As such, they remain unanswered regardless of individual contributions, and thus, they shall be reduced to simpler lower-level questions. In particular, two additional levels are expected:

- **Meso level:** adds requirements to limit the research context, but still unsolvable. Meso-level questions roughly correspond to the investigation of several PhD theses.
- **Micro level:** reduces the investigation to a measurable outcome that can assess the validity of the contribution, e.g., a research paper.

Therefore, we used the answers to the questionnaire to reformulate **Q** into meso and micro questions to characterise the different areas within Stream Reasoning research. Moreover, we asked the participants to sustain their answers with a thorough analysis of the Stream Reasoning state of the art. Our goal is indeed grounding the pillars of Stream Reasoning research to the extent of guiding the future of this research community. The result redefines the aforementioned terms into four partly overlapping areas of research:

Stream Processing, i.e., the area concerned with developing systems that can efficiently process large data streams. Given the focus on data management, this research area is traditionally embedded in the database and complex event processing communities.

Streaming Linked Data, i.e., the area that focuses on extending the Semantic Web stack to deal with streaming data. Because of this, contributions in this area are primarily presented in the Semantic Web community.

Deductive Stream Reasoning, i.e., the area that focuses on designing deductive reasoning techniques that can infer implicit knowledge from the stream. Most techniques in this area are based on logic-based methods and come from the Knowledge Representation community.

Inductive Stream Reasoning, i.e., the area that studies how we can infer new knowledge using inductive reasoning techniques. To this end, the most recent contributions exploit the latest developments in Machine Learning to learn new knowledge from the data.

While SLD encompasses several areas of research that are interested in data sharing and integration for evolving data, inductive and deductive stream reasoning focus on efficiency and expressiveness. To clarify the difference between deductive and inductive reasoning, in deductive reasoning, one evaluates logical statements to make implicit knowledge explicit; prototypical reasoning is making proofs in a logical calculus, applying rules, etc. For example, from a and $a \rightarrow b$ we may conclude b . Notably, the reasoning is sound. In inductive reasoning, one infers rules from data; e.g., from images showing white swans, one may infer that, as a rule, swans are white. In contrast to deductive reasoning, inductive reasoning is not generally sound, and the result may be incorrect. To address this, inferences may be drawn under uncertainty, often resorting to probabilities. Notably, deductive reasoning may involve uncertainty, but all knowledge is already implicit.

We will discuss each of the areas in the following four sections. The discussion will follow the same structure in each section. First, we will formulate a meso question for that specific sub-area of stream reasoning, and then we will dig into the following sections:

- In the “Make Sense” section, we investigate the standard way to express a Stream Reasoning problem in that sub-area, e.g., continuous querying or logical program.
- In the section “Taming Volume”, we describe what research efforts in that particular sub-area address the scalability problem, e.g., using distributed systems to scale out;
- In the section “Taming Velocity”, we focus on those research efforts in that sub-area that relate to the hurdle of processing data as soon as possible, e.g., adopting window-based processing;

2:4 Grounding Stream Reasoning Research

- In the section “Taming Variety”, we discuss how existing works in that sub-area approach the challenge of information integration, e.g., using graph-based data models;
- In the section “Domain Complexity”, we present the results for representing domain knowledge, e.g., ontologies;
- Finally, in the “Data Quality” section, we discuss what methods were adopted or assumptions were made regarding data quality issues, e.g., missing data.

We will conclude in Section 6 with a discussion on how the various areas relate, primarily pointing out when they overlap and how to move forward in this exciting field with a list of what we view as critical open challenges.

2 Stream Processing

Stream Processing is a technological solution and a research field that first addressed the problem of continuously analysing data in near-real-time. Stream Processing pre-dates Stream Reasoning research. Indeed, Execution models for Stream Processing have been around for decades [159]. Therefore, it had a direct influence on Stream Reasoning research. At the same time, the push towards a broad form of intelligence and decision support that does not neglect reactivity, which is one common theme in Stream Reasoning research, had a return on Stream Processing as a field.

Thus, it makes sense to look at Stream Processing from a Stream Reasoning perspective, to understand how it contributes to the latter vision. To this extent, we formulate the research question below to capture the objectives of Stream Processing research that align with the one captured by the macro question:

Meso (Stream Processing): Can we continuously query, using declarative SQL-like languages, vast, rapidly evolving, constantly varying, potentially noisy data streams, minimising latency and maximising throughput?

In the remainder of the section, we discuss how Stream Processing has answered such a question.

Stream Processing covers the whole life-cycle of streaming data: from their ingress to manipulation and eventual egress.

2.1 Make Sense

As motivated by Cugola and Margara [67] in their overview of what they call information flow processing systems:

Many distributed applications require continuous and timely information processing as they flow from the periphery to the system’s centre.

Such Stream Processing systems are designed to support large applications in which data are generated from multiple sources and pushed asynchronously to servers responsible for analysing them [115]. Traditionally, analytics is the main objective of the processing, with Stream Processing systems focusing on low-latency, high-throughput online analytical processing (OLAP) workloads.

In terms of **making sense** of the data, Stream Processing introduced the notion of Continuous Querying, i.e., queries that continuously run against streaming or real-time data to produce results or output whenever new data meets the query’s conditions. Traditional database queries are one-time operations: a query is executed and results are obtained based on the current state of the database. In contrast, continuous queries persist and constantly check incoming data.

■ **Table 1** Description of the taxi *ride* stream data. ■ **Table 2** Description of the taxi *fare* stream data.

field	description	field	description
<i>rideId</i>	the unique ride id	<i>rideId</i>	the unique ride id
<i>taxiId</i>	the unique id for the taxi itself	<i>taxiId</i>	the unique id for the taxi itself
<i>driverId</i>	the unique id of the taxi driver	<i>driverId</i>	the unique id of the taxi driver
<i>isStart</i>	has the ride has started or ended	<i>startTime</i>	the time the ride started
<i>eventTime</i>	timestamp of the event	<i>paymentType</i>	the type of payment(<i>cash/card</i>)
<i>startLon</i>	the longitude where the ride started	<i>tip</i>	the tip amount for the ride
<i>startLat</i>	the latitude where the ride started	<i>tolls</i>	the amount of tolls paid
<i>endLon</i>	the longitude where the ride ended	<i>totalFare</i>	the total fare
<i>endLat</i>	the latitude where the ride ended		
<i>passengerCnt</i>	the number of passengers		

Continuous Queries are more specialised than general coding tasks, and thus, they are typically supported by algebra or formal semantics. To our knowledge, the first appearance dates back to the seminal work of Terry et al. [211]. Since then, continuous queries have been discussed extensively [23, 16, 62]. Limiting our mention to fully-declarative languages, we can distinguish two families of continuous queries, which differ on the expressivity of the languages they use:

- SQL-Like Languages based on the foundational CQL models by Arasu et al [9]. Such languages allow expression *window-based* continuous queries over relational data streams. Three types of windows have been considered: *time-based (sliding) window* which discards all data beyond a certain point in time; *tuple-based window* which dumps all data that has arrived prior to a predefined number of tuples (e.g., keep only the last 10 facts); *partition-based windows*, which partitions the stream in various substreams based on the attributes of the data in the stream.
- Complex Event Recognition Languages focus on detecting regular expressions over streams of typed events. Although operators like Sequence (follow by) and Allen Algebras are well-accepted, a universally accepted foundational algebra is still missing.

► **Example 1 (Taxi).** To illustrate the difference between the research areas, we provide examples of various queries typical for each research area. We will utilise the taxi dataset provided by the ACM DEBS 2015 Grand Challenge¹ as an ongoing example. The DEBS challenge centers around analysing taxi routes within the city of New York. The dataset encompasses two streams: the *ride* stream, which describes taxi journeys including (i) taxi specifications, (ii) pick-up and drop-off details (such as geographical coordinates and timestamps); and (iii) passenger count; and the *fare* stream, which describes payment details for the rides (such as tip, payment method, and total fare). Specifically, Table 1 outlines the attributes found in the *ride* stream, while Table 2 delineates the attributes in the *fare* stream. Note that *rideId*, *taxiId*, and *driverId* are contained in both streams.

Listing 1 shows an example of a CQL query that combines both streams, counting all the rides over the last hour that had more than 2 passengers and cost more than 10 dollars. The *Istream* operator in the *Select* clause describes that the result of the query will be a new stream containing the new results within the window of 1 hour.

¹ <http://www.debs2015.org/call-grand-challenge.html>

```

1 Select Istream(Count(*))
2 From RideStream [Range 1 Hour Slide 1 Minute]
3 From FareStream [Range 1 Hour Slide 1 Minute]
4 Where RideStream.rideId = FareStream.rideID AND
5     RideStream.passengerCnt > 2 AND
6     FareStream.totalFare > 10

```

■ **Listing 1** An example of an CQL query on the taxi stream.

Carbone et al. [59] studied the field’s maturity concerning processing. Initially, research focused on languages and paradigms for continuous querying and designing Data Stream Management Systems (which extend Data Base Management Systems to support continuous semantics). Later, research moved towards Scalable Stream Processing, motivated by the advent of Big Data challenges. More recently, the authors claim, Stream Processing is moving beyond analytical workloads, welcoming concepts like database transactions, stateful functions, and model serving. Moreover, Stream Processing has been applied beyond continuous queries, addressing tasks such as conformance checking [185], continuous pattern-matching streaming graphs [169, 168], and graph partitioning [1].

2.2 Taming Volume

As highlighted by Carbone et al., the first generation of streaming systems was centred around proving the feasibility of continuous querying and paying little attention to scalability. Hence, the first generation of streaming engines is limited to vertical scaling, e.g., IBM System S, Esper, Oracle CQL/CEP and TIBCO.

Later, due to the introduction of MapReduce and the popularisation of cloud computing, Stream Processing research and development started shifting to the scalability problem. Although velocity (described below) was always the priority, data parallel and distributed solutions became the de facto standard.

In particular, it is worth mentioning Apache Flink [60], which uses a streaming dataflow engine that provides data distribution, communication, and fault tolerance for distributed computations. Apache Flink features two relational APIs – the Table API and SQL—for unified stream and batch processing. Flink’s Streaming SQL support is based on Apache Calcite, which implements the SQL standard. Apache Spark [11] is a versatile distributed computing platform that offers convenient programming interfaces in Java, Scala, Python, and R, along with a well-optimised engine that is capable of handling various execution graphs. At the core of Spark’s abstractions are resilient distributed datasets, which represent collections of elements distributed across nodes within the cluster, enabling parallel data processing. Apache Kafka [231] works as a distributed streaming platform, operating as a cluster on one or more servers called brokers. This cluster can span across multiple data centres. Kafka’s primary role is to store continuous streams of records in what are known as topics, which are essentially unbounded, append-only log structures. Each record within these topics comprises three main components: a key, a value, and a timestamp. A Stream Processing library called Kafka Streams is also built on Apache Kafka’s producer and consumer APIs. It operates on a model known as Stream/Table duality [195].

2.3 Taming Variety

The support for data heterogeneity is limited in general streaming systems. Indeed, Data Stream Management Systems (DSMS) and Complex Event Processing (CEP) engines inherit their data model and query languages from the database community. The seminal work from Babu et Widom [16] poses the basis for relational Stream Processing and influences various languages.

The data models are evolving, with stream processing systems supporting more complex data types inspired by object-oriented programming languages. Indeed, Flink, Spark, Kafka Streams and many more support nested data structures, allowing users to design hierarchies of event types.

Notably, the approach taken from existing DSMSs to address the data variety is rather practical and lacks formal foundations. Data integration is performed through custom data pipelines rather than following information integration principles [140]. Conversely, relational languages have been extended to navigate simple nested structures like JSON. For example, Spark SQL has included operators to manipulate CSV and JSON data since 2017. KSQL and Flink added the opportunity to access nested fields in JSON data within the SQL dialect last year. Nonetheless, data access is managed without source data mapping, making fraternisation somewhat arbitrary and porting queries across systems nearly impossible when semistructured data are involved. Although the notion of an event, as a typed notification of fact at a given time, can be seen as a shared abstraction that can glue DSMS together, few attempts remain in the realm of Stream Processing systems.

It is worth noticing, though, that there are emerging more specialised Stream Processing systems capable of handling more sophisticated data structures such as interval-based events [15], streaming graphs [169], and property graph streams [93].

Orthogonal to the data representation, the Stream Processing literature distinguishes two types of streaming data, i.e., record streams and change data capture. The former indicates positive tuples like sensor network observations, while the latter describes changes within a database (additions and deletions). Although Stream Processing does not typically consider variety in the data model, these two types of streams typically co-exist in the context of streaming systems [195]. Additionally, in the context of system observability, such a dichotomy has evolved into a trichotomy including metrics, logs, and traces, which represents numerical observations, factors or changes, as well as the propagation of information across systems, respectively [199].

2.4 Taming Velocity

Data velocity, i.e., the requirement for processing data as soon as possible and before they are no longer valuable, is the first and foremost priority for Stream Processing research. The velocity challenge has a direct impact on data storage. Indeed, putting data at rest and processing them later is no longer possible, as it would require too much time. In practice, data velocity is treated by operating in memory. Stream Processing Engines, i.e., systems capable of handling data with high throughput and low latency, employ sophisticated mechanisms to reduce the memory footprint without compromising performance.

Their performance is measured alongside two axes, each representing a key performance indicator, i.e., end-to-end latency (the time passed from when a data point enters the system and when it exits as part of the output) and maximum throughput, the amount of data processed within a unit of time, e.g., a second. The two dimensions are in a clear trade-off, pulling the Stream Processing envelope on from two sides, i.e., incremental vs batch computations.

Another substantial change happens in the query model. Queries are no longer issued online but are instead registered and compiled into pipelines, which typically avoid loops for efficiency. As a query can run indefinitely and until explicitly suspended, the result is a stream of answers. The query evaluation occurs upon the arrival of individual data elements or in small (micro) batches. Punctuation mechanisms, i.e., the presence of particular landmarks in the data or the query, are used to progress the execution in a distributed setting. On the data side, punctuation is the minimal informational unit that constitutes a single item in the stream. On the query side, punctuation assumes the role of operators, commonly named windows, that allow the gathering of multiple elements in the stream that should be processed simultaneously. Ultimately, windows can

introduce a delay in different parts of the framework. Modern engines that simultaneously address the velocity and volume challenge may consume millions of events per second, guaranteeing an end-to-end sub-millisecond latency.

2.5 Domain Complexity

In Stream Processing, the complexity of the domain is usually considered relatively limited. Domain modelling is reduced to relational and document data when considering production-graph Stream Processing systems like Flink, Spark Streaming and the Kafka suite. Notably, the presence of a schema, be it relational or document-based, as in the case of binary formats like Avro or JSON Schema, is essential to decouple the production and consumption of streaming data. In terms of conceptual modelling, approaches for event data representation emerged, e.g., event sourcing [35], but only as methods for system integration and without formal semantics [165]. Lastly, in Complex Event Processing, hierarchical data models are often adopted but limited to taxonomical relations inspired by inheritance in object-oriented programming languages [104].

The adoption of Stream Processing systems into application domains that require strong consistency guarantees, e.g., financial analysis or traffic management applications, called for more sophisticated domain modelling techniques. While on the conceptual level, everything remains unchanged, at lower levels of abstraction, the Stream Processing engines require awareness of partitioning schemes, possible faults, and out of order. To this extent, researchers have focused on *consistency* in terms of transactional behaviour [238, 2, 50]. The ACID properties, which are standard in the database context, ensure that the (database) state is consistent to the degree required by a given isolation level. In Stream Processing, the focus shifted to the interaction across systems. Thus, the notion of consistency is discussed in terms of delivery guarantees: *At-least-once* ensures that input data are not lost, *at-most-once* eliminates duplicate processing, and *exactly-once* combines both, ensuring the absence of input data losses and repeated delivery of results [212]. The definition of such guarantees is expressed at the logical level: individual data items are extended with metadata to be used downstream for controlling consumption. Transactional Stream Processing is an ongoing research that is gaining traction at the industrial level².

Last but not least, the role of provenance in Stream Processing represents the most notable attempt to manage additional domain complexity, i.e., reason about the *why* and the *how* of continuous query answers [105]. Vijayakumar et Plale [224] first proposed a low-latency method for generating coarse-grained provenance information that focuses on capturing dependencies between different data streams instead of individual tuples. Wang et al. [232] spot the limitations of techniques based on annotations and suggest a rule-based approach for provenance in Stream Processing applied to the medical domain. However, this approach requires access to all intermediate streams, making it less suitable for modern Stream Processing systems. Glavic et al. [106] proposed a set of instrumented operators to track the provenance of select-project-join queries in Stream Processing scenarios. More recently, the works of Palyvos-Giannas explore richer provenance models, in particular: *Ananke* allows users to track richer provenance information, not only specifying which source tuples contribute to which query results but also whether each source tuple can potentially contribute to future results [171]. *GeneaLog* is similar to *Ananke* but with a focus on the edge [170]. Finally, *Erebus* investigates the aspect of *completeness*, relying on why-provenance [172] for identifying missing answers in the result by explaining the mismatch between actual and expected answers for continuous queries. As such, explaining the inconsistency of continuous queries is not applicable.

² <https://github.com/ververica/streaming-ledger>

2.6 Data Quality

Several factors impact streaming data quality, e.g., noisiness, incompleteness, and timeliness. Stream Processing systems must be able to handle situations where individual data points are missing, entire data streams stop suddenly, or queries are changed. These situations can occur as the result of for example data loss during transmission, changing streaming resources, or changing user/agent needs. This can be regarded as an orchestration problem, where resources are carefully managed to minimise latency and maximise throughput even in the face of changing circumstances.

3 Streaming Linked Data

Throughout the years, the Semantic Web has built and standardised a stack of technologies that enable the vision of publishing, accessing, and processing data on the Web, as if in a database [34]. Among these technologies and standards, IRIs [88] are used to identify resources, RDF provides a data model to describe such resources and their relations in graph-based data structures, ontologies such as RDFS/OWL offer languages to specify schemas (consisting of concepts and the relations between these concepts), and SPARQL provides a declarative query language to execute CRUD operations on RDF graphs (to Create, Read, Update, and Delete resources).

These technologies were built without including time as an intrinsic part of their data model. While it is easy to understand this choice – many systems do not deal with time or delegate its management to the application layer – data evolves, and it is often necessary to address it. Therefore, the community started to build time-aware solutions on top of the Semantic Web stack. For instance, there have been initiatives at the modelling level, such as OWL-Time [65] that allow defining temporal concepts, or the Semantic Sensor Network ontology [64], which provides a vocabulary to describe sensor observations over time. The Semantic Web standards themselves evolved, accounting for time. For example, the RDF 1.1 recommendation [69] states:

The RDF data model is atemporal: RDF graphs are static snapshots of information. [...] RDF graphs can express information about events and temporal aspects of other entities, given appropriate vocabulary terms.

In practice, this implies that time information can be included within an RDF graph, without time-specific semantics. In addition to use cases where it is necessary to account for time, a second need emerged: responsiveness. An increasing number of applications require not only managing temporal data, but also timely processing of results. These requirements are frequent in a large number of domains including social media analytics on the Web, or data management for the Web of Things (WoT). In these applications, it is vital to provide instantaneous query and analysis results, for which time order and recency play a crucial role.

These needs led to a novel research area within the Semantic Web community, under the denominations of *RDF Stream Processing* (RSP) or *Streaming Linked Data* (SLD) [217]. RSP research has focused more on the temporal extensions for RDF data and query modelling, while SLD has centred on the implications of Stream Processing for graphs that comply with the Linked Data principles [41, 46]. Beyond these minor differences, this line of research has delineated an agenda that has studied the following aspects:

- Modelling data streams and complex events using RDF graphs, including syntactic, semantic, and operational implications.
- Extending RDF query languages with streaming data operators.
- Building RDF stream Continuous Query processors, including different reasoning and processing variants.

2:10 Grounding Stream Reasoning Research

- Evaluating and benchmarking Stream Processing engines, including performance, and correctness, among other metrics.
- Interconnecting RDF stream processors through Web interfaces.

This allows us to reformulate the macro-level research question to the following specific meso-level research question for RSP/SLD:

Meso (Streaming Linked Data): Can we evaluate Continuous Queries, expressed as a dialect of the SPARQL language, over RDF streams with limited latency while incorporating domain knowledge through RDFS ontologies?

This and other subsequent research questions have been explored and discussed, many of which converged around the RDF Stream Processing Community Group (RSP CG), within the context of the W3C³. This group served as a central discussion square that led to different formalisation, implementation, and benchmarking initiatives in this area.

RSP/SLD has been successfully used in a variety of use cases, ranging from social media analytics [20], traffic monitoring in Smart Cities [139], large-scale streaming data retrieval in Smart Farming [124], to monitoring the performance of athletes [158] and the health of patients [71].

We will now explain how RSP/SLD research has targeted different aspects of the original Stream Reasoning macro-level research question.

3.1 Make Sense

In order to process RDF streams, it was observed that Semantic Web technologies and Stream Processing technologies are complementary for solving the problems that Stream Reasoning tries to tackle. In terms of **making sense** of the data, RSP and SLD are fundamentally based on *continuous querying* and *data integration* approaches. The former, takes the idea from SP, where queries are registered only once and continuously produce results as they are evaluated over streams of data. Moreover, RSP and SLD inherit the data integration capabilities from the Semantic Web, as they seamlessly integrate Stream Processing and Semantic Web technologies. Through the use of ontology models to represent the stream data elements, these query languages were able to integrate different data sources, including both static and streaming data.

Over the years, several languages have been proposed. Most of them aim to process extensions of RDF where triples or graphs are annotated with temporal information, such as individual timestamps or time intervals. Examples of these languages include C-SPARQL [24], Streaming SPARQL [43], CQELS-QL [134], or SPARQL-Stream [55]. Most of these languages extend the SPARQL syntax with time-based sliding window operators, as found in Stream Processing; and in some cases, additional query functions. The semantics of how these windows work, however, were not uniform and were shown to have different operational behaviours. In consequence, these languages disagreed on the correctness of query results in certain cases [81], as they had different properties that made them difficult to compare,

To address this issue, a unifying formalisation of continuous query processing over RDF streams was proposed in [78], named RSP-QL. This model was able to include streaming query evaluation semantics, as well as operational semantics of windows, thus allowing to characterise existing extensions of SPARQL for continuous querying. The ability to represent different types of queries using RSP-QL is a first step towards the standardisation of continuous querying extensions for

³ W3C RSP Community Group: <https://www.w3.org/community/rsp/>.

RDF streams. However, RSP-QL still inherits practical inconveniences from SPARQL, such as the difficulty of generating and re-consume RDF stream results (e.g., through using the SPARQL `CONSTRUCT` clause).

There are more operators beyond the sliding window operator. Examples include basic CEP sequence matching, which was integrated into RSP-QL through RSEP-QL [82], and monotonicity conditions, which were proposed in STARQL [166]. Nevertheless, these elements add substantial complexity to the formalisation and eventually to the implementation of querying engines, as we will see next.

3.2 Taming Volume

Although most of the contributions in the SLD research area have focused on addressing the inherent velocity of data streams, taming volume has not been thoroughly investigated. There have been some efforts, such as CQELS-Cloud [135] and Strider [189] that build upon the elasticity of existing Stream Processing frameworks, respectively Apache Storm and Apache Spark. However, the focus of taming huge volumes of data has been rather limited.

Nevertheless, this dimension has indirectly been addressed through the analysis of query execution efficiency and response time constraints. Velocity can be analysed in terms of volume over time, which was analysed in RSP benchmarking efforts [239, 133]. Among the works specifically targeting stream data volume we can mention efforts for reducing the actual size of serialised RDF streams, using the compressed ERI interchange format [94]. The usage of reduced formats for RDF stream data exchange are of primary importance for IoT environments where message volume is critical [122], such as constrained devices, limited network bandwidth, and reduced storage sensors.

Other approaches addressed volume from the processing perspective, for instance proposing load-shedding techniques to limit the number of stream data items to be processed [33], or data eviction strategies to reduce the cardinality in join operations over RDF streams [100].

3.3 Taming Variety

RDF graphs allow modelling all sorts of information on the Web, enabling wide exchange and interoperability. However, these graphs are atemporal, and RSP needs an adequate data model to publish and exchange data streams while handling data variety. *RDF streams* address this challenge by extending the RDF model with notions of time-based order. The initial attempts to define RDF streams were crystallised by the RSP CG, which proposed the following requirements for the abstract model of RDF streams [7]:

- R1** It should be possible to represent RDF streams with an abstract RDF-based model, whose semantics should provide the basis for producing and consuming streams.
- R2** It should be possible to identify an RDF stream using IRIs.
- R3** It should be possible to serialise the RDF stream abstract model into RDF formats derived from existing standards, extending them only when necessary.
- R4** It should be possible for RDF Stream to have timestamps based on different notions of time (time instants, intervals) with different semantics (application, validity, transactional).
- R5** In case no timestamp is associated with an RDF stream data element, the system should be responsible for managing the time-based ordering of stream elements.
- R6** It should be possible to restrict the RDF stream model to facilitate implementation and support efficient representation.

```

1 prefixes:
2 taxi: "http://linkeddata.stream/ontologies/taxi#"
3
4 mappings:
5 rides:
6   s: taxi:ride/${rideId}
7   po:
8     - [a, taxi:RideEvent]
9     - [taxi:hasEndLon, ${endLon}]
10    - [taxi:hasEndLat, ${endLat}]
11    ...

```

■ **Listing 2** An example RML Mapping on the taxi dataset.

The above requirements call for reusing existing Semantic Web technologies and account for different types of temporal information. One can consider an RDF stream as a sequence of RDF graphs, each identified by an IRI and optionally associated with temporal annotations, such as creation time or validity interval. The generality of the RDF streams definition aims at opening the door to different kinds of streams, which may occur in different application scenarios. However, it also implies the challenge of dealing with the complexity of covering modelling variations. This specification led to the implementation of systems capable of producing streams of RDF data [24, 136, 216]. At the simplest level, plain RDF can be used to represent streaming information without specific semantics for time annotations. For example, the Linked Sensor Data [175] initiative proposed the publication of meteorological sensor data using stored RDF. Although these RDF graphs represent observations that were originally streamed by sensors, with explicitly recorded time annotations, the system only provided static access to the data. RDF libraries such as Jena⁴, RDF4J⁵, or RDFLib⁶ provide IO methods to read and write plain RDF graphs in a streaming fashion, but they do not support producing and consuming RDF streams.

TripleWave [150] is one of the systems that addressed this limitation, proposing a full pipeline for the generation of RDF streams. It included the production of live RDF streams consisting of time-annotated graphs, which could be fed from non-RDF data sources.

More recently, RMLStreamer was introduced, focusing on the generation of RDF streams in a low latency and high throughput fashion. RMLStreamer is a parallel and scalable Stream Processing engine built on Apache Flink that is able to generate RDF streams from heterogeneous data streams of any format (e.g., JSON, CSV, XML, etc.), using RML mappings [83].

► **Example 2** (Taxi cont'd). SLD allows to integrate the taxi streams with additional static information, e.g., a dataset that describes Points of Interest (POI) within the city. The use of SLD enables this integration, even though the underlying data representation of the POI dataset is not compatible with the raw taxi streams. Mapping the taxi streams and POI dataset to RDF allows to smoothly integrate both datasets, allowing to make more informed decisions regarding the available data. The taxi streams can be mapped to RDF in a streaming fashion through the RMLStreamer. Listing 2 shows how this mapping can be defined in YARRML [114], i.e., a more concise RML syntax. The mapping defines how various fields of the taxi dataset, e.g., *rideId*, *endLon* and *endLat*, can be converted to RDF triples. A similar mapping can be conducted for the POI dataset, regardless of its underlying data format.

The example RSP-QL query in Listing 3 counts all taxi drop offs near a hospital in the last hour, which has been made possible through the integration of the POI dataset. Joins in RSP-QL can be applied to a combination of both windows and stored graphs as seen in the example.

⁴ <https://jena.apache.org/>

⁵ <https://rdf4j.org/>

⁶ <https://rdflib.dev/>

RSP and SLD are thus able to tame data stream variety by reusing and extending Semantic Web technologies, by introducing fundamental temporal semantics into the data model, and by providing tools that implement them.

3.4 Taming Velocity

Each query language designed to process RDF streams was accompanied by working prototypes, also called RSP engines. The first contributions investigated how Semantic Web technologies can be combined with DSMS and CEP engines in order to incorporate Stream Processing capabilities that could target velocity. These systems propose different approaches to continuous query answering over data streams, shifting from the query-response paradigm of conventional SPARQL engines. In addition, these systems have a special focus on reactive question answering, proposing methods that allow for delivering results as soon as possible. Each of these systems incorporates variations of windowing implementations, in order to limit the possibly unbounded stream in processable chunks.

Among the first generation of RSP engines, C-SPARQL [24] adapts a black box approach by pipelining a DSMS with a SPARQL engine. The DSMS is used for handling the Stream Processing capabilities of the engine, e.g., windowing the stream into processable chunks. Each window is then fed to the SPARQL engine for evaluation of the query. In contrast, the CQELS engine [134] employs a white box approach; instead of pipelining existing systems, it integrates the Stream Processing operators in the evaluation of the SPARQL query, opening up various opportunities for optimisation. Morph-streams [56] takes a different approach and uses Ontology Based Data Access (OBDA, or Virtual Knowledge Graphs) to virtually process RDF streams, while their underlying representation is still the raw data (e.g., a relational data stream, or a streaming CSV). It uses a mapping language, i.e., R2RML⁷, to define the relation between the underlying relational data and RDF. Morph-streams uses query rewriting to virtually answer SPARQL-like queries over relational data streams, giving the illusion data is available in RDF.

Other approaches focused on extending existing infrastructures for distributed data processing, such as the aforementioned Strider (see Section 3.2). Regarding the integration and interoperability of RSP engines, RSP4J [216] proposed an API for the development of RSP engines under RSP-QL semantics, providing many of the needed abstractions and interfaces that can be used for building blocks when creating new RSP engines or testing out algorithms and optimisations. RSP4J

⁷ <https://www.w3.org/TR/r2rml/>

```

1 PREFIX taxi: <http://linkeddata.stream/ontologies/taxi#>
2 PREFIX : <http://linkeddata.stream/resource/>
3 SELECT (COUNT(?d) AS ?num_hospitalDropOff)
4 FROM NAMED <citymap.rdf>
5 FROM NAMED WINDOW <w> ON :taxiStream [RANGE PT1H STEP PT5M]
6 WHERE {
7     Graph <citymap.rdf> {?place :hasLat ?lat; :hasLon ?lon; :hasPOI ?poi.
8         ?poi a Hospital. }
9     WINDOW <w> { ?d a taxi:DropOffEvent; taxi:hasEndLon ?lon; taxi:hasEndLat ?lat. }
10 }

```

■ **Listing 3** An example of an RSP-QL query on the taxi stream.

also provides two implementations, Yasper and CSPARQL2.0, that follow the RSP4J interfaces. Following the principles of RSP4J, but written in Rust, RoXi [44] brings RSP engines to the browser through WebAssembly support.

There have been a number of contributions centred on the evaluation of RSP engines. As for Stream Processing solutions, the principal metrics are latency (the time required to process a stream) and throughput (the amount of data processed in a given amount of time). Further metrics include memory footprints, expressiveness (which query operators are supported), and correctness (compliance of a system to its evaluation semantics). Among the proposed benchmarks, LSBench [239] and SRBench [133] proposed re-playable data streams, evaluation queries and a set of metrics to assess the performance and expressiveness of the engines. The YABench [128] framework proposed a more comprehensive coverage of RSP features, while Citybench [4] proposed more realistic and configurable testing datasets. Finally, RSPLab [219] focused on the provision of an open-source environment for RSP reproducibility.

3.5 Domain Complexity

In RSP and SLD, the incorporation of complex domain modelling is usually satisfied by using RDF and RDFS ontologies. In general, the domain complexity in RSP is kept low in order to realise highly reactive systems, given the potential latency that reasoning can add to the query processing stack. Nevertheless, there are some hybrid approaches where RSP and reasoning overlap, for instance, incorporating query rewriting through ontology-based data access or materialising window content and enabling Datalog reasoning. Some of these hybrid approaches are further described in Section 6. When increased domain complexity and expressivity are needed, a sacrifice in latency and throughput is acceptable. To the opposite extreme of this trade-off, we enter the realm of Deductive Stream Reasoning (Section 4), which privileges domain complexity in a dynamic environment.

3.6 Data Quality

Handling *veracity* and *incompleteness* has so far not received much attention within RSP, given that in many cases, the RDF streams are previously pre-processed or fed through streaming pipelines that already perform minimal data cleansing (e.g., through Kafka pipelines [126]). Otherwise, stream data quality control is seldom incorporated into RSP engines. In some cases, Continuous Queries filter out anomalous data, or external data mining and outlier detection modules are employed before the RSP engine receives the stream. When dealing with *constantly-varying data*, Strider and CQELS provide optimisations to reorder the execution of their query execution plans based on the rates of the various streams that are being processed. When the rates of the streams change, the execution plans are reordered to maintain reactivity. The quality of Continuous Query results may sometimes degrade when the stream rate rises. In consequence, it can be helpful in use load-shedding and similar techniques to limit the number of stream items to be consumed [33].

Finally, quality can also be considered regarding the correctness of the Continuous Query processor. In the case of RSP engines, this topic was addressed in [81], which verified that seemingly similar queries resulted in different answers, in some cases not entirely predictable. Based on these results, the operational semantics of RSP query languages have been further studied [78], and other benchmarking frameworks have adopted correctness criteria for their test suites [219].

4 Deductive Stream Reasoning

The contributions presented in the previous sections assume that all the knowledge is stated *explicitly* in the data streams. In some cases, however, there is a wealth of *implicit* knowledge that can be *inferred* with some non-trivial computation.

We refer to systems that do this by evaluating logic-based statements in a deductive manner as *Deductive Stream Reasoners (DSRs)*. Inductive Stream Reasoning, which aims at inferring new knowledge from data, will be considered in the next section.

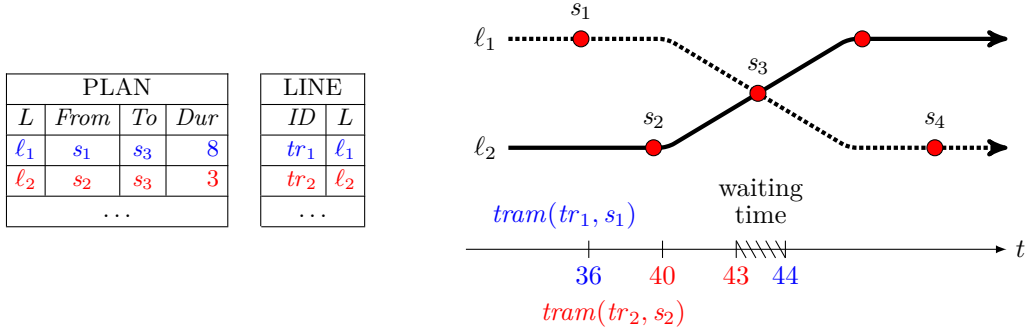


Figure 1 Transportation example.

► **Example 3** (Vienna tram connection). Staying in the transport sector, let us exemplify the general notions of DSRs on the following navigation problem, this time considering public transport instead of taxis: Samantha is travelling in Vienna with her baby and a stroller on a tram ℓ_2 from s_2 to s_4 , which is served by the line ℓ_1 . Thus, Samantha needs to change the line on the stop s_3 .

According to the plan, shown in Figure 1, a vehicle tr_1 that serves the line ℓ_2 requires 3 minutes to reach s_3 from s_2 and a vehicle tr_1 needs 8 minutes to get from s_1 to s_3 . A transportation application that Samantha is using must solve at least the following two problems: (i) get information about the current schedule and delays of trams running on ℓ_1 and (ii) find expected good connections between s_2 to s_4 with less than 5 minutes waiting time at s_3 .

An application based on a DSR gets its knowledge about the transportation problem explicitly, i.e., an expert provides it as a knowledge base KB , such as an ontology or a logic program. DSR then uses KB to solve various problems, e.g., to find suitable routes or inform users about expected arrivals. A data stream comprising information about the current state of the transportation system is pushed to the DSR from sensors and other systems. DSR systems can represent these streams in two possible ways: point-wise or interval-wise. In the *point-wise* representation DSR discretises the time into a set of time point, e.g., a second or a minute, depending on the system architecture. The encoding of data might also vary. Thus, many rule-based DSR systems require incoming data to be encoded as facts, which are associated with time points when they were received, e.g., $36 \rightarrow \{tram(tr_1, s_1)\}$. Other popular representations include database tables, RDF triples, and labelled values in a similar way as the atoms above. The *interval-wise* representation appears to be more natural since time discretisation is not required as in the point-wise case. Hence, a DSR system might associate a set of intervals with every data value appearing in the stream. The main caveat of this representation is that it requires DSR systems to determine the end of each interval. Thus, if a movement sensor reports only changes in tram velocity between the stations, the system cannot determine if the tram is still moving at a constant speed or if the sensor is malfunctioning.

Given the background knowledge about timetables and a stream comprising facts about the positions of trams on their lines, the application needs to retrieve data relevant to Samantha's situation. Most DSR systems use various kinds of *window functions* to retrieve relevant parts of the stream, similar to the windowing introduced in Stream Processing. In our transportation example, a time-based window for the interval $[35, 45]$ will return $\{tram(tr_1, s_1), tram(tr_2, s_2)\}$, and a tuple-based window of size 1 from $t = 45$ will return only $\{tram(tr_2, s_2)\}$.

In general, DSRs are useful in complex domains where applications should be able to continuously make decisions using knowledge explicitly provided by experts. That is, in contrast to the inductive systems, it is not realistic to expect that knowledge required for decision-making is provided in the stream data, like observations and/or labels. Such domains include Cyber-Physical Systems (CPS), Digitalization of Industry, Internet of Things (IoT), and Social Networks. Examples of such applications are

- *monitoring and surveillance*, e.g., of gas turbines [52], maritime vessels [194], or healthcare [111];
- *decision making*, e.g., for video streaming and games [28, 6];
- *planning*, e.g., trajectory planning for UAVs [112];
- *analysis and query answering*, e.g., in social networks [25], smart infrastructures [178], and in intelligent transportation systems [89, 197].

The macro-level SR research question can be reformulated to the following generic meso question for DSR:

Meso (Deductive Stream Reasoning): How can we make knowledge about complex domains, represented in expressive Knowledge Representation languages, available for Stream Reasoning in the realm of vast, rapidly evolving, constantly varying, inevitably noisy, incomplete and heterogeneous data streams?

This generic question gives rise to several concrete meso-level questions that need attention:

1. How can we reuse previously inferred knowledge to minimise the reasoning time upon receiving updates respectively changes in data?
2. How can we extend existing Knowledge Representation languages suitably with temporal operators?
3. How can we achieve a balance between the expressiveness of the Knowledge Representation and the efficiency of reasoning in particular for maintaining a high throughput?
4. How to deal with noise and uncertainty appropriately in expressive Knowledge Representation formalisms, both regarding the quality of results and performance?

Solutions to these questions will be instrumental for achieving the macro goal of Stream Reasoning from above, as rich Knowledge Representation formalisms allow us to express and reason about properties and relationships between data at a deeper level. They enable us to obtain more insight transparently, and provide a basis for developing explanation and justification facilities that will aid in analytics and increase transparency, and hence, trustworthiness.

The first question is at the heart of Stream Reasoning, and requires to face the challenge that conclusions may be obtained by reasoning processes that involve several steps of inferences, depending on the complexity of the underlying Knowledge Representation. *Materialisation*, i.e., computing and storing the valuation of predicates/relations that are defined from given data, and related techniques play an essential role here [156, 225, 157, 221, 177, 118, 222]. *Data parallelisation*, i.e., enabling parallelism in reasoning by data partitioning, has also been considered and investigated as a possible way to tackle this issue [180, 179]. However, for expressive Knowledge Representation languages, incremental evaluation under frequently changing data is not at a level of performance as one would desire in real-time applications like traffic monitoring.

The second question has led to several works in which (static) Knowledge Representation formalisms have been extended with operators and constructs from temporal logics and reasoning, e.g., [86, 112, 52, 29, 226, 58]. However, they are quite diverse and it is at this point open whether the requirements of Stream Reasoning are well covered and which selection and combination of operators would be beneficial, and whether novel operators should be introduced.

The third question is important as, intuitively, constructs in a language that allows for expressing more involving relationships (e.g., joins with negation, nested relations, or recursion) require more computational resources for evaluation [102]. However, even comparatively high resources may not empower one to compute answers over varying inputs, as well-known from descriptive complexity theory [121] and researched extensively in databases and knowledge representation. In DSR, this question—in particular with an eye on high throughput—has been not been much explored yet.

The fourth question arises as commonly declarative languages based on logic assume a well-behaved environment in which data is consistent and uncertainty, if at all, is limited to missing or indefinite information. In the streaming context, this calls for extensions of DSR formalisms that can deal with inconsistent data, outliers, and quantitative uncertainty, especially with probabilistic information. This has been addressed in several works, e.g., [161, 214, 51, 90, 181, 74, 215], but there is no gence nor uniform approach to serve this need, and performance guarantees are an issue. In general, blending uncertainty with logic is a popular topic of interest in AI, with many ongoing works in several communities. In a streaming context, we identify two main research avenues. The first is studying whether existing techniques, in particular those that use deep learning architectures in static contexts, can be successfully adapted so that they can work in a streaming context. The second avenue consists of designing novel techniques specifically for streaming scenarios. This type of combination will be discussed in more detail in Section 5 dedicated on inductive Stream Reasoning.

We conclude by emphasising the critical importance of establishing comprehensive metrics and clear evaluation criteria for assessing contributions to the aforementioned research questions. This is a problem that has been receiving considerable attention in the community (see, e.g., [196]), especially for the following reasons:

- If two solutions implement two different formalisms, then it can be that it is precisely the differences between the two which are responsible for a certain increase/decrease of performance. Thus, it is hard to distinguish the value of a certain solution;
- If we adopt absolute metrics, like runtime or memory consumption, then it becomes arguable when a solution is “good enough” since small variations in the use case can lead to a completely different outcome;
- It is also difficult (or even impossible) to determine which are the most important without resorting to concrete use cases.

As previously mentioned, several RSP benchmarking efforts were developed [239, 133]. These platforms require a graph-based data model and are tailored towards benchmarking query answering, hence are well suited for OWL-based languages. For instance in [57], the authors showed that queries with an OWL2 QL-based engine could be answered up to a throughput of 200K triples/s. Since the mentioned efforts do not cover more challenging reasoning tasks and program sizes, some researchers rely on artificial micro-benchmarks to conduct the experiments and to report empirical evaluations. For instance in [27], LARS-based implementations were compared among themselves and against RSP engines featuring that a response time below 100ms can be achieved for multi-rule programs with a throughput of 800 triples/s. It is likely that a more widespread adoption of DSRs in the real world will guide the research community in choosing more meaningful evaluation criteria.

4.1 Make Sense

A central problem for DSR systems is to promptly answer the question “*What is true now?*”, which has been a widely-studied problem in Knowledge Representation since the inception of the field [96]. The number of contributions made in this area is so high that it is not possible to present a concise summary without running the risk of missing out on some important work. Therefore, we will limit ourselves to pointing the reader to some encyclopedic texts [153, 95, 85, 116, 96] and focus instead on the most recent works that are closely connected to the ones in the other sections.

First of all, let us define a DSR as a system that receives as input a data stream and possibly some background knowledge, either in the form of facts or more complex expressions like rules. The system aims to process the data stream to infer new conclusions using a deductive logic-based process. The computation is specified in a *declarative* manner, that is, we tell the system *what* to compute and let it decide *how to do it*. Typically, it is expected to yield the answers to a given query. For instance, a DSR may receive as input a query in the form of a set of rules and use those to compute the answers.

Since the deductive process is based on logic, DSRs require that the input (stream, query, background knowledge, etc.) is expressed with a formal language. Different such languages have been proposed, based on temporal logic as in the DyKnow framework [112], on extensions of description logics as in SPARQLstream [57] and STARQL [167], or on logical rules as in the popular LARS [29] and DatalogMTL [52, 227] formalisms. The first is grounded on Answer Set Programming (ASP) [53], one of the most well-known languages in the Knowledge Representation community while the other is grounded on Datalog [61], another established formalism in the community. These two languages define the semantics (*what does it mean to answer a query?*) and the supportive expressive power (*what kind of queries can we write?*) in a formal and unambiguous way. In general, we observe here a trade-off that is common with logic-based reasoning: the higher the expressive power is, the more challenging the computation becomes, to the point it is no longer feasible. This trade-off has motivated the design of formalisms, like LARS and DatalogMTL, that have computational bounds that meet the demands of streaming scenarios.

4.2 Taming Volume

First of all, it is essential to mention that while some approaches assume that the stream is infinite (e.g., DatalogMTL [227]), others (e.g., LARS [29]) assume that there is a time point in the future when the stream ends, respectively data beyond it will be ignored. Of course, from a practical point of view, we can set the time when the stream ends to a point which is very far in the future to simulate the case of an infinite stream. From a more formal point of view, however, the assumption that the stream is finite has essential consequences related to the decidability of some critical operations like query answering.

In this context, a *data stream* is often viewed as an ordered collection of timestamped *facts*, e.g., in Example 3 it consists of $tram(36, tr_1, s_1)$ and $tram(40, tr_2, s_2)$. The facts become available as time passes by, which means that the system does not have immediate access to all the data. The data stream is augmented with timestamped atoms that are derived, which in Example 3 may be $exp(44, tr_1, s_3)$ and $exp(43, tr_2, s_3)$ for the projection of the expected arrival times of tram tr_1 and tr_2 , respectively, at stop s_3 . Since we are often interested in obtaining answers immediately, the system must continuously re-evaluate the input queries as new data becomes available. Clearly, to support large volumes of data, it is more efficient to reuse all the inferences previously derived instead of re-computing them from scratch. In a static setting, this problem has been widely studied and is commonly referred to as “incremental reasoning” or “knowledge base maintenance”. Indeed, some of the techniques used for incremental reasoning can be adapted to work on data

streams. For instance, a well-known technique developed for Datalog is *semi-naïve evaluation* [21], which prevents a rule instantiation being evaluated more than once. This technique has been adapted, with some modifications, to work with data streams [27]. Other techniques include multi-shot solving [164], overgrounding of rules [119], and truth-maintenance methods [30] for ASP based stream reasoners.

4.3 Taming Variety

Streams may originate from various sources, such as sensors with different modalities, but also as output of processing components in a system. This naturally leads to a variety of data formats that would need to be accommodated. However, DSR has so far not put much emphasis on heterogeneous data streams, and the systems and approaches available focus on a specific data format. Specifically, as mentioned above symbolic streams are commonly represented as collections of ground atoms that represent any input; the proper treatment and reconstruction of the meaning of the data lies with the stream queries using them. While plain, this approach akin to data models in relational databases still allows for embracing a number of data domains. In some cases like e.g., for RDF, mapping data into logical atoms while preserving the meaning is rather easy, while for richer data formats, such as (part of) a knowledge graph or graph data generated by a camera describing a scene and how it is evolving may be more demanding; *flattening*, i.e., converting structured to plain relation data may serve here as a key technique and predefined data schemes of fixed structure can be used to ease the meaning reconstruction for query answering.

4.4 Taming Velocity

In order to provide responses that are still valuable and not outdated, limiting the data to snapshots is a common approach, in which merely data available at some specific time point is considered. By doing so, one is taking into account that the answer may possibly diverge from the one when the evaluation would happen over the whole stream. *Windowing* is an essential notion in this context, shared among the various approaches, which can be defined as input or dynamically computed. In the first case, the user decides for how long in the past (or in the future) the system is allowed to consider input data. This can be done through *time-, tuple- or partition-based windows*, similar to the windowing functionality in Stream Processing and SLD. In the second case, a reasoner may infer automatically when some data in the past (or in the future) should be ignored. An important aspect in both cases is whether forgetting respectively ignoring data will affect the reasoning outcome; clearly one desires (or may even request) that this is not the case. Unfortunately, the deductive setting comes with computational obstacles: for temporal Datalog, which is a core rule language for temporal reasoning, it is in general undecidable whether forgetting data using finite sliding windows is possible without loss of inferences, as well as to recognize suitable sizes of such windows [191]. Thus, either a (deliberate) loss of inferences is accepted or restrictions on the programs and/or assumptions on the data have to be adopted. In frameworks like LARS, windows can be nested, which seems to occur less frequently in practice. In addition, time points may be abstracted in a window, such that data occurrence *somewhere* (i.e., at some specific point in time) or, dually, *everywhere* (i.e., at all time points) in the window is considered; e.g., DatalogMTL [52, 227] and LARS [29] offer this feature. The language of the i-dlv-sr stream reasoner [58], which leverages on Apache Flink and the incremental ASP solver i²-dlv [119], supports moreover non-contiguous windows that may be time- or tuple-based; however, the rules of a program must use stratified negation, i.e., negation can be evaluated in a layered fashion.

4.5 Domain Complexity

This research dimension is inherently tied to a selected domain language family and the reasoning task at hand, which leads to a wide range of approaches, mainly covered by the fields of Semantic Web technologies and Logic Programming, which adhere to different views of how the world of interest is modelled; this in particular concerns incompleteness of data, which will be addressed in Section 4.6 below.

Temporal Logic. As stream reasoning involves time, temporal logic is a natural basis for DSR. Linear time logic (LTL) [182] is perhaps the most prominent temporal logic. Besides Boolean connectives, temporal operators are available that allow for expressing statements $\mathbf{X}\phi$ and $\phi \mathbf{U}\psi$, which informally mean that ϕ holds in the next stage resp. that ϕ holds always until ψ holds at some stage; $\mathbf{F}\phi$ and $\mathbf{G}\phi$ are shortcuts where $\phi = \top$ (truth) and $\psi = \perp$ (falsity). respectively, meaning that ψ holds at some stage resp. that ϕ always holds. Formulas are evaluated over infinite paths $s_0, s_1, \dots, s_i, \dots$ in a Kripke structure, which intuitively is a transition graph over truth assignments to a set of propositional atoms; this provides a natural link to (infinite) streams. For example, the formula $\phi_1 = \mathbf{G}g \rightarrow \mathbf{F}r$ intuitively expresses that whenever a request (r) is made, it will be granted (g) instantly or at some later stage, while $\phi_2 = \mathbf{G}g \rightarrow \mathbf{X}r$ expresses that whenever a request (r) is made, it will be granted (g) in the next stage; the formula $\phi_3 = \neg g \mathbf{U}r$ intuitively says that no grant occurs prior to the first request. On the infinite path $\emptyset, \emptyset, \{r\}, \emptyset, \{g\}, \{r\}, \{g\}, \emptyset^\omega$, where each set are the atoms assigned true at the respective state, formula ϕ_1 evaluates to true, while ϕ_2 evaluates to false: r is true at stage 2, while g is false at stage 3. The formula ϕ_3 evaluates to true on this path, since r is true at stage 2 and g is false at stages 0 and 1.

Beyond a simple ordinal timeline of consecutive stages $0, 1, 2, \dots$, metric temporal logic (MTL) [130] and variants are considered in DSR in which \mathbf{G} and \mathbf{F} are relative to an interval $I = [a, b]$, written \boxplus_I resp. \boxminus_I , such that $\boxplus_I\phi$ (resp. \boxminus_I) is true at time t in a path, if ϕ is true at every (some) time t' where $t+a \leq t' \leq t+b$. This in particular allows for modelling data snapshots respectively windows as described above, where only part of the stream data is considered for evaluation.

Relational Domains. In a plain relational setting, a domain is similar as with relational databases more or less given by a list of elementary predicates, and any relationships among them have to be expressed by statements in the program or theory for Stream Reasoning.

MTL is used for Stream Reasoning in planning and execution monitoring is [86], which is part of the DyKnow framework [112]. The latter streams data to a monitor which continuously evaluates formulas over them. E.g.,

$$\boxplus((\neg onroad(car1) \vee slow(car1)) \rightarrow \boxminus_{[0,30]}(\boxplus_{[0,10]}onroad(car1) \wedge travel_speed(car1)))$$

may express that if $car1$ is off-road or at slow speed, it will within 30 secs be for at least 10 secs on the road at travel speed. The stream is incrementally incorporated into the formulas by means of the *progression* syntactic rewriting process [17]; this also enables runtime verification.

Rule-based languages can be used to capture complex domains where we distinguish between Prolog-, Datalog-, and ASP-based languages that share rules of the form:

$$a_0 \leftarrow a_1, \dots, a_n, not\ a_{n+1}, \dots, not\ a_m$$

where the a_i 's are first-order atoms and *not* is negation-as-failure (aka default negation).

Pure Prolog was used for implementing real-time complex event detection, such as shown in ETALIS [8] and RTEC [14], as one of its strengths is efficient list processing. Prolog was also more tailored for “native” Stream Processing by lazy evaluation techniques [173] and stream

transducers [174]. The first “streamed Datalog” language is Streamlog [236], which uses the notion of progressive closing world assumption (PCWA) to deal with stream data, considered in Section 4.6 below. Recursive queries further extended the initial language and aggregates in [70]. A different approach was pursued with DatalogMTL extensions [52, 226, 227] allowing for MTL operators in rules that are evaluated over a dense timeline. DatalogMTL was subsequently extended with stratified negation in rules [66] and recently with stable semantics for unstratified rules [229, 228].

Answer Set Programming is well-suited for reasoning tasks that require to model and solve NP-hard search problems. ASP evolved for Stream Reasoning on the level of modelling/language features with StreamRule [152], C-ASP [178] and LARS [29], where all languages introduce various window operators and the latter lifted answer sets to answer streams inheriting their properties (e.g., minimality) and allowing for LTL operators (without next nor until) to be evaluated over a window. LARS was later extended to model quantitative extensions in stream reasoning [90], while StreamRule was later extended to cater for uncertainty [162, 163]. On the level of processing features, the multi-shot solving feature of Clingo facilitated the continuous evaluation of changing logic programs [101]. Fragments of the LARS language were implemented in Ticker [30] and Laser [27], where the later is geared for high throughput on large data volumes with the restriction to stratified programs. Distributed evaluation of LARS programs was introduced in [92], where a program can be decomposed and evaluated by several engines using an interval-based semantics.

Ontologies. Different from the rule-based formalisms above, other DSR languages leverage an ontology of the domain that is given in a customary language, such as the RDF(S) and the OWL2 standard. A basis for temporal reasoning in Description Logics (DL), on which several languages of OWL2 are based, was given in [12], which extended DL with LTL, allowing for temporal operators in DL axioms, with two-sorted semantics for objects and the temporal domain. Furthermore, temporal query answering was investigated, e.g., in [13, 49], where query rewriting over DL-Lite ontologies was extended for LTL operators in queries. A direct extension for RDF(S) ontologies as used in RDF Stream Processing are OWL-based ontology languages such as OWL2 RL, OWL2 QL or OWL2 DL [110]. In particular, OWL2 QL is well-suited for Stream Reasoning since it is first-order-rewritable and can be evaluated on a streaming database system (DBS), i.e., a data management system geared to store and process an incoming data stream in real time. SPARQLstream [57], STARQL framework [167], and the work of [89] allow for query rewriting over a streaming DBS, where the ontology is rewritten into the query that supports window operators. OWL2 RL reasoning that comprises also recursive rules is supported by RDFox [160], where the combination of a main-memory DBS and incremental update enable the use in a stream reasoning setting. An approach that supports more expressive ontologies is TrOWL [213] where the combination of incremental reasoning and with semantic and syntactic approximation of OWL2-DL by OWL2-QL and of OWL2 by OWL2-EL allows for query answering and classification, respectively, over streams of ontologies.

Stochastic Domains. In real-world domains, dealing with quantified uncertainty is an important aspect, which has been addressed in several extensions of DSR languages. PrASP [161] is a probabilistic extension of ASP, which offers probabilistic annotations of formulas, including facts and rules, which induce a possible worlds semantics. LARS has been extended to model quantitative extensions in stream reasoning [90]. Among them is probabilistic reasoning, which has been demonstrated for object tracking in [181]. P-MTL [214] and ProbSTL [215] are probabilistic extensions of MTL and STL respectively, which allow for incremental runtime verification with explicit constraints over deterministic observations and uncertain predictions inside the logic itself. Notably, ProbSTL can express confidence in predictive capabilities by comparing past predictions of the present state with estimations of the current state.

In conclusion, various domain complexities are supported in Deductive Stream Reasoning, ranging from low language complexity as with OWL2 QL and positive Datalog programs, to ASP-related languages, DatalogMTL, and TrOWL, which offer the highest degree of language expressiveness on the level of program structure, temporal operators, and ontology model, respectively. Furthermore, some support of probabilistic reasoning is available.

4.6 Data Quality

Veracity. In approaches based on a crisp 2-valued logical semantics, the data in a stream is expected to be pre-processed and is assumed to be verified in a credible manner. However, facts might still be checked for inconsistencies with respect to the domain knowledge using constraints in rule-based languages, or disjointness in ontology-based languages. In [51], the authors suggested temporal query answering in OWL2 QL over inconsistent data streams with three inconsistency-tolerant semantics designed to automatically repair inconsistencies, e.g., a brave semantics in respect to rigid concepts/roles. If veracity is caused by a sense-reasoning gap between a lower-level probabilistic inference and a higher-level logical reasoning, probabilistic reasoning methods are applied to bridge this gap [113]. Notably, the DyKnow extensions [86] of P-MTL [214] and ProbSTL [215] allow Stream Reasoning with probabilistic temporal logics, where complex formulas can be embedded in probability conditions such as $Pr(a \leftarrow b) < 1$. In [181], the authors followed a different approach by designing a neuro-symbolic stream fusion framework, which includes the learning of rule weights that are mutually independent probabilities. The LARS language was generalised to quantitative extensions in [90] using weighted logic over semirings, which are algebraic structures with multiplication and addition, e.g., the natural numbers, the integers, etc., obeying specific reasonable axioms. Weighted LARS allows lifting several quantitative extensions of logic programming to the streaming setting, among them Problog [184], P-Log [22], and LP^{MLN} [233]. In particular, weighted rules of the form $0.8 : a \leftarrow b$ are supported that induce a probability distribution over the possible answer sets (models) of a program.

Incompleteness. Incompleteness occurs on the level of missing facts in a stream but might also include missing domain knowledge. In rule-based languages, it is handled by non-monotonic and default reasoning approaches that work under the *Closed World Assumption (CWA)* [188], thus stating that a lack of knowledge evidence that a statement is true entails its falsity. Tightly connected to CWA is weak negation, also called *negation as failure (NAF)* in logic programs, which allow the use of NAF literals, i.e., literals of the form *not a*, to express that the atomic formula *a* is not derivable by the program. The already mentioned approaches of the Datalog, DatalogMTL, and ASP-families support CWA (and possible extensions such as PCWA) as well as NAF, whereas certain restrictions are imposed regarding the usage of NAF literals in programs, e.g., stratified programs. In particular, the PCWA addresses the issue of stream data that is not (yet) available at the time when a literal *not a* in a rule is evaluated. For example, Datalog rules $last(T, E) \leftarrow occurs(T, E)$, $not\ later(T, E)$ and $later(T, E) \leftarrow occurs(T1, E), T < T1$ where the first argument encodes time, informally capture the last occurrence of an event *E*; however, when the event *E* occurs at time *T*, evaluation of the first rule is blocked since $later(T, E)$ has to be evaluated, which by the second rule may be postponed indefinitely. To avoid such blocking of the evaluation, only references to past or current data is permitted. The PCWA principle may then be applied: “If $stream(T, \dots)$ is observed in the input stream, conclude $not\ stream(T1, \dots)$, provided that $T1 < T$ and $stream(T1, \dots)$ is not entailed by the fact base augmented with the stream facts having some timestamp $T0 \leq T$.” Syntactically, PCWA can be enforced using local stratification on time. For ontology-based languages, handling incompleteness needs to be addressed differently as they work under the Open World Assumption (OWA), which means that a lack of current

knowledge leaves both possibilities for a statement, being true or false, open. This indicates that missing facts for these approaches either could be settled in a pre-processing step or be asserted by the use of existential quantifiers as available in OWL2 languages. Incompleteness may be also described using quantitative methods, where e.g., the weight of a fact expresses the certainty or probability that an observation is made, whereas the latter values 1 and 0 recover complete knowledge. Other work [73, 74] considered the handling of incomplete information in state streams for runtime verification with MTL. Here, vertices in a progression graph represent formulas, with directed labelled edges between vertices indicating that a formula can be obtained from progressing another (input) formula with a state indicated in the edge's label. A probability mass – representing the ratio of progression paths having reached a formula so far – is pushed between nodes, with terminal nodes representing verdicts (i.e., \top , \perp) and their associated probabilities, allowing for the tracking of verdict probability during progression with incomplete state information.

Constantly-varying. The aspect of constantly changing streams is less of a focus in current research due to three reasons. First, the availability of new data might not trigger the re-evaluation of the conclusions as some approaches are pull-based, whereas in push-based approaches a re-evaluation is triggered. Second, a central processing feature of these approaches is incremental updates, where the variation in the number of updated terms and not the size of updated data matters. For instance, a single ground fact deletion can trigger the re-evaluation of the full knowledge base. Note that not all mentioned approaches in this section support incremental updates. Third, variability can be considered from the semantic point of view, when the meaning of categories, concepts, or relationships changes over time. This concept drift [99] requires a DSR to include monitoring and learning components that can detect and address the drift, respectively. Automatic addressing the drift might be especially complicated since it might require updating the knowledge of a DSR, e.g., add, delete, or update rules in the case of a relational DSR. For instance, in [181], the authors equip their system with a learning algorithm to learn weights of rules and thus counteract the concept drift. The authors of [63] go even further and directly address concept drifts by applying semantic embeddings, i.e., vectors capturing KB consistency, and supervised learning to detect concept drifts in ontology streams. Nevertheless, the problem of concept drift remains largely unaddressed by modern DSR.

5 Inductive Stream Reasoning

Inductive Stream Reasoning (ISR) aims to support reasoning with new knowledge that is generated bottom-up from the data itself, which then may be used to augment deductive reasoning. In particular, this includes integrating knowledge generated by Deep Neural Networks (DNNs) from sub-symbolic inputs using machine learning algorithms, e.g., object or activity classification, but also extracting rules from stream data. In this context, dealing with uncertainty is a key issue. Applications are widespread and include critical areas such as social media analytics [25], robotics, traffic surveillance [87, 76], and autonomous driving [197]. From a sub-symbolic method perspective, a data stream can be seen as an unbounded ordered sequence of data points $S : d_1, d_2, \dots, d_i, d_{i+1}, \dots$ with $i \in \mathbb{N}$. Each data point is represented by a feature vector X_i [242]. The different data points are generated over time, and the method cannot access the entire data stream simultaneously. Usually, most of the methodologies in this context focus on the data stream classification problem, where the goal is to predict the target label y_i associated with each data point d_i whenever d_i is generated. Since existing Machine Learning solutions are not intended for use in a pure streaming scenario where the learning algorithm continuously learns from an ongoing data stream, two main areas emerged: Streaming Machine Learning (SML) [38] and Continual Learning (CL) [141].

The macro-level Stream Reasoning research question can be reformulated to the following generic meso question for ISR:

Meso (Inductive Stream Reasoning): How can we continuously make up-to-date predictions over raw data formats, e.g. sensory observations, that are constantly changing and inevitably noisy?

5.1 Make Sense

By quantifying uncertainty and creating probabilistic models, ISR systems can make more nuanced decisions based on available data streams. One of the most generic formalisations so far towards probabilistic reasoning is proposed in [91]. In another significant development, neuro-symbolic approaches [181] have been formulated to combine the generalisation ability of neural networks with the structural rigour of symbolic logic. By accommodating semantic streams embedded with probabilistic [215] and temporal dimensions [72], the ISR models become highly capable of adapting to dynamic, real-world conditions.

More concretely, given a data stream S of data points, each represented by a vector X_i , a sub-symbolic method produces a data stream of insights generated by implementing a specific learning algorithm. The potential integration of sub-symbolic methods with deductive reasoning offers various architectural possibilities. One approach involves the integration of deductive reasoning with insights generated by sub-symbolic methods. In a notable example, Kirkpatrick et al. [125] leverage sub-symbolic methods across heterogeneous data streams. The varied insights derived are then unified and employed by a deductive reasoner. Belcao et al. [32] use a similar approach to propose a bridge between Big Data Analytics and Semantic Technologies. Conversely, an alternate solution follows a different path. Here, a deductive reasoner is directly applied to a data stream, leading to continuous deduction and transformation. The transformed data becomes the canvas for sub-symbolic methods to apply inductive reasoning and yield outputs, as demonstrated by Barbieri et al. [26]. To make this integration between deductive and inductive reasoning more concrete, let's introduce a practical problem.

► **Example 4 (Taxi cont'd).** Suppose a taxi driver must answer questions like “*What museum can I reach in less than 25 minutes leaving at this exact moment?*” To solve this problem, Della Valle et al. [76] use different types of information. Firstly, the work retrieves monuments, attractions, exhibitions, and events in the city of Milan from different open data in RDF format. It also adds the topology of the city's streets, detections of traffic sensors and weather information. For each traffic sensor, a specific sub-symbolic method is applied. Particularly, the authors train Recurrent Neural Networks to forecast traffic. The different sensors' predictions are propagated to generalise beyond the sensors' locations by exploiting the street graph topology. Ultimately, a deductive stream reasoner comes into play, addressing user queries through deductive reasoning. This reasoner seamlessly integrates RDF data and traffic predictions obtained from sub-symbolic methods.

5.2 Taming Volume

The massive volume of streaming data in real-time from various sources is another issue that ISR aims to tackle. A robust sub-symbolic streaming method should be easily embeddable in a stream processing pipeline capable of running multiple concurrent queries on big data volumes while dealing with high update loads. In terms of data volume, it could range from megabytes in

nanoseconds to terabytes in minutes, depending on the specific requirements of an application. For example, autonomous cars generate around 25 Gigabytes of data per hour including 4-6 radars (0.1-15Mbit/s), 1-5 LIDARs(20-100/Mbit/s), 6-12 cameras (500-3500Mbit/s) and under 0.01Mbit/s sensors such as Ultrasonic, Vehicle motion, GNSA and IMU. Moreover, most of the systems have to deal with multiple concurrent queries on big data volume paired with high update loads generated continually from multiple streams from multiple sources. On top of that, it depends on the reactivity constraint. It can be MB in nanoseconds, GB in seconds, TB in minutes. As mentioned above, it should be impossible to tame that volume, ignoring the streaming nature of the data.

5.3 Taming Variety

Streaming sub-symbolic methods can support different natures of data. SML is usually applied to structured data streams containing data points with tens of features. Classical real-world benchmarks include Airline [120], containing flight arrival and departure details; Forest Cover Type [42], representing forest cover types of specific geographical areas based on different attributes determined by the US Forest Service; and KDDCup99 [210], including data for intrusion detection in a network. On the contrary, CL usually deals with unstructured data like images. Each data point can contain hundreds of thousands of features. Standard benchmarks include streaming versions of the most known computer vision benchmarks (MNIST [237, 108] or CIFAR [147]). An interesting case is represented by the OpenLORIS [201] benchmark, which provides a comprehensive set of visual, inertial, and odometry data captured with real robots in authentic scenes. The goal of the learning model can be scene understanding or evaluating Simultaneous Localisation and Mapping.

Moreover, more complex multi-model data streams can be supported, such as in autonomous driving applications, where data originates from various sensors, each providing a unique lens through which to view the environment. For instance, as shown in [197], an autonomous vehicle may use radar, lidar, and cameras to understand its surroundings, requiring the reasoning system to integrate and make sense of this disparate data using deep neural networks and other signal processing components to lift these raw data into symbolic forms to symbolic solvers or reasoners. In essence, DSRs above can then be used as an underlying component to make logical decisions regarding the observed data.

5.4 Taming Velocity

One of the most critical aspects of ISR is its near real-time decision-making capabilities. For instance, in traffic surveillance applications, the system must make immediate decisions based on incoming data. It cannot afford to wait for the entire data set before beginning the analysis. This is especially true for trajectory predictions, where potential paths or actions are inferred even before complete data is received. Additionally, as soon as data becomes available, immediate insights are formulated and communicated, ensuring minimal delay. However, this challenges existing machine learning and reasoning systems, which may not have been designed to handle the high-speed, ever-changing nature of data streams.

Regarding the sub-symbolic methods, an SML learning model aims to predict the label \hat{y}_i whenever a new data point is generated. It assumes that the actual label y_i arrives after casting the prediction. The model is updated incrementally whenever a new y_i is available. SML models can use each data point only once. In Batch Incremental Learning (BIL), data points are accumulated in fixed-size batches containing tens of them [186]. The model is updated once the mini-batch fills up. SML prioritizes computational efficiency in time and memory. Following BIL's direction,

CL assumes the data points will be grouped into large batches called experiences. However, each experience e_i can contain thousands of data points (rather than tens), randomly accessible as many times as the model requires. A CL strategy can process each e_i for as long as necessary before accepting a new experience.

Concept drift. Concept drift is a critical aspect associated with the evolving environment of data streams. The traditional Machine Learning assumption that data is independent and identically distributed does not hold in this context, where data can change its distribution. A concept is the unobservable random process producing data points [97]. Concept drift is a phenomenon in which the statistical properties of a domain change over time in an arbitrary way [148]. We can categorise concept drift into two primary types: virtual and real. Virtual concept drifts occur when the probabilities $P(X|y)$ or $P(y)$ change. Real concept drifts happen, instead, when there is a change in the $P(y|X)$ probability. Therefore, a virtual concept drift does not affect the class boundary, while a real concept drift introduces a change. Additionally, in cases of abrupt drifts, the new concept instantaneously replaces the old one. Conversely, the new concept gradually or incrementally replaces the old in gradual and incremental drifts. Lastly, it's also possible for concepts to re-occur over time. SML assumes the distribution within a concept to be fixed. SML literature puts a strong effort into automatically detecting concept drifts. These solutions can deal with all concept drifts. Conversely, CL assumes that each new experience introduces an abrupt concept drift. For this reason, it does not use concept drift detectors.

Temporal dependence. Numerous data streams exhibit dependencies on their past values [40, 242]. For instance, an attribute value may result from an auto-regressive transformation applied to preceding instances, such as the fluctuation in commodity prices like electricity [36] or the evolution of weather conditions [84]. Modelling the evolving temporal patterns, e.g., trends or seasonalities, can be challenging, and traditional methods assuming independence between observations are unsuitable [241]. Consequently, addressing this intricate issue requires the development of specialised methods capable of capturing the dependencies inherent in the data. While there has been significant emphasis on detecting concept drifts and developing techniques to adapt to such changes, the issue of independence has received comparatively less attention. Approaching this matter from a SML standpoint, the filtering task within a sequential-state space model, such as Kalman Filters, emerges as a promising avenue for managing concept drift and temporal dependence [240]. Similarly, applying CL methods on Recurrent Neural Networks to learn sequences within a data stream presents another encouraging approach to address this complex problem, as evidenced by the introduction of Continuous Progressive Neural Networks [103].

The challenge of temporal evolution extends to the integration between ISR and DSR. Envisioning an inductive reasoner, such as SML or CL models, processing data and recognizing the entities for input to a deductive reasoner introduces potential issues when entities undergo evolution over time, necessitating adaptive responses from inductive reasoners. In such instances, the challenge may not be adapting to a concept drift but learning the entity's natural evolution. Consequently, accounting for temporal coherence within the data streams becomes crucial. Notably, intriguing benchmarking datasets are showcasing temporal dependencies within the Continual Learning (CL) field. Recent studies [143, 235] introducing datasets with inherent temporal aspects and concepts like temporal distribution shift and coherence, underscore a growing community interest in this direction.

Learning goals and evaluation. Despite both SML and CL learning from data streams and managing concept drifts, they have different objectives. The main goal of SML is to detect concept drifts automatically and quickly adapt to new concepts. An SML model must learn fast, react

quickly to concept drift, and perform well on the current concept. It is also subjected to strict constraints on time and memory consumption. Conversely, CL addresses the stability-plasticity dilemma [151]. When learning new experiences, the model may forget what it has learned during the previous ones. The ability to remember past knowledge is called stability, while learning new knowledge is called plasticity. Too much stability could lead to difficulties in learning new knowledge. Conversely, too much plasticity may lead to forgetting past knowledge and raising the problem known as catastrophic forgetting [132]. The goal is to achieve a trade-off between stability and plasticity. A CL strategy must perform well on all the seen experiences from the first to the current one. This difference in objectives is reflected in different evaluation procedures. SML does not distinguish between training and test data, and each data point is used for both purposes. The evaluation protocols usually include a prequential evaluation [98]. Each time a new data point is generated, the SML model predicts its label \hat{y}_i . When the actual label y_i is available, the protocol updates the evaluation metric (usually accuracy or Cohen's Kappa Score) and then trains the model on d_i . Conversely, CL evaluates the model's ability to mitigate forgetting and learning new experiences [141]. Each experience e_i is split into a training set D_i^{train} and a test set D_i^{test} . All the D_i^{test} are always available, while the D_i^{train} are provided over time. Different types of metrics exist [141]. Accuracy is usually evaluated, after each experience's training, how the model performs on the current experience's test set and all the previous experiences' test sets. Average accuracy evaluates the model's overall performance after the last experience's training by considering the average accuracy on all the D_i^{test} . The Backward Transfer Metric measures the stability of the model and, more in general, how the final version of the model has improved or decreased the performance of the previous versions. After training on the last experience, for each previous experience e_i , it subtracts the accuracy of the model trained on the experience e_i from the one of the current model tested on D_i^{test} . A negative value indicates forgetting. Finally, the Forward Transfer Metric measures how the training on the current experience is useful also for learning the next experience. For each experience e_i , it subtracts the accuracy of a random model tested on D_i^{test} from the one achieved by the model after the training on e_{i-1} . A positive value indicates that the current training positively affects the performance for the next experience.

5.5 Domain Complexity

The main focus of inductive reasoning is on the data and the signals it carries. When extracted, such signals can be used to construct the domain. In this context, the domain complexity is usually a *tuned* parameter: it is up to the scientist or engineer to determine the adequate complexity of the model that will fit the data. Existing solutions try to build models of various complexities. Lecue and Pan [138] propose an approach that extracts association rules from streaming data. Balduini et al. [19] study how to extend inductive reasoning methods to a streaming scenario. The resulting system uses RDF streams to create matrices, which are fed to an inductive reasoner, SUNS, to recommend items.

Learning algorithms. For the sub-symbolic part, different choices are possible. SML usually applies simple models, often based on Statistical Machine Learning. Frequency-based methods track feature frequencies and calculate posterior probabilities using Bayes's theorem. Neighbourhood-based techniques identify neighbours for new samples based on distance, often using a sliding window to manage recent instances. Tree-based classification algorithms are streaming versions of decision trees that use the Hoeffding bound [117] for incremental split node decisions. Techniques like Hoeffding Adaptive Trees (HAT) [37] address concept drift, incorporating concept drift detectors. Ensemble-based methods combine predictions from individual models to enhance generalisation with well-known techniques like Online Bagging, Leveraging Bagging, and Adaptive

Random Forests [107]. Conversely, CL usually applies more complex models based on Deep Learning. It employs three main categories of strategies [141]. Replay approaches (e.g., [176, 183, 5]) store a subset of examples encountered during training in external memory to combat forgetting. They blend this memory with the current data during each iteration to update the model. In this context, Generative Replay methods (e.g., [202]) employ generative models to recreate past examples as needed, eliminating the need for external memory. Regularisation strategies (e.g., [142, 127]) bolster the loss function with additional terms to enhance model stability and mitigate forgetting. They can, for instance, restrict changes in parameters crucial for previous data or enforce consistent network activation over time. Architectural strategies (e.g., [145, 193, 200]) adapt the model's architecture to incorporate new knowledge while minimising forgetting. Popular techniques include expanding the number of layers or units over time and compressing or freezing previous model components. Hybrid approaches (e.g., [147, 187, 198]) that combine elements from more strategy families are often highly effective.

Frameworks. Various frameworks facilitate the application of SML and CL algorithms. Notably, almost all existing frameworks are mainly used for research, as they still have significant limitations for industrial applications. The Massive Online Analysis (MOA) framework [39] presents a broad spectrum of algorithms designed for multiple tasks related to data stream analysis. MOA's tasks encompass classification, regression, multi-label, multi-target, clustering, outlier detection, concept drift detection, active learning, and more. In addition to learning algorithms, MOA offers data generators (e.g., AGRAWAL, Random Tree Generator, and SEA), evaluation methods (e.g., periodic holdout, test-then train, prequential), and statistics (CPU time, RAM-hours, Kappa). The Scalable Advanced Massive Online Analysis (SAMOA)[155] is both a framework and a library that combines stream mining and distributed computing (i.e., MapReduce). SAMOA allows users to abstract the underlying stream processing execution engine and concentrate on the learning problem. It provides adapted versions of stream learners for distributed processing, including the Vertical Hoeffding Tree algorithm[129], bagging, and boosting. Vowpal Wabbit (VW) is an open-source machine learning library featuring an efficient and scalable implementation with several learning algorithms. VW has demonstrated its capability by learning from a tera feature dataset using 1000 nodes in approximately an hour [3]. StreamDM, an open-source framework for big data stream mining, utilizes the Spark Streaming extension of the core Spark API. One notable advantage of StreamDM over existing frameworks is its direct integration with the Spark Streaming API, which efficiently handles complex issues arising from the underlying data sources, such as out-of-order data and recovery from failures. There has been a significant recent development in SML algorithms for Python, with the River package [154] being particularly noteworthy. River supports various ML tasks, including regression, classification, and clustering. Moreover, River is versatile enough for ad hoc tasks, such as computing online metrics and detecting concept drift. Finally, Avalanche [146] deserves mention as the first experiment of an end-to-end Library for reproducible CL research and development. It encompasses implementing state-of-the-art CL strategies, standard benchmarks, evaluation metrics, and evaluation scenarios.

5.6 Data Quality

ISR has the potential, to address data imperfections, such as incompleteness and noise, that can have a disruptive effect on Deductive Stream Reasoning. Data incompleteness arises in sensor networks due to factors like sensor battery depletion or network link interruptions. In social media, instead, it arises due to limited sampling rates in social stream APIs or – in a certain sense luckily – because conversations also occur outside social networks. Noise issues encompass sensor network imperfections or operational deviations. In processing unstructured data such as text, sounds,

images and videos, it occurs due to low accuracy in tools used for analysing them such as the inability to catch irony, difficulties in transcribing phonetically ambiguous words, or in evaluating occlusions in object detection and tracking.

DSMS and CEP have traditionally handled noise [68]. Two main types of noise have been identified, affecting content and temporal annotations. Content noise pertains to inaccuracies in data from sensors or human interactions, potentially leading to incorrect conclusions. Statistical methods can manage noise in simple schemas, but more complex schemas require advanced techniques such as [137]. Researchers can explore streaming machine learning approaches to process noisy data, coupled with deductive reasoning techniques like inconsistency repair [10, 51], and belief revision [192, 190]. Temporal annotation noise involves out-of-order data items, particularly when multiple streams with different time annotations are involved. Solutions exist for handling temporal noise, with room for semantic enhancements [203, 144]. Addressing temporal noise may involve aligning diverse temporal annotations from different sources. Existing solutions can be adapted, with semantic enhancements offering promising opportunities [51].

In summary, ISR must increase efforts in tackling imperfections like incompleteness and noise, necessitating innovative solutions and approaches for both content and temporal issues.

6 Discussion

This paper *grounds* the Stream Reasoning research, by providing a clear overview of its different constituent research areas, and explaining how each of these areas target its different dimensions. For over a decade, practitioners from different research communities have contributed to Stream Reasoning research, each from within their perspective and background. Since 2015, researchers in these different communities have organised the *Stream Reasoning Workshops*, a recurring event with the purpose of sharing perspectives, challenges, and experiences around Stream Reasoning topics.

This paper provides an overview of the main research contributions discussed during the *Stream Reasoning Workshops*. Moreover, this paper provides a crystallisation of how each of the different research areas within Stream Reasoning perceive and tackle the Stream Reasoning research dimensions. By understanding how the different areas differ and relate and how they perceive the various Stream Reasoning research dimensions, this paper *grounds* the Stream Reasoning research. We conclude with a discussion of the take-away messages and open challenges for the next years.

6.1 Discussion of the Research Dimensions

We will now discuss how the different areas differ or relate in tackling the dimensions introduced by the original SR research questions. Table 3 summarises the discussion by providing an overview of how the different areas target the research dimensions.

- **Making Sense:** Even though each area has a different focus when *making sense* of the data streams, e.g., Stream Processing focuses on Continuous Querying, RSP/SLD on Continuous Data Integration and Querying, DSR on incremental materialisation, model checking and planning, each area has a *continuous* component when making sense and is typically done through some kind of query. In Stream Processing this is an SQL-like language, in RSP/SLD a dialect of SPARQL, while in DSR this is mostly done through rules.
- **Taming Volume:** Volume has not been the main point of focus for any of the approaches, except for Stream Processing which incorporates techniques to scale horizontally. The mechanisms to tame variety, as in RSP/SDL, or to incorporate rich domain complexity, as in DSR, have a negative impact on the volume of data that can be processed.

■ **Table 3** Continuous Querying (CQ); Consistency (C); Data Integration (DI); Model Checking (MC); Materialisation (MAT); Planning (P); Clustering (CLU); Classification (CLA); Temporal Logic (TL); Incompleteness (I); Noise (N); Volatility (V).

Dimension	SP	SLD	DSR	ISR
Making Sense	CQ	C, DI	MAT/MC/P	CLU/CLA
Velocity	Sub-millisecond	Milliseconds	Seconds	Milliseconds
Variety	Relational, Document	RDF	Relational	Multimodal
Volume (Scale Up/Out)	Yes/Yes	Yes/Limited	Yes/No	Yes/Edge
Domain Complexity	Data Schema	RDFS+	OWL2, ASP, TL	Variable
Data Quality	I, N	I	I, C	I, V, N

- **Taming Variety:** Stream Processing requires a manual mapping to the used relational schema, which is not a flexible approach and is typically not well-suited for data integration purposes. RSP/SLD are able to solve a data integration problem in a continuous fashion by relying on RDF and the extension of the Semantic Web stack and is thus the best-suited approach for handling variety. DSR and ISR typically map to RDF to increase the support of data variety, but do not directly build upon the Semantic Web stack. ISR tames variety in a different way by supporting multi-modal streams, e.g. integrating video with numerical sensor readings.
- **Taming Velocity:** All approaches use some form of *windowing* to deal with data streams, however, the requirements in terms of responsiveness differ, Stream Processing focuses on sub-milliseconds latency, RSP/SLD and ISR focus on milliseconds latency, while DSR is satisfied with latency in terms of seconds. Stream Processing is the fastest, as it does not require any overhead to perform data integration such as RSP/SLD, checking models and incorporating complex domains as in DSR, or performing predictions as in ISR.
- **Domain Complexity:** DSR allows the incorporation of the most complex domain knowledge, at the cost of performance. RSP/SLD supports little domain complexity in order to prioritise responsiveness. Although the focus is growing, Stream Processing has limited support for domain complexity, rather than focusing on volume and velocity.
- **Data Quality** has not been properly addressed by the different approaches. ISR has valuable solutions for *veracity* through predictions, while DSR can handle *incompleteness* very well by inferring missing facts in a deductive manner through the incorporation of domain knowledge. Stream Processing solutions to deal with *constantly varying* data, which have been adopted to some extent by SLD and RSP. Different areas have focused to some extent on the different aspects of data quality, however, none of the approaches has targeted them simultaneously.

It is clear that each area has tackled different aspects of the original SR research dimensions. Some dimensions have been tackled by all of them, e.g. Velocity, while others have received more attention in one of the areas, e.g. Variety in RSP/SLD, Domain Complexity in DSR, and incompleteness in ISR. In order to realise the SR vision, cross-pollination between different areas is needed to cover all the dimensions simultaneously.

6.2 Overlapping Approaches

Even though the large body has been done in the distinct areas of the SR research, there has been research conducted that combines ideas from multiple areas:

Streaming Linked Data & Deductive Stream Reasoning

From within RSP/SLD, there has been a quest to increase the expressiveness of the reasoning capabilities and thus increase the domain complexity. This has led to an investigation into how more expressive reasoners can be combined or integrated with RSP engines.

Morph-Streams [56] and OntopStream [32], which employ an OBDA approach can support OWL2 QL ontologies through their rewriting regimes. RoXi [44] is a recent effort to increase the domain complexity of RSP engines to OWL2 RL, through various optimisations to enable Datalog reasoning over RDF streams in an efficient fashion, e.g., through efficient maintenance of the materialisation in the window [79] or pruning of the reasoning rules [47].

However, in general, there is a mismatch between the complexity of more expressive reasoning algorithms and the change frequency of the data streams that RSP engines try to tackle. To solve this mismatch, the vision of Cascading Reasoning [204] emerged, which suggests a layered approach of processing and reasoning engines where the lower layers process the high-velocity data with techniques of limited complexity, going up in the layers, the amount of data decreases while the complexity of data increasing, ultimately resulting in support for expressive reasoning over high-velocity streams. There have been first realisations in this area, such as StreamRule [152], which combines the CQELS engine for RSP with ASP reasoning, and Streaming Massif [48], which combines C-SPARQL with the HermiT reasoner for Description Logic and features complex event processing capabilities. However, these initial approaches still require manually defining the processing at each layer and thus do not constitute a generic approach to define the reasoning and processing across various layers. Some interesting early developments in that area aim to rewrite registered continuous queries and to prune datalog rules in order to push the processing to lower layers in the hierarchy [45].

Streaming Linked Data & Inductive Stream Reasoning

In the early days of Stream Reasoning, there were several attempts to combine Inductive and RSP for the analysis of social media streams. The first seminal work [25] introduced a pipeline combining the deduction of a C-SPARQL engine [24] with a long and with a short window, whose contents is transformed into a matrix factorisation system, in order to recommend links in a bipartite graph that pairs users to movies. The combination of a long and short window allows to capture both long lasting knowledge and hype effects. A similar approach was further developed in [19]. Follow-up work led to demonstration of the applicability of this schema to venue recommendation [18].

CQELS 2.0 [136] extends the CQELS engine with more powerful inductive capabilities such as Deep Neural Networks, and it allows for the fusion of various multi-modal data streams. For example, by fusing object detection on video streams with sensor readings of location and velocity and by converting the results to the RDF model, the streams can be queried continuously in order to solve multi-object tracking in a declarative fashion.

Interestingly, the approaches in this overlap augment the data itself and extend the query language in order to support query answering over certain predictions as a result of the inductive part.

Deductive Stream Reasoning & Inductive Stream Reasoning

Above we presented ISR as a method to generate new knowledge that can be used in combination with deductive reasoning. A natural question is then a fruitful combination of symbolic deductive reasoning techniques with inductive techniques. Induction may further be interrelated with abductive reasoning for learning, by generating hypothetical facts from which new complex knowledge may be inferred, possibly in a cycle [123]. For such learning, dealing with uncertainty is an important aspect. Specifically, in the PrASP [161] approach programs may inductively learn from data streams and can be incrementally evaluated. For Stream Reasoning programs in [181], merely the weights of rules as independent probabilities can be learned; that work showed how the potential of combining DSR and ISR in the realm of object tracking under uncertainty in traffic monitoring, demonstrated in [205], can be pushed to real-time performance with proper Stream Reasoning infrastructure. These hybrid approaches are also particularly important in applications like robotics, where generalisation and structured knowledge are vital [214].

6.3 Open Challenges

In terms of open challenges, we discuss the open opportunities for each research area that should be solved in the next years in order to push the field closer to the true realisation of the Stream Reasoning vision.

6.3.1 Stream Processing

With the availability of several large-scale technological infrastructures for stream processing, which have been successfully deployed in multiple industries, it may be tempting to consider stream processing as a solved problem. We warn the reader to make such a conclusion as we believe that there are still several important challenges that need to be addressed.

First of all, current solutions, e.g., **Apache Flink**, are designed to be general-purpose solutions. Such solutions sacrifice some performance to support a wider range of applications. Since approaches for RSP, SLD, DSR, and ISR can be computationally demanding, we argue that an important challenge is:

- *How can we optimise current general-purpose solutions for stream processing to support more efficient reasoning applications?*

An alternative approach consists of improving current reasoning solutions exploiting what has been learned while developing the current state-of-the-art for generic stream processing. Hence another important challenge is:

- *How can we improve the state-of-the-art for stream reasoning adopting the best practices in large-scale stream processing?*

In both cases, the challenges call for a deeper collaboration between members of various communities. This collaboration has a huge potential, not only to solve the problem at hand but also to discover new research avenues that can benefit both sides.

6.3.2 Streaming Linked Data

There are still various open challenges in the realm of RSP and SLD. We summarise the most important ones in the form of micro questions:

- *How can we increase the expressivity of the ontologies when supporting reasoning in SLD?* Most approaches provided limited to no reasoning capabilities or simple regimes such as RDFS. The reason is that there is a mismatch between the complexity of the algorithms to perform more expressive reasoning and the responsiveness and low latency requirements of many of the

use cases targeted in RSP. One vision to mitigate this is the idea of Cascading Reasoning [204], which suggests a layered approach of processing and reasoning engines where the lower layers process the high-velocity data with techniques with limited complexity, going up in the layers, the amount of data decreases and the complexity of data increasing, ultimately resulting in supporting expressive reasoning over high-velocity streams. There have been first realisations in this area, such as Streaming Massif [48], however, it is still required to manually define the processing at each layer.

- *How can virtual processing of RDF Streams be integrated with techniques that process “real” RDF Streams?* Both techniques have their benefits, but they have not been combined in order to reap the benefits of both approaches. Doing so would allow various optimisations as the conversion from virtual to “real” RDF Streams could be done in an optimal fashion, while integrating different sources of data. This could lead to more flexible RSP engines, able to adapt to RDF and non-RDF streams through virtualisation. The question of seamless mapping of these sources to WoT or IoT ontologies can also be beneficial, especially for environments where native RDF is not necessarily the most efficient option.
- *How can we use SLD as the basis for autonomous computing on the Web in rapidly changing environments?* The *linked* nature of SLD has only been exploited to a limited extent [54]. However, the potential for autonomous agency in stream processors [218] is high in terms of distribution of query processing load, and local processing capabilities – e.g., for WoT environments. For this to be properly implemented, it would be necessary to specify agent-based primitives such as goals, intentions, or beliefs, related to the capabilities of RSP engines. Moreover, it would be necessary to include scheduling and coordination mechanisms for enabling efficient interactions among the autonomous RDF stream engines [209]. Moreover, when data streams are managed using the web as the processing platform, new problems may emerge, in particular when publishing personal data which may contain sensitive information. It is therefore important to account for the privacy issues that may arise [77].
- *How can agents on the web collaborate to answer continuous queries?* Although the vision of interconnected RSP engines has been discussed in the past [80], its realisation is yet to become a reality. It will be necessary to further investigate decentralised query processing for stream environments, as well as explore how to formalise the semantics of cascading stream processing, including how temporal aspects are affected by distributed query answering.
- *How can we continuously query RDF streams that have a much higher change frequency than the milliseconds change frequency that SLD solution can currently handle?* To cope with streaming loads under high demand as it is currently done in non-RDF systems, hybrid approaches that combine RSP and native stream data processing can lead to promising solutions. This would require further exploring optimisation opportunities, as well as exploiting OBDA-based approaches using underlying high-performance native engines.
- *How can we perform data integration continuously over data streams of different formats?* Although this topic has been addressed through virtualisation and materialisation to some extent, most streams on the Web are not RDF nor follow RDF-like formats. Moreover, in many cases there is no interest in necessarily transforming all of the contents into RDF. Hence, there is a challenge to manage hybrid RDF streams, which could be processed by engines that can handle different stream models.

6.3.3 Deductive Stream Reasoning

There are numerous challenges related to the development of DSRs. Among the most important ones are undoubtedly those concerning the efficiency of the reasoning process so that it can meet the requirements of a typical streaming scenario. In this context, we can distinguish two main challenges:

- *How can we shorten the response time of query answering with current formalisms such as LARS and DatalogMTL?* Examples of works in this category are the ones that focus on maintaining the materialisation after updates [27], or the ones that combine materialisation with other techniques [230].
- *Can we find a good balance between the expressivity of the reasoning while still maintaining a high throughput?* For instance, works in this category are the ones that restrict existing formalisms to allow a faster execution [27] or the ones that provide extensions to support higher expressivity without compromising performance [223]. Another type of approaches falling into this category is focused on formal analysis of the input stream for data and/or reasoning parallelisation [179, 92].

Another important challenge consists of improving the usability of DSRs. Currently, the usage of a DSR requires a good understanding of the underlying technology *and* a proper way to represent the domain using a formal language. It may be challenging to meet these requirements in practice. Hence, two important research directions are:

- *How can we automatically capture knowledge in a formal language so that DSRs can use it to reason over the data streams?* Although inductive logic programming systems are available for popular languages like Prolog or ASP, as far as we know, this research question has not been properly investigated yet.
- *How can we communicate to users that are not familiar with formal languages why a reasoner has returned some particular information?* This topic has become particularly important since AI is being adopted in an increasing number of domains. Like the previous question, this topic of research on streaming scenarios has not yet been studied, although in this case there are numerous works in static contexts that can be used as a starting point. Furthermore, recent developments in large language models open an interesting perspective for user-friendly communication between formal-language based systems and humans with little formal background and training. Current attempts to exploit such models in fields like planning, argumentation, and temporal logic may be lifted to the DSR setting.

Finally, another important topic of research consists of combining the power of logic-based reasoning with techniques that can deal with uncertain data. This combination will allow us to counter more effectively all the challenges related to data quality. Moreover, uncertainty is inherently present in many domains and dealing with it is becoming increasingly important as more and more machine learning techniques are being introduced in key decision processes.

Also in this case we identify two main research directions:

- *How can we include uncertainty, either aleatoric or epistemic, into the query answering process?* This question can be investigated by trying to adopt existing approaches to uncertainty such as the possible world semantics [207] into a DSR or by the development of completely new frameworks tailored to the needs of streaming scenarios. The PrASP system [161] discussed in Section 4.5 represents a preliminary attempt to rely on possible world semantics to handle uncertainty in an ASP-based stream reasoner, but the limited scalability makes it unsuitable for real-world scenarios. This calls for the investigation of better sampling strategies and uncertainty reasoning that does not necessarily rely on external solvers, but specifically tailored to work with data streams.
- *How can we exploit stream reasoning with uncertainty to formulate what-if scenarios?* This problem is particularly important when a DSR is used to help decision processes when errors can be very costly. Reasoning to determine not only what is true now, but also what can/cannot be true in the near future can be invaluable as it could help to prevent malfunctioning in power plants or disasters in crowd management. Unfortunately, as far as we know this topic has not been investigated yet in a streaming setting.

The aforementioned list is by no means a complete enumeration of all issues. There are also several other problems which are equally valuable and deserve much attention by the community. For instance, currently we lack solid and fair evaluation frameworks to compare the performance of DSRs. In other more mature fields, the community has agreed on establishing an independent team, which include many representatives from industry and academia, to define comprehensive benchmarks. Examples of such initiatives are the TPC-H benchmark suite [220] and the LDBC consortium [208]. Doing so also for DSR systems or, more in general, for stream reasoning in general is a natural next step. Another important problem relates to the development of tools that are beyond proof-of-concepts and which are robust enough to be used outside the community.

6.3.4 Inductive Stream Reasoning

By addressing challenges related to real-time decision-making, data variety, veracity, and massive volumes, an ISR system aims to offer a robust framework for reasoning in dynamic environments. Meso-level questions, like how to ground multi-modal data into high-level reasoning, still require further exploration and implementation to be used in killer applications such as autonomous vehicles and robotics. Likewise, the application of model checking under conditions of uncertainty and incompleteness remains a vital area for future research. Therefore, as we move towards an increasingly connected and data-intensive world, the role of inductive stream reasoning as a harmonising factor between traditional models and complex reality is set to become even more crucial. In this context, we identify the following research challenges:

- *How to ground multimodal sub-symbolic data streams into high-level reasoning?* The neural-symbolic field proposes several solutions to connect sub-symbolic data and high-level reasoning facts and rules, such as DeepProblog [149] and NeurASP [234]. However, there are only a few works, such as [206] and [181] considering streaming aspects of data. Most of them only address this problem in very narrow domains or specific types of data and rules. Hence, this calls for a systematic investigation of stream-first approaches, including learning to inference phases of grounding multimodal stream data into high-level stream reasoning.
- *How to implement performant and scalable ISR systems for real world applications?* Most of the motivated applications for ISR have very critical requirements for low-latency in conjunction with data volume like pointed out in Section 5. However, so far, there is no implementation that can deal with the data scale of the targeted applications, e.g., autonomous driving, robotics, and automation. To make ISR usable for such targeted applications, there is an urgent need for more systems taking operational requirements seriously to make real-life impact to relevant industries.
- *Can we perform model checking under uncertainty applied to signals modelled as streams representing the physical world, when that incomplete and rapidly-available information is assumed to evolve over time as the result of external processes?* Model checking has the potential to introduce formal verification in stream processing pipelines [31], thus contributing to verifying correctness of processing results, identifying problematic stream processes, or providing resource allocation information. Run time verification of observations and events in data streams can also benefit from these works [109], although current approaches do not fully incorporate uncertainty and domain knowledge as it is the case in ISR.

6.4 Summary

For more than a decade, researchers and practitioners from different communities have contributed to advance Stream Reasoning, each from within their area and perspective. Since 2015, members of these communities have organised the *Stream Reasoning Workshop*, an annually recurring event

with the purpose of sharing perspectives, challenges, and experiences around the Stream Reasoning topic. This paper not only reviews the main research contributions discussed at these workshops, but provides a clear overview of the different areas that constitute the research field. Furthermore, it crystallises how each of these areas perceives and tackles the various dimensions of Stream Reasoning research. By understanding the views of these areas and how they relate and differ, the Stream Reasoning research is *grounded* in sense. Future research efforts in the areas may be aligned and benefit from each other based on our analysis, leading to powerful stream reasoning technology and systems that are urgently needed in an ever streaming world.

References

- 1 Zainab Abbas, Vasiliki Kalavri, Paris Carbone, and Vladimir Vlassov. Streaming graph partitioning: An experimental study. *Proc. VLDB Endow.*, 11(11):1590–1603, 2018. doi:10.14778/3236187.3236208.
- 2 Lorenzo Affetti, Alessandro Margara, and Gianpaolo Cugola. Tspoon: Transactions on a stream processor. *J. Parallel Distributed Comput.*, 140:65–79, 2020. doi:10.1016/j.jpdc.2020.03.003.
- 3 Alekh Agarwal, Olivier Chapelle, Miroslav Dudík, and John Langford. A reliable effective terascale linear learning system. *J. Mach. Learn. Res.*, 15(1):1111–1133, 2014. doi:10.5555/2627435.2638571.
- 4 Muhammad Intizar Ali, Feng Gao, and Alessandra Mileo. Citybench: A configurable benchmark to evaluate rsp engines using smart city datasets. In *International semantic web conference*, pages 374–389. Springer, 2015. doi:10.1007/978-3-319-25010-6_25.
- 5 Rahaf Aljundi, Eugene Belilovsky, Tinne Tuytelaars, Laurent Charlin, Massimo Caccia, Min Lin, and Lucas Page-Caccia. Online Continual Learning with Maximal Interfered Retrieval. In *NeurIPS*, pages 11849–11860, 2019. URL: <https://proceedings.neurips.cc/paper/2019/hash/15825aee15eb335cc13f9b559f166ee8-Abstract.html>.
- 6 Denise Anglica, Giovambattista Ianni, Francesco Pacenza, and Jessica Zangari. Integrating aspbased incremental reasoning in the videogame development workflow (application paper). In Michael Hanus and Daniela Incezan, editors, *Practical Aspects of Declarative Languages - 25th International Symposium, PADL 2023, Boston, MA, USA, January 16-17, 2023, Proceedings*, volume 13880 of *Lecture Notes in Computer Science*, pages 96–106. Springer, 2023. doi:10.1007/978-3-031-24841-2_7.
- 7 Darko Anicic, Jean-Paul Calbimonte, Óscar Corcho, Daniele Dell’Aglio, Emanuele Della Valle, Shen Gao, Alasdair J.G. Gray, Danh Le Phuoc, Robin Keskisärkkä, Alejandro Llaves, Alessandra Mileo, Bernhard Ortner, Adrian Paschke, Monika Solanki, Roland Stühmer, Kia Teymourian, and Peter Wetz. RDF Stream Processing: Requirements and Design Principles. Technical report, W3C RSP Community Group, 2015. URL: https://streamreasoning.org/RSP-QL/RSP_Requirements_Design_Document/.
- 8 Darko Anicic, Sebastian Rudolph, Paul Fodor, and Nenad Stojanovic. Stream reasoning and complex event processing in ETALIS. *Semantic Web*, 3(4):397–407, 2012. doi:10.3233/SW-2011-0053.
- 9 Arvind Arasu, Shivnath Babu, and Jennifer Widom. The CQL continuous query language: semantic foundations and query execution. *VLDB J.*, 15(2):121–142, 2006. doi:10.1007/s00778-004-0147-z.
- 10 Marcelo Arenas, Leopoldo E. Bertossi, and Jan Chomicki. Consistent query answers in inconsistent databases. In Victor Vianu and Christos H. Papadimitriou, editors, *Proceedings of the Eighteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, May 31 - June 2, 1999, Philadelphia, Pennsylvania, USA*, pages 68–79. ACM Press, 1999. doi:10.1145/303976.303983.
- 11 Michael Armbrust, Tathagata Das, Joseph Torres, Burak Yavuz, Shixiong Zhu, Reynold Xin, Ali Ghodsi, Ion Stoica, and Matei Zaharia. Structured streaming: A declarative api for real-time applications in apache spark. In *SIGMOD*, 2018. doi:10.1145/3183713.3190664.
- 12 Alessandro Artale and Enrico Franconi. Temporal description logics. In *Handbook of Temporal Reasoning in AI*, pages 375–388. Elsevier, 2005. doi:10.1016/S1574-6526(05)80014-8.
- 13 Alessandro Artale, Roman Kontchakov, Alisa Kovtunova, Vladislav Ryzhikov, Frank Wolter, and Michael Zakharyashev. First-order rewritability of ontology-mediated queries in linear temporal logic. *CoRR*, abs/2004.07221, 2020. doi:10.48550/arXiv.2004.07221.
- 14 Alexander Artikis, Marek J. Sergot, and Georgios Paliouras. An event calculus for event recognition. *IEEE Trans. Knowl. Data Eng.*, 27(4):895–908, 2015. doi:10.1109/TKDE.2014.2356476.
- 15 Ahmed Awad, Riccardo Tommasini, Samuele Langhi, Mahmoud Kamel, Emanuele Della Valle, and Sherif Sakr. D²ia: User-defined interval analytics on distributed streams. *Inf. Syst.*, 104:101679, 2022. doi:10.1016/j.is.2020.101679.
- 16 Shivnath Babu and Jennifer Widom. Continuous queries over data streams. *SIGMOD Rec.*, 30(3):109–120, 2001. doi:10.1145/603867.603884.
- 17 Fahiem Bacchus and Froduald Kabanza. Planning for temporally extended goals. In *Proceedings of the 13th AAAI conference of Artificial Intelligence*, pages 1215–1222, 1996. URL: <http://www.aaai.org/Library/AAAI/1996/aaai96-180.php>.

- 18 Marco Balduini, Alessandro Bozzon, Emanuele Della Valle, Yi Huang, and Geert-Jan Houben. Recommending venues using continuous predictive social media analytics. *IEEE Internet Comput.*, 18(5):28–35, 2014. doi:10.1109/MIC.2014.84.
- 19 Marco Balduini, Irene Celino, Daniele Dell’Aglia, Emanuele Della Valle, Yi Huang, Tony Kyung-il Lee, Seon-Ho Kim, and Volker Tresp. Reality mining on micropost streams - deductive and inductive reasoning for personalized and location-based recommendations. *Semantic Web*, 5(5):341–356, 2014. doi:10.3233/SW-130107.
- 20 Marco Balduini, Irene Celino, Daniele Dell’Aglia, Emanuele Della Valle, Yi Huang, Tony Lee, Seon-Ho Kim, and Volker Tresp. Bottari: An augmented reality mobile application to deliver personalized and location-based recommendations by continuous analysis of social media streams. *Journal of Web Semantics*, 16:33–41, 2012. doi:10.1016/j.websem.2012.06.004.
- 21 François Bancelhon. Naive evaluation of recursively defined relations. In Michael L. Brodie and John Mylopoulos, editors, *On Knowledge Base Management Systems: Integrating Artificial Intelligence and Database Technologies, Book resulting from the Islamorada Workshop 1985 (Islamorada, FL, USA)*, Topics in Information Systems, pages 165–178. Springer, 1985.
- 22 Chitta Baral, Michael Gelfond, and J. Nelson Rushton. Probabilistic reasoning with answer sets. *Theory Pract. Log. Program.*, 9(1):57–144, 2009. doi:10.1017/S1471068408003645.
- 23 Daniel Barbará. The characterization of continuous queries. *Int. J. Cooperative Inf. Syst.*, 8(4):295, 1999. doi:10.1142/S0218843099000150.
- 24 Davide Francesco Barbieri, Daniele Braga, Stefano Ceri, Emanuele Della Valle, and Michael Grossniklaus. C-sparql: a continuous query language for rdf data streams. *International Journal of Semantic Computing*, 4(01):3–25, 2010. doi:10.1142/S1793351X10000936.
- 25 Davide Francesco Barbieri, Daniele Braga, Stefano Ceri, Emanuele Della Valle, Yi Huang, Volker Tresp, Achim Rettinger, and Hendrik Wermser. Deductive and inductive stream reasoning for semantic social media analytics. *IEEE Intell. Syst.*, 25(6):32–41, 2010. doi:10.1109/MIS.2010.142.
- 26 Davide Francesco Barbieri, Daniele Braga, Stefano Ceri, Emanuele Della Valle, Yi Huang, Volker Tresp, Achim Rettinger, and Hendrik Wermser. Deductive and inductive stream reasoning for semantic social media analytics. *IEEE Intell. Syst.*, 25(6):32–41, 2010. doi:10.1109/MIS.2010.142.
- 27 Hamid R. Bazoobandi, Harald Beck, and Jacopo Urbani. Expressive stream reasoning with laser. In *ISWC*, pages 87–103, 2017. doi:10.1007/978-3-319-68288-4_6.
- 28 Harald Beck, Bruno Bierbaumer, Minh Dao-Tran, Thomas Eiter, Hermann Hellwagner, and Konstantin Schekotihin. Stream reasoning-based control of caching strategies in CCN routers. In *IEEE International Conference on Communications, ICC 2017, Paris, France, May 21-25, 2017*, pages 1–6. IEEE, 2017. doi:10.1109/ICC.2017.7996762.
- 29 Harald Beck, Minh Dao-Tran, and Thomas Eiter. LARS: A logic-based framework for analytic reasoning over streams. *Artif. Intell.*, 261:16–70, 2018. doi:10.1016/j.artint.2018.04.003.
- 30 Harald Beck, Thomas Eiter, and Christian Folie. Ticker: A system for incremental asp-based stream reasoning. *TPLP*, 17(5-6):744–763, 2017. doi:10.1017/S1471068417000370.
- 31 Alexis Bédard and Sylvain Hallé. Model checking of stream processing pipelines. In *28th International Symposium on Temporal Representation and Reasoning (TIME 2021)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.TIME.2021.5.
- 32 Matteo Belcao, Emanuele Falzone, Enea Bionda, and Emanuele Della Valle. Chimera: A bridge between big data analytics and semantic technologies. In *The Semantic Web–ISWC 2021: 20th International Semantic Web Conference, ISWC 2021, Virtual Event, October 24–28, 2021, Proceedings 20*, pages 463–479. Springer, 2021. doi:10.1007/978-3-030-88361-4_27.
- 33 Fethi Belghaoui, Amel Bouzeghoub, Zakia Kazi-Aoul, and Raja Chiky. Pol: A pattern oriented load-shedding for semantic data stream processing. In *Web Information Systems Engineering–WISE 2016: 17th International Conference, Shanghai, China, November 8-10, 2016, Proceedings, Part II 17*, pages 157–171. Springer, 2016. doi:10.1007/978-3-319-48743-4_13.
- 34 Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web: A new form of web content that is meaningful to computers will unleash a revolution of new possibilities. In Oshani Seneviratne and James A. Hendler, editors, *Linking the World’s Information: Essays on Tim Berners-Lee’s Invention of the World Wide Web*, volume 52 of *ACM Books*, pages 91–103. ACM, 2023. doi:10.1145/3591366.3591376.
- 35 Dominic Betts, Julian Dominguez, Grigori Melnik, Fernando Simonazzi, and Mani Subramanian. Exploring cqrs and event sourcing: A journey into high scalability, availability, and maintainability with windows azure, 2013. doi:10.5555/2509680.
- 36 Albert Bifet. Classifier concept drift detection and the illusion of progress. In *ICAISC (2)*, volume 10246 of *LNCS*, pages 715–725. Springer, 2017. doi:10.1007/978-3-319-59060-8_64.
- 37 Albert Bifet and Ricard Gavaldà. Adaptive Learning from Evolving Data Streams. In *IDA*, volume 5772 of *LNCS*, pages 249–260. Springer, 2009. doi:10.1007/978-3-642-03915-7_22.
- 38 Albert Bifet, Ricard Gavaldà, Geoff Holmes, and Bernhard Pfahringer. *Machine learning for data streams: with practical examples in MOA*. MIT press, 2018.
- 39 Albert Bifet, Geoff Holmes, Richard Kirkby, and Bernhard Pfahringer. MOA: massive online analysis. *J. Mach. Learn. Res.*, 11:1601–1604, 2010. doi:10.5555/1756006.1859903.
- 40 Albert Bifet, Jesse Read, Indre Zliobaite, Bernhard Pfahringer, and Geoff Holmes. Pitfalls in benchmarking data stream classification and how to avoid them. In *ECML/PKDD (1)*, volume 8188 of *Lecture Notes in Computer Science*,

- pages 465–479. Springer, 2013. doi:10.1007/978-3-642-40988-2_30.
- 41 Christian Bizer, Tom Heath, Kingsley Idehen, and Tim Berners-Lee. Linked data on the web (ldow2008). In *Proceedings of the 17th international conference on World Wide Web*, pages 1265–1266, 2008. doi:10.1145/1367497.1367760.
 - 42 Jock A Blackard and Denis J Dean. Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables. *Computers and electronics in agriculture*, 24(3):131–151, 1999. doi:10.5555/928509.
 - 43 Andre Bolles, Marco Grawunder, and Jonas Jacobi. Streaming sparql-extending sparql to process data streams. In *The Semantic Web: Research and Applications: 5th European Semantic Web Conference, ESWC 2008, Tenerife, Canary Islands, Spain, June 1-5, 2008 Proceedings 5*, pages 448–462. Springer, 2008. doi:10.1007/978-3-540-68234-9_34.
 - 44 Pieter Bonte and Femke Ongenae. Roxi: a framework for reactive reasoning. In *The Semantic Web: ESWC 2023 Satellite Events*, pages 159–163. Springer Nature Switzerland, 2023. doi:10.1007/978-3-031-43458-7_30.
 - 45 Pieter Bonte and Femke Ongenae. Towards cascading reasoning for generic edge processing. In *ESWC2023, the First International Workshop on Semantic Web on Constrained Things*, 2023. URL: <https://ceur-ws.org/Vol-3412/paper4.pdf>.
 - 46 Pieter Bonte and Riccardo Tommasini. Streaming linked data: A survey on life cycle compliance. *Journal of Web Semantics*, 77:100785, 2023. doi:10.1016/j.websem.2023.100785.
 - 47 Pieter Bonte, Riccardo Tommasini, Filip De Turck, Femke Ongenae, and Emanuele Della Valle. C-sprite: efficient hierarchical reasoning for rapid rdf stream processing. In *Proceedings of the 13th ACM International Conference on Distributed and Event-based Systems*, pages 103–114, 2019. doi:10.1145/3328905.3329502.
 - 48 Pieter Bonte, Riccardo Tommasini, Emanuele Della Valle, Filip De Turck, and Femke Ongenae. Streaming MASSIF: Cascading reasoning for efficient processing of iot data streams. *Sensors*, 18(11):3832, 2018. doi:10.3390/s18113832.
 - 49 Stefan Borgwardt, Marcel Lippmann, and Veronika Thost. Temporalizing rewritable query languages over knowledge bases. *J. Web Semant.*, 33:50–70, 2015. doi:10.2139/ssrn.3199188.
 - 50 Irina Botan, Peter M. Fischer, Donald Kossmann, and Nesime Tatbul. Transactional stream processing. In Elke A. Rundensteiner, Volker Markl, Ioana Manolescu, Sihem Amer-Yahia, Felix Naumann, and Ismail Ari, editors, *15th International Conference on Extending Database Technology, EDBT '12, Berlin, Germany, March 27-30, 2012, Proceedings*, pages 204–215. ACM, 2012. doi:10.1145/2247596.2247622.
 - 51 Camille Bourgaux, Patrick Koopmann, and Anni-Yasmin Turhan. Ontology-mediated query answering over temporal and inconsistent data. *Semantic Web*, 10(3):475–521, 2019. doi:10.3233/SW-180337.
 - 52 Sebastian Brandt, Elem Güzel Kalayci, Vladislav Ryzhikov, Guohui Xiao, and Michael Zakharyashev. Querying log data with metric temporal logic. *J. Artif. Intell. Res.*, 62:829–877, 2018. doi:10.1613/jair.1.11229.
 - 53 Gerhard Brewka, Thomas Eiter, and Miroslaw Truszczynski. Answer set programming at a glance. *Commun. ACM*, 54(12):92–103, 2011. doi:10.1145/2043174.2043195.
 - 54 Jean-Paul Calbimonte, Davide Calvaresi, and Michael Schumacher. Multi-agent interactions on the web through linked data notifications. In *Multi-Agent Systems and Agreement Technologies: 15th European Conference, EUMAS 2017, and 5th International Conference, AT 2017, Evry, France, December 14-15, 2017, Revised Selected Papers 15*, pages 44–53. Springer, 2018. doi:10.1007/978-3-030-01713-2_4.
 - 55 Jean-Paul Calbimonte, Oscar Corcho, and Alasdair JG Gray. Enabling ontology-based access to streaming data sources. In *The Semantic Web-ISWC 2010: 9th International Semantic Web Conference, ISWC 2010, Shanghai, China, November 7-11, 2010, Revised Selected Papers, Part I 9*, pages 96–111. Springer, 2010. doi:10.1007/978-3-642-17746-0_7.
 - 56 Jean-Paul Calbimonte, Hoyoung Jeung, Oscar Corcho, and Karl Aberer. Enabling query technologies for the semantic sensor web. *International Journal On Semantic Web and Information Systems (IJSWIS)*, 8(1):43–63, 2012. doi:10.4018/jswis.2012010103.
 - 57 Jean-Paul Calbimonte, José Mora, and Óscar Corcho. Query rewriting in RDF stream processing. In *ESWC*, pages 486–502, 2016. doi:10.1007/978-3-319-34129-3_30.
 - 58 Francesco Calimeri, Marco Manna, Elena Mastria, Maria Concetta Morelli, Simona Perri, and Jessica Zangari. I-DLV-sr: A stream reasoning system based on I-DLV. *Theory Pract. Log. Program.*, 21(5):610–628, 2021. doi:10.1017/S147106842100034X.
 - 59 Paris Carbone, Marios Fragkoulis, Vasiliki Kalavri, and Asterios Katsifodimos. Beyond analytics: The evolution of stream processing systems. In *SIGMOD. ACM*, 2020. doi:10.1145/3318464.3383131.
 - 60 Paris Carbone, Asterios Katsifodimos, Stephan Ewen, Volker Markl, Seif Haridi, and Kostas Tzoumas. Apache flink: Stream and batch processing in a single engine. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, 36(4), 2015. URL: <http://sites.computer.org/debull/A15dec/p28.pdf>.
 - 61 Stefano Ceri, Georg Gottlob, and Letizia Tanca. *Logic Programming and Databases*. Surveys in computer science. Springer, 1990. URL: <https://www.worldcat.org/oclc/20595273>.
 - 62 Sirish Chandrasekaran and Michael J. Franklin. Streaming queries over streaming data. In *VLDB*, pages 203–214. Morgan Kaufmann, 2002. doi:10.1016/B978-155860869-6/50026-3.
 - 63 Jiaoyan Chen, Freddy Lecue, Jeff Z. Pan, and Huajun Chen. Learning from ontology streams with semantic concept drift. In *Proceedings of the Twenty-Sixth International Joint Conference on*

- Artificial Intelligence, IJCAI-17*, pages 957–963, 2017. doi:10.24963/ijcai.2017/133.
- 64 Michael Compton, Payam Barnaghi, Luis Bermudez, Raul Garcia-Castro, Oscar Corcho, Simon Cox, John Graybeal, Manfred Hauswirth, Cory Henson, Arthur Herzog, et al. The ssn ontology of the w3c semantic sensor network incubator group. *Journal of Web Semantics*, 17:25–32, 2012. doi:10.1016/j.websem.2012.05.003.
- 65 Simon Cox and Chris Little. Time ontology in OWL. Technical report, W3C, Spatial Data on the Web Working Group, 2022. W3C Candidate Recommendation Draft, Nov 25, 2022, <https://www.w3.org/TR/owl-time/>. URL: <https://www.w3.org/TR/owl-time/>.
- 66 David J. Tena Cucala, Przemyslaw Andrzej Walega, Bernardo Cuenca Grau, and Egor V. Kostylev. Stratified negation in datalog with metric temporal operators. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 6488–6495. AAAI Press, 2021. doi:10.1609/AAAI.V35I7.16804.
- 67 Gianpaolo Cugola and Alessandro Margara. Processing flows of information: From data stream to complex event processing. *ACM Comput. Surv.*, 44(3):15:1–15:62, 2012. doi:10.1145/2187671.2187677.
- 68 Gianpaolo Cugola, Alessandro Margara, Matteo Matteucci, and Giordano Tamburrelli. Introducing uncertainty in complex event processing: model, implementation, and validation. *Computing*, 97(2):103–144, 2015. doi:10.1007/s00607-014-0404-y.
- 69 Richard Cyganiak, David Wood, and Markus Lanthaler. RDF 1.1 Concepts and Abstract Syntax. W3C Recommendation, W3C, 2014. URL: <https://www.w3.org/TR/rdf11-concepts/>.
- 70 Ariyam Das, Sahil M. Gandhi, and Carlo Zaniolo. ASTRO: A datalog system for advanced stream reasoning. In *CIKM*, pages 1863–1866, 2018. doi:10.1145/3269206.3269223.
- 71 Mathias De Brouwer, Pieter Bonte, Dörthe Arndt, Miel Vander Sande, Pieter Heyvaert, Anastasia Dimou, Ruben Verborgh, Filip De Turck, and Femke Ongenaë. Distributed continuous home care provisioning through personalized monitoring & treatment planning. In *Companion Proceedings of the Web Conference 2020*, pages 143–147, 2020. doi:10.1145/3366424.3383528.
- 72 Daniel de Leng and Fredrik Heintz. DyKnow: A dynamically reconfigurable stream reasoning framework as an extension to the robot operating system. In *SIMPAR*, pages 55–60. IEEE, 2016. doi:10.1109/SIMPAR.2016.7862375.
- 73 Daniel de Leng and Fredrik Heintz. Partial-state progression for stream reasoning with metric temporal logic. In *Sixteenth International Conference on Principles of Knowledge Representation and Reasoning*, 2018. URL: <https://aaai.org/ocs/index.php/KR/KR18/paper/view/17988>.
- 74 Daniel de Leng and Fredrik Heintz. Approximate stream reasoning with metric temporal logic under uncertainty. In *AAAI*, pages 2760–2767, 2019. doi:10.1609/aaai.v33i01.33012760.
- 75 Emanuele Della Valle. *On Stream Reasoning*. PhD Thesis, Vrije Universiteit Amsterdam, 2015. URL: <https://research.vu.nl/en/publications/on-stream-reasoning>.
- 76 Emanuele Della Valle, Irene Celino, Daniele Dell’Aglío, Ralph Grothmann, Florian Steinke, and Volker Tresp. Semantic traffic-aware routing using the larkc platform. *IEEE Internet Comput.*, 15(6):15–23, 2011. doi:10.1109/MIC.2011.107.
- 77 Daniele Dell’Aglío and Abraham Bernstein. Differentially private stream processing for the semantic web. In *WWW*, pages 1977–1987. ACM / IW3C2, 2020. doi:10.1145/3366423.3380265.
- 78 Daniele Dell’Aglío, Emanuele Della Valle, Jean-Paul Calbimonte, and Óscar 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. doi:10.4018/ijswis.2014100102.
- 79 Daniele Dell’Aglío, Emanuele Della Valle, et al. Incremental reasoning on rdf streams, 2014. doi:10.1201/B16859-22.
- 80 Daniele Dell’Aglío, Danh Le Phuoc, Anh Le-Tuan, Muhammad Intizar Ali, and Jean-Paul Calbimonte. On a web of data streams. In *Proceedings of the workshop on decentralizing the semantic Web 2017 co-located with 16th International Semantic Web Conference (ISWC 2017)*. 22 October 2017, 2017. URL: <https://ceur-ws.org/Vol-1934/contribution-11.pdf>.
- 81 Daniele Dell’Aglío, Jean-Paul Calbimonte, Marco Balduini, Oscar Corcho, and Emanuele Della Valle. On correctness in rdf stream processor benchmarking. In *The Semantic Web–ISWC 2013: 12th International Semantic Web Conference, Sydney, NSW, Australia, October 21-25, 2013, Proceedings, Part II 12*, pages 326–342. Springer, 2013. doi:10.1007/978-3-642-41338-4_21.
- 82 Daniele Dell’Aglío, Minh Dao-Tran, Jean-Paul Calbimonte, Danh Le Phuoc, and Emanuele Della Valle. A query model to capture event pattern matching in rdf stream processing query languages. In *European Knowledge Acquisition Workshop*, pages 145–162. Springer, 2016. doi:10.1007/978-3-319-49004-5_10.
- 83 Anastasia Dimou, Miel Vander Sande, Pieter Colpaert, Ruben Verborgh, Erik Mannens, and Rik Van de Walle. Rml: A generic language for integrated rdf mappings of heterogeneous data. *Ldow*, 1184, 2014. URL: https://ceur-ws.org/Vol-1184/ldow2014_paper_01.pdf.
- 84 Gregory Ditzler and Robi Polikar. Incremental Learning of Concept Drift from Streaming Imbalanced Data. *IEEE Trans. Knowl. Data Eng.*, 25(10):2283–2301, 2013. doi:10.1109/TKDE.2012.136.
- 85 Patrick Doherty and Jonas Kvarnström. Temporal action logics. In Frank van Harmelen, Vladimir Lifschitz, and Bruce W. Porter, editors, *Handbook of Knowledge Representation*, volume 3 of *Foundations of Artificial Intelligence*, pages 709–757. Elsevier, 2008. doi:10.1016/S1574-6526(07)03018-0.

- 86 Patrick Doherty, Jonas Kvarnström, and Fredrik Heintz. A temporal logic-based planning and execution monitoring framework for unmanned aircraft systems. *Auton. Agent Multi-Ag.*, 19(3):332–377, 2009. doi:10.1007/s10458-009-9079-8.
- 87 Manh Nguyen Duc, Anh Lê Tuán, Manfred Hauswirth, and Danh Le Phuoc. Towards autonomous semantic stream fusion for distributed video streams. In Alessandro Margara, Emanuele Della Valle, Alexander Artikis, Nesime Tatbul, and Helge Parzyjeglá, editors, *15th ACM International Conference on Distributed and Event-based Systems, DEBS 2021, Virtual Event, Italy, June 28 - July 2, 2021*, pages 172–175. ACM, 2021. doi:10.1145/3465480.3467837.
- 88 Martin Dürst and Michel Suignard. Internationalized resource identifiers (IRIs). Technical report, W3C, Internationalization Working Group, 2005. doi:10.17487/RFC3987.
- 89 Thomas Eiter, Ryutaro Ichise, Josiane Xavier Parreira, Patrik Schneider, and Lihua Zhao. Deploying spatial-stream query answering in C-ITS scenarios. *Semantic Web*, 12(1):41–77, 2021. doi:10.3233/SW-200408.
- 90 Thomas Eiter and Rafael Kiesel. Weighted LARS for quantitative stream reasoning. In *ECAI*, pages 729–736, 2020. doi:10.3233/FAIA200160.
- 91 Thomas Eiter and Rafael Kiesel. Semiring reasoning frameworks in AI and their computational complexity. *J. Artif. Intell. Res.*, 77:207–293, 2023. doi:10.1613/jair.1.13970.
- 92 Thomas Eiter, Paul Ogris, and Konstantin Schekotihin. A distributed approach to LARS stream reasoning (system paper). *TPLP*, 19(5-6):974–989, 2019. doi:10.1017/S1471068419000309.
- 93 Emanuele Falzone, Riccardo Tommasini, Emanuele Della Valle, Petra Selmer, Stefan Plantikow, Hannes Voigt, Keith Hare, Ljubica Lazarevic, and Tobias Lindaaker. Semantic foundations of seraph continuous graph query language. *arXiv preprint arXiv:2111.09228*, 2021. doi:10.48550/arXiv.2111.09228.
- 94 Javier D Fernández, Alejandro Llavés, and Oscar Corcho. Efficient rdf interchange (eri) format for rdf data streams. In *The Semantic Web-ISWC 2014: 13th International Semantic Web Conference, Riva del Garda, Italy, October 19-23, 2014. Proceedings, Part II 13*, pages 244–259. Springer, 2014. doi:10.1007/978-3-319-11915-1_16.
- 95 Michael Fisher. Temporal representation and reasoning. In Frank van Harmelen, Vladimir Lifschitz, and Bruce W. Porter, editors, *Handbook of Knowledge Representation*, volume 3 of *Foundations of Artificial Intelligence*, pages 513–550. Elsevier, 2008. doi:10.1016/S1574-6526(07)03012-X.
- 96 Michael Fisher, Dov M. Gabbay, and Lluís Vila, editors. *Handbook of Temporal Reasoning in Artificial Intelligence*, volume 1 of *Foundations of Artificial Intelligence*. Elsevier, 2005. doi:10.5555/2974992.
- 97 João Gama, Pedro Medas, Gladys Castillo, and Pedro Pereira Rodrigues. Learning with Drift Detection. In *SBIA*, volume 3171 of *LNCS*, pages 286–295. Springer, 2004. doi:10.1007/978-3-540-28645-5_29.
- 98 João Gama, Raquel Sebastião, and Pedro Pereira Rodrigues. Issues in evaluation of stream learning algorithms. In *KDD*, pages 329–338. ACM, 2009. doi:10.1145/1557019.1557060.
- 99 Jing Gao, Wei Fan, Jiawei Han, and Philip Yu. A general framework for mining concept-drifting data streams with skewed distributions. In *Proceedings of the 2007 SIAM international conference on data mining*, pages 3–14, apr 2007. doi:10.1137/1.9781611972771.1.
- 100 Shen Gao, Thomas Scharrenbach, and Abraham Bernstein. The clock data-aware eviction approach: Towards processing linked data streams with limited resources. In *European Semantic Web Conference*, pages 6–20. Springer, 2014. doi:10.1007/978-3-319-07443-6_2.
- 101 Martin Gebser, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub. Multi-shot ASP solving with clingo. *TPLP*, 19(1):27–82, 2019. doi:10.1017/S1471068418000054.
- 102 Stefano Germano, Thu-Le Pham, and Alessandra Mileo. Web stream reasoning in practice: On the expressivity vs. scalability tradeoff. In Balder ten Cate and Alessandra Mileo, editors, *Web Reasoning and Rule Systems - 9th International Conference, RR 2015, Berlin, Germany, August 4-5, 2015, Proceedings*, volume 9209 of *Lecture Notes in Computer Science*, pages 105–112. Springer, 2015. doi:10.1007/978-3-319-22002-4_9.
- 103 Federico Giannini, Giacomo Ziffer, and Emanuele Della Valle. cpnn: Continuous progressive neural networks for evolving streaming time series. In Hisashi Kashima, Tsuyoshi Idé, and Wen-Chih Peng, editors, *Advances in Knowledge Discovery and Data Mining - 27th Pacific-Asia Conference on Knowledge Discovery and Data Mining, PAKDD 2023, Osaka, Japan, May 25-28, 2023, Proceedings, Part IV*, volume 13938 of *Lecture Notes in Computer Science*, pages 328–340. Springer, 2023. doi:10.1007/978-3-031-33383-5_26.
- 104 Nikos Giatrakos, Alexander Artikis, Antonios Deligiannakis, and Minos N. Garofalakis. Complex event recognition in the big data era. *Proc. VLDB Endow.*, 10(12):1996–1999, 2017. doi:10.14778/3137765.3137829.
- 105 Boris Glavic, Kyumars Sheykh Esmaili, Peter M. Fischer, and Nesime Tatbul. Efficient stream provenance via operator instrumentation. *ACM Trans. Internet Techn.*, 14(1):7:1–7:26, 2014. doi:10.1145/2633689.
- 106 Boris Glavic, Kyumars Sheykh Esmaili, Peter Michael Fischer, and Nesime Tatbul. Ariadne: managing fine-grained provenance on data streams. In Sharma Chakravarthy, Susan Darling Urban, Peter R. Pietzuch, and Elke A. Rundensteiner, editors, *The 7th ACM International Conference on Distributed Event-Based Systems, DEBS '13, Arlington, TX, USA - June 29 - July 03, 2013*, pages 39–50. ACM, 2013. doi:10.1145/2488222.2488256.
- 107 Heitor Murilo Gomes, Albert Bifet, Jesse Read, Jean Paul Barddal, Fabrício Enembreck, Bernhard Pfahringer, Geoff Holmes, and Talel Abdesslem. Adaptive random forests for evolving data stream

- classification. *Mach. Learn.*, 106(9-10):1469–1495, 2017. doi:10.1007/s10994-017-5642-8.
- 108 Ian J. Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. An Empirical Investigation of Catastrophic Forgetting in Gradient-Based Neural Networks. In *2nd International Conference on Learning Representations*, 2015-03-03. doi:10.48550/arXiv.1312.6211.
- 109 Felipe Gorostiaga and César Sánchez. Hstriver: a very functional extensible tool for the runtime verification of real-time event streams. In *International Symposium on Formal Methods*, pages 563–580. Springer, 2021. doi:10.1007/978-3-030-90870-6_30.
- 110 OWL Working group. OWL 2 web ontology language overview (second edition). Technical report, W3C, dec 2012. W3C recommendation. URL: <http://www.w3.org/TR/2012/REC-owl2-overview-20121211/>.
- 111 Aakansha Gupta and Rahul Katarya. Social media based surveillance systems for healthcare using machine learning: A systematic review. *J. Biomed. Inform.*, 108:103500, 2020. doi:10.1016/j.jbi.2020.103500.
- 112 Fredrik Heintz, Jonas Kvarnström, and Patrick Doherty. Bridging the sense-reasoning gap: Dknow - stream-based middleware for knowledge processing. *Adv. Eng. Inform.*, 24(1):14–26, 2010. doi:10.1016/j.aei.2009.08.007.
- 113 Fredrik Heintz, Jonas Kvarnström, and Patrick Doherty. Stream-based reasoning support for autonomous systems. In Helder Coelho, Rudi Studer, and Michael J. Wooldridge, editors, *ECAI 2010 - 19th European Conference on Artificial Intelligence, Lisbon, Portugal, August 16-20, 2010, Proceedings*, volume 215 of *Frontiers in Artificial Intelligence and Applications*, pages 183–188. IOS Press, 2010. doi:10.3233/978-1-60750-606-5-183.
- 114 Pieter Heyvaert, Ben De Meester, Anastasia Dimou, and Ruben Verborgh. Declarative rules for linked data generation at your fingertips! In *European Semantic Web Conference*, pages 213–217. Springer, 2018. doi:10.1007/978-3-319-98192-5_40.
- 115 Martin Hirzel, Guillaume Baudart, Angela Bonifati, Emanuele Della Valle, Sherif Sakr, and Akrivi Vlachou. Stream processing languages in the big data era. *SIGMOD Record*, 47(2):29–40, 2018. doi:10.1145/3299887.3299892.
- 116 Ian M. Hodkinson and Mark Reynolds. Temporal logic. In Patrick Blackburn, J. F. A. K. van Benthem, and Frank Wolter, editors, *Handbook of Modal Logic*, volume 3 of *Studies in logic and practical reasoning*, pages 655–720. North-Holland, 2007. doi:10.1016/s1570-2464(07)80014-0.
- 117 Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *The collected works of Wassily Hoeffding*, pages 409–426, 1994. doi:10.1007/978-1-4612-0865-5_26.
- 118 Pan Hu, Boris Motik, and Ian Horrocks. Modular materialisation of Datalog programs. *Artif. Intell.*, 308:103726, 2022. doi:10.1016/j.artint.2022.103726.
- 119 Giovambattista Ianni, Francesco Pacenza, and Jessica Zangari. Incremental maintenance of overgrounded logic programs with tailored simplifications. *Theory Pract. Log. Program.*, 20(5):719–734, 2020. doi:10.1017/S147106842000040X.
- 120 Elena Ikonomovska, João Gama, and Saso Dzeroski. Learning model trees from evolving data streams. *Data Min. Knowl. Discov.*, 23(1):128–168, 2011. doi:10.1007/s10618-010-0201-y.
- 121 Neil Immerman. *Descriptive complexity*. Graduate texts in computer science. Springer, 1999. doi:10.1007/978-1-4612-0539-5.
- 122 Sebastian Kabisch, Daniel Peintner, and Darko Anicic. Standardized and efficient rdf encoding for constrained embedded networks. In *European Semantic Web Conference*, pages 437–452. Springer, 2015. doi:10.1007/978-3-319-18818-8_27.
- 123 Antonis C. Kakas. Abduction. In Claude Sammut and Geoffrey I. Webb, editors, *Encyclopedia of Machine Learning*, pages 3–9. Springer, 2010. doi:10.1007/978-0-387-30164-8_1.
- 124 Andreas Kamilaris, Feng Gao, Francesc X. Prenafeta-Boldu, and Muhammad Intizar Ali. Agri-iot: A semantic framework for internet of things-enabled smart farming applications. In *3rd IEEE World Forum on Internet of Things, WF-IoT 2016, Reston, VA, USA, December 12-14, 2016*, pages 442–447. IEEE Computer Society, 2016. doi:10.1109/WF-IoT.2016.7845467.
- 125 Evgeny Kharlamov, Yannis Kotidis, Theofilos Mailis, Christian Neuenstadt, Charalampos Nikolaou, Özgür L. Özçep, Christoforos Svingos, Dmitriy Zheleznyakov, Sebastian Brandt, Ian Horrocks, Yannis E. Ioannidis, Steffen Lamparter, and Ralf Möller. Towards analytics aware ontology based access to static and streaming data. In *ISWC (2)*, volume 9982 of *Lecture Notes in Computer Science*, pages 344–362, 2016. doi:10.1007/978-3-319-46547-0_31.
- 126 Houda Khrouf, Badre Belabess, Laurent Bihanic, Gabriel Kepekian, and Olivier Curé. Waves: Big data platform for real-time rdf stream processing. In Daniele Dell’Aglio, Emanuele Della Valle, Markus Krötzsch, Thomas Eiter, Maria Maleshkova, Ruben Verborgh, Federico M. Facca, and Michael Mrissa, editors, *Joint Proceedings of the 3rd Stream Reasoning (SR 2016) and the 1st Semantic Web Technologies for the Internet of Thing (SWIT 2016) workshops (SR+SWIT)*, number 1783 in CEUR Workshop Proceedings, pages 37–48, Aachen, 2016. URL: <https://ceur-ws.org/Vol-1783/paper-04.pdf>.
- 127 James Kirkpatrick, Razvan Pascanu, Neil C. Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13), 2017. doi:10.1073/pnas.1611835114.
- 128 Maxim Kolchin, Peter Wetz, Elmar Kiesling, and A Min Tjoa. Yabench: A comprehensive framework for rdf stream processor correctness and performance assessment. In *Web Engineering: 16th International Conference, ICWE 2016*,

- Lugano, Switzerland, June 6-9, 2016. *Proceedings 16*, pages 280–298. Springer, 2016. doi:10.1007/978-3-319-38791-8_16.
- 129 Nicolas Kourtellis, Gianmarco De Francisci Morales, Albert Bifet, and Arinto Murdopo. VHT: vertical hoeffding tree. In James Joshi, George Karypis, Ling Liu, Xiaohua Hu, Ronay Ak, Yinglong Xia, Weijia Xu, Aki-Hiro Sato, Sudarsan Rachuri, Lyle H. Ungar, Philip S. Yu, Rama Govindaraju, and Toyotaro Suzumura, editors, *2016 IEEE International Conference on Big Data (IEEE BigData 2016), Washington DC, USA, December 5-8, 2016*, pages 915–922. IEEE Computer Society, 2016. doi:10.1109/BIGDATA.2016.7840687.
 - 130 Ron Koymans. Specifying real-time properties with metric temporal logic. *Real-Time Syst.*, 2(4):255–299, 1990. doi:10.1007/BF01995674.
 - 131 Jeffrey R Lacasse and Eileen Gambrill. Making assessment decisions: Macro, mezzo, and micro perspectives. *Critical thinking in clinical assessment and diagnosis*, pages 69–84, 2015. doi:10.1007/978-3-319-17774-8_4.
 - 132 Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ales Leonardis, Gregory G. Slabaugh, and Tinne Tuytelaars. A Continual Learning Survey: Defying Forgetting in Classification Tasks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 44(7):3366–3385, 2022. doi:10.1109/TPAMI.2021.3057446.
 - 133 Danh Le-Phuoc, Minh Dao-Tran, Minh-Duc Pham, Peter Boncz, Thomas Eiter, and Michael Fink. Linked stream data processing engines: Facts and figures. In *International Semantic Web Conference*, pages 300–312. Springer, 2012. doi:10.1007/978-3-642-35173-0_20.
 - 134 Danh Le-Phuoc, Minh Dao-Tran, Josiane Xavier Parreira, and Manfred Hauswirth. A native and adaptive approach for unified processing of linked streams and linked data. In *International Semantic Web Conference*, pages 370–388. Springer, 2011. doi:10.1007/978-3-642-25073-6_24.
 - 135 Danh Le-Phuoc, Hoan Nguyen Mau Quoc, Chan Le Van, and Manfred Hauswirth. Elastic and scalable processing of linked stream data in the cloud. In *International Semantic Web Conference*, pages 280–297. Springer, 2013. doi:10.1007/978-3-642-41335-3_18.
 - 136 Anh Le-Tuan, Manh Nguyen-Duc, Chien-Quang Le, Trung-Kien Tran, Manfred Hauswirth, Thomas Eiter, and Danh Le-Phuoc. Cqels 2.0: Towards a unified framework for semantic stream fusion. *arXiv preprint arXiv:2202.13958*, 2022. doi:10.48550/arXiv.2202.13958.
 - 137 Freddy Lécué. Towards scalable exploration of diagnoses in an ontology stream. In Carla E. Brodley and Peter Stone, editors, *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada*, pages 87–93. AAAI Press, 2014. doi:10.1609/aaai.v28i1.8708.
 - 138 Freddy Lécué and Jeff Z. Pan. Predicting knowledge in an ontology stream. In *IJCAI*, pages 2662–2669. IJCAI/AAAI, 2013. doi:10.5555/2540128.2540512.
 - 139 Freddy Lécué, Simone Tallevi-Diotalleivi, Jer Hayes, Robert Tucker, Veli Bicer, Marco Luca Sbodio, and Pierpaolo Tommasi. Smart traffic analytics in the semantic web with STAR-CITY: scenarios, system and lessons learned in dublin city. *J. Web Semant.*, 27-28:26–33, 2014. doi:10.1016/j.websem.2014.07.002.
 - 140 Maurizio Lenzerini. Data integration: A theoretical perspective. In Lucian Popa, Serge Abiteboul, and Phokion 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. doi:10.1145/543613.543644.
 - 141 Timothée Lesort, Vincenzo Lomonaco, Andrei Stoian, Davide Maltoni, David Filliat, and Natalia Díaz Rodríguez. Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges. *Inf. Fusion*, 58:52–68, 2020. doi:10.1016/j.inffus.2019.12.004.
 - 142 Zhizhong Li and Derek Hoiem. Learning Without Forgetting. In *ECCV (4)*, volume 9908 of *Lecture Notes in Computer Science*, pages 614–629. Springer, 2016. doi:10.1007/978-3-319-46493-0_37.
 - 143 Zhiqiu Lin, Jia Shi, Deepak Pathak, and Deva Ramanan. The CLEAR benchmark: Continual learning on real-world imagery. In Joaquin Vanschoren and Sai-Kit Yeung, editors, *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*, 2021. URL: <https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/2838023a778dfaecd212708f721b788-Abstract-round2.html>.
 - 144 Mo Liu, Ming Li, Denis Golovnya, Elke A. Rundensteiner, and Kajal T. Claypool. Sequence pattern query processing over out-of-order event streams. In Yannis E. Ioannidis, Dik Lun Lee, and Raymond T. Ng, editors, *Proceedings of the 25th International Conference on Data Engineering, ICDE 2009, March 29 2009 - April 2 2009, Shanghai, China*, pages 784–795. IEEE Computer Society, 2009. doi:10.1109/ICDE.2009.95.
 - 145 Vincenzo Lomonaco, Davide Maltoni, and Lorenzo Pellegrini. Rehearsal-Free Continual Learning over Small Non-I.I.D. Batches. In *CVPR Workshops*, pages 989–998. Computer Vision Foundation / IEEE, 2020. doi:10.1109/CVPRW50498.2020.00131.
 - 146 Vincenzo Lomonaco, Lorenzo Pellegrini, Andrea Cossu, Antonio Carta, Gabriele Graffieti, Tyler L. Hayes, Matthias De Lange, Marc Masana, Jary Pomponi, Gido van de Ven, Martin Mundt, Qi She, Keiland Cooper, Jeremy Forest, Eden Belouadah, Simone Calderara, German I. Parisi, Fabio Cuzzolin, Andreas Tolia, Simone Scardapane, Luca Antiga, Subutai Amhad, Adrian Popescu, Christopher Kanan, Joost van de Weijer, Tinne Tuytelaars, Davide Bacciu, and Davide Maltoni. Avalanche: an end-to-end library for continual learning. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2nd Con-

- tinual Learning in Computer Vision Workshop, 2021. doi:10.1109/CVPRW53098.2021.00399.
- 147 David Lopez-Paz and Marc'Aurelio Ranzato. Gradient Episodic Memory for Continual Learning. In *NIPS*, pages 6467–6476, 2017. doi:10.5555/3295222.3295393.
 - 148 Jie Lu, Anjin Liu, Fan Dong, Feng Gu, João Gama, and Guangquan Zhang. Learning under Concept Drift: A Review. *IEEE Trans. Knowl. Data Eng.*, 31(12):2346–2363, 2019. doi:10.1109/TKDE.2018.2876857.
 - 149 Robin Manhaeve, Sebastijan Dumancic, Angelika Kimmig, Thomas Demeester, and Luc De Raedt. Neural probabilistic logic programming in deep-probrog. *Artif. Intell.*, 298:103504, 2021. doi:10.1016/j.artint.2021.103504.
 - 150 Andrea Mauri, Jean-Paul Calbimonte, Daniele Dell'Aglia, Marco Balduini, Marco Brambilla, Emanuele Della Valle, and Karl Aberer. Triple-Wave: Spreading RDF Streams on the Web. In *International Semantic Web Conference (2)*, volume 9982 of *Lecture Notes in Computer Science*, pages 140–149. Springer, 2016. doi:10.1007/978-3-319-46547-0_15.
 - 151 Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier, 1989. doi:10.1016/S0079-7421(08)60536-8.
 - 152 Alessandra Mileo, Ahmed Abdelrahman, Sean Polcarpio, and Manfred Hauswirth. Streamrule: A nonmonotonic stream reasoning system for the semantic web. In *RR*, pages 247–252, 2013. doi:10.1007/978-3-642-39666-3_23.
 - 153 Alessandra Mileo, Minh Dao-Tran, Thomas Eiter, and Michael Fink. Stream reasoning. In Ling Liu and M. Tamer Özsu, editors, *Encyclopedia of Database Systems, Second Edition*. Springer, 2018. doi:10.1007/978-1-4614-8265-9_80715.
 - 154 Jacob Montiel, Max Halford, Saulo Martiello Mastelini, Geoffrey Bolmier, Raphaël Sourty, Robin Vaysse, Adil Zouitine, Heitor Murilo Gomes, Jesse Read, Talel Abdesslem, and Albert Bifet. River: machine learning for streaming data in python. *J. Mach. Learn. Res.*, 22:110:1–110:8, 2021. URL: <http://jmlr.org/papers/v22/20-1380.html>.
 - 155 Gianmarco De Francisci Morales and Albert Bifet. SAMOA: scalable advanced massive online analysis. *J. Mach. Learn. Res.*, 16:149–153, 2015. doi:10.5555/2789272.2789277.
 - 156 Boris Motik, Yavor Nenov, Robert Piro, and Ian Horrocks. Maintenance of datalog materialisations revisited. *Artif. Intell.*, 269:76–136, 2019. doi:10.1016/j.artint.2018.12.004.
 - 157 Boris Motik, Yavor Nenov, Robert Piro, Ian Horrocks, and Dan Olteanu. Parallel Materialisation of Datalog Programs in Centralised, Main-Memory RDF Systems. In Carla E. Brodley and Peter Stone, editors, *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada*, pages 129–137. AAAI Press, 2014. doi:10.1609/aaai.v28i1.8730.
 - 158 Esteban Municio, Glenn Daneels, Mathias De Brouwer, Femke Ongenaes, Filip De Turck, Bart Braem, Jeroen Famaey, and Steven Latré. Continuous athlete monitoring in challenging cycling environments using iot technologies. *IEEE Internet of Things Journal*, 6(6):10875–10887, 2019. doi:10.1109/JIOT.2019.2942761.
 - 159 S. Muthukrishnan. Data streams: Algorithms and applications. *Found. Trends Theor. Comput. Sci.*, 1(2), 2005. doi:10.1561/0400000002.
 - 160 Yavor Nenov, Robert Piro, Boris Motik, Ian Horrocks, Zhe Wu, and Jay Banerjee. RD-Fox: A Highly-Scalable RDF Store. In Marcelo Arenas, Óscar Corcho, Elena Simperl, Markus Strohmaier, Mathieu d'Aquin, Kavitha Srinivas, Paul Groth, Michel Dumontier, Jeff Heflin, Krishnaprasad Thirunarayan, and Steffen Staab, editors, *The Semantic Web - ISWC 2015 - 14th International Semantic Web Conference, Bethlehem, PA, USA, October 11-15, 2015, Proceedings, Part II*, volume 9367 of *Lecture Notes in Computer Science*, pages 3–20. Springer, 2015. doi:10.1007/978-3-319-25010-6_1.
 - 161 Matthias Nickles and Alessandra Mileo. Web stream reasoning using probabilistic answer set programming. In Roman Kontchakov and Marie-Laure Mugnier, editors, *Web Reasoning and Rule Systems - 8th International Conference, RR 2014, Athens, Greece, September 15-17, 2014. Proceedings*, volume 8741 of *Lecture Notes in Computer Science*, pages 197–205. Springer, 2014. doi:10.1007/978-3-319-11113-1_16.
 - 162 Matthias Nickles and Alessandra Mileo. A hybrid approach to inference in probabilistic non-monotonic logic programming. In Fabrizio Riguzzi and Joost Vennekens, editors, *Proceedings of the 2nd International Workshop on Probabilistic Logic Programming co-located with 31st International Conference on Logic Programming (ICLP 2015), Cork, Ireland, August 31st, 2015*, volume 1413 of *CEUR Workshop Proceedings*, pages 57–68. CEUR-WS.org, 2015. URL: <https://ceur-ws.org/Vol-1413/paper-05.pdf>.
 - 163 Matthias Nickles and Alessandra Mileo. A system for probabilistic inductive answer set programming. In Christoph Beierle and Alex Dekhtyar, editors, *Scalable Uncertainty Management - 9th International Conference, SUM 2015, Québec City, QC, Canada, September 16-18, 2015. Proceedings*, volume 9310 of *Lecture Notes in Computer Science*, pages 99–105. Springer, 2015. doi:10.1007/978-3-319-23540-0_7.
 - 164 Philipp Obermeier, Javier Romero, and Torsten Schaub. Multi-shot stream reasoning in answer set programming: A preliminary report. *OJDB*, 6(1):33–38, 2019. URL: https://www.ronpub.com/ojdb/OJDB_2019v6i1n04_Obermeier.html.
 - 165 Michiel Overeem, Marten Spoor, and Slinger Jansen. The dark side of event sourcing: Managing data conversion. In *2017 IEEE 24th international conference on software analysis, evolution and reengineering (SANER)*, pages 193–204. IEEE, 2017. doi:10.1109/SANER.2017.7884621.
 - 166 Özgür Lütü Özçep, Ralf Möller, and Christian Neuenstadt. A stream-temporal query language for ontology based data access. In *KI 2014: Advances in Artificial Intelligence: 37th Annual German Conference on AI, Stuttgart, Ger-*

- many, September 22–26, 2014. *Proceedings 37*, pages 183–194. Springer, 2014. doi:10.1007/978-3-319-11206-0_18.
- 167 Özgür Lütfü Özçep, Ralf Möller, and Christian Neuenstadt. Stream-query compilation with ontologies. In Bernhard Pfahringer and Jochen Renz, editors, *AI 2015: Advances in Artificial Intelligence - 28th Australasian Joint Conference, Canberra, ACT, Australia, November 30 - December 4, 2015, Proceedings*, volume 9457 of *Lecture Notes in Computer Science*, pages 457–463. Springer, 2015. doi:10.1007/978-3-319-26350-2_40.
- 168 Anil Pacaci, Angela Bonifati, and M. Tamer Özsu. Regular path query evaluation on streaming graphs. In David Maier, Rachel Pottinger, An-Hai Doan, Wang-Chiew Tan, Abdussalam Alawini, and Hung Q. Ngo, editors, *Proceedings of the 2020 International Conference on Management of Data, SIGMOD Conference 2020, online conference [Portland, OR, USA], June 14–19, 2020*, pages 1415–1430. ACM, 2020. doi:10.1145/3318464.3389733.
- 169 Anil Pacaci, Angela Bonifati, and M. Tamer Özsu. Evaluating complex queries on streaming graphs. In *38th IEEE International Conference on Data Engineering, ICDE 2022, Kuala Lumpur, Malaysia, May 9–12, 2022*, pages 272–285. IEEE, 2022. doi:10.1109/ICDE53745.2022.00025.
- 170 Dimitris Palyvos-Giannas, Vincenzo Gulisano, and Marina Papatriantafidou. Genealog: Fine-grained data streaming provenance at the edge. In *Proceedings of the 19th International Middleware Conference*, pages 227–238, 2018. doi:10.1145/3274808.3274826.
- 171 Dimitris Palyvos-Giannas, Bastian Havers, Marina Papatriantafidou, and Vincenzo Gulisano. Ananke: A streaming framework for live forward provenance. *Proc. VLDB Endow.*, 14(3):391–403, 2020. doi:10.5555/3430915.3442437.
- 172 Dimitris Palyvos-Giannas, Katerina Tzompanaki, Marina Papatriantafidou, and Vincenzo Gulisano. Erebus: Explaining the outputs of data streaming queries. *Proc. VLDB Endow.*, 16(2):230–242, 2022. doi:10.14778/3565816.3565825.
- 173 Douglas S. Parker. Integrating AI and DBMS through stream processing. In *ICDE*, pages 259–260, 1989. doi:10.1109/ICDE.1989.47224.
- 174 Stott D. Parker. Stream data analysis in prolog. In *The Practice of Prolog*, pages 249–301. MIT Press, 2003.
- 175 Harshal Patni, Cory Henson, and Amit Sheth. Linked sensor data. In *2010 International Symposium on Collaborative Technologies and Systems*, pages 362–370. IEEE, 2010. doi:10.1109/CTS.2010.5478492.
- 176 Lorenzo Pellegrini, Gabriele Graffieti, Vincenzo Lomonaco, and Davide Maltoni. Latent Replay for Real-Time Continual Learning. In *IROS*, pages 10203–10209. IEEE, 2020. doi:10.1109/IROS45743.2020.9341460.
- 177 Romana Pernisch, Daniele Dell’Aglia, and Abraham Bernstein. Beware of the hierarchy — An analysis of ontology evolution and the materialisation impact for biomedical ontologies. *Journal of Web Semantics*, 70:100658, jul 2021. doi:10.1016/j.websem.2021.100658.
- 178 Thu-Le Pham, Muhammad Intizar Ali, and Alessandra Mileo. C-ASP: Continuous ASP-based reasoning over RDF streams. In *LPNMR*, pages 45–50, 2019. doi:10.1007/978-3-030-20528-7_4.
- 179 Thu-Le Pham, Muhammad Intizar Ali, and Alessandra Mileo. Enhancing the scalability of expressive stream reasoning via input-driven parallelization. *Semantic Web*, 10(3):457–474, 2019. doi:10.3233/SW-180330.
- 180 Thu-Le Pham, Alessandra Mileo, and Muhammad Intizar Ali. Towards scalable non-monotonic stream reasoning via input dependency analysis. In *33rd IEEE International Conference on Data Engineering, ICDE 2017, San Diego, CA, USA, April 19–22, 2017*, pages 1553–1558. IEEE Computer Society, 2017. doi:10.1109/ICDE.2017.226.
- 181 Danh Le Phuoc, Thomas Eiter, and Anh Lê Tuán. A scalable reasoning and learning approach for neural-symbolic stream fusion. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2–9, 2021*, pages 4996–5005. AAAI Press, 2021. doi:10.1609/aaai.v35i6.16633.
- 182 Amir Pnueli. The temporal logic of programs. In *FOCS*, pages 46–57, 1977. doi:10.1109/SFCS.1977.32.
- 183 Ameya Prabhu, Philip H. S. Torr, and Puneet K. Dokania. GDumb: A Simple Approach that Questions Our Progress in Continual Learning. In *ECCV (2)*, volume 12347 of *Lecture Notes in Computer Science*, pages 524–540. Springer, 2020. doi:10.1007/978-3-030-58536-5_31.
- 184 Luc De Raedt, Angelika Kimmig, and Hannu Toivonen. Problog: A probabilistic prolog and its application in link discovery. In Manuela M. Veloso, editor, *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6–12, 2007*, pages 2462–2467, 2007. URL: <http://ijcai.org/Proceedings/07/Papers/396.pdf>.
- 185 Kristo Raun, Riccardo Tommasini, and Ahmed Awad. I will survive: An event-driven conformance checking approach over process streams. In Valerio Schiavoni, Marcelo Pasin, Bettina Kemme, and Etienne Rivière, editors, *Proceedings of the 17th ACM International Conference on Distributed and Event-based Systems, DEBS 2023, Neuchatel, Switzerland, June 27–30, 2023*, pages 49–60. ACM, 2023. doi:10.1145/3583678.3596887.
- 186 Jesse Read, Albert Bifet, Bernhard Pfahringer, and Geoff Holmes. Batch-incremental versus instance-incremental learning in dynamic and evolving data. In *IDA*, volume 7619 of *Lecture Notes in Computer Science*, pages 313–323. Springer, 2012. doi:10.1007/978-3-642-34156-4_29.
- 187 Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. iCaRL: Incremental Classifier and Representation Learning. In *CVPR*, pages 5533–5542. IEEE Computer Society, 2017. doi:10.1109/CVPR.2017.587.

- 188 Raymond Reiter. On closed world data bases. In Hervé Gallaire and Jack Minker, editors, *Logic and Data Bases*, pages 55–76. Plenum Press, New York, 1978. doi:10.1007/978-1-4684-3384-5_3.
- 189 Xiangnan Ren and Olivier Curé. Strider: A hybrid adaptive distributed rdf stream processing engine. In *The Semantic Web–ISWC 2017: 16th International Semantic Web Conference, Vienna, Austria, October 21–25, 2017, Proceedings, Part I 16*, pages 559–576. Springer, 2017. doi:10.1007/978-3-319-68288-4_33.
- 190 Márcio Moretto Ribeiro. *Belief Revision in Non-Classical Logics*. Springer Briefs in Computer Science. Springer, 2013. doi:10.1007/978-1-4471-4186-0.
- 191 Alessandro Ronca, Mark Kaminski, Bernardo Cuenca Grau, and Ian Horrocks. The delay and window size problems in rule-based stream reasoning. *Artif. Intell.*, 306:103668, 2022. doi:10.1016/J.ARTINT.2022.103668.
- 192 Hans Rott. *Change, choice and inference: A study of belief revision and nonmonotonic reasoning*, volume 42 of *Oxford logic guides*. Oxford University Press, 2001.
- 193 Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive Neural Networks. *CoRR*, abs/1606.04671, 2016. doi:10.48550/arXiv.1606.04671.
- 194 Georgios M. Santipantakis, Akrivi Vlachou, Christos Doukeridis, Alexander Artikis, Ioannis Kontopoulos, and George A. Vouros. A Stream Reasoning System for Maritime Monitoring. In Natasha Alechina, Kjetil Nørnvåg, and Wojciech Penczek, editors, *25th International Symposium on Temporal Representation and Reasoning (TIME 2018)*, volume 120 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 20:1–20:17. Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.TIME.2018.20.
- 195 Matthias J. Sax, Guozhang Wang, Matthias Weidlich, and Johann-Christoph Freytag. Streams and tables: Two sides of the same coin. In Malú Castellanos, Panos K. Chrysanthis, Badrish Chandramouli, and Shimin Chen, editors, *Proceedings of the International Workshop on Real-Time Business Intelligence and Analytics, BIRTE 2018, Rio de Janeiro, Brazil, August 27, 2018*, pages 1:1–1:10. ACM, 2018. doi:10.1145/3242153.3242155.
- 196 Thomas Scharrenbach, Jacopo Urbani, Alessandro Margara, Emanuele Della Valle, and Abraham Bernstein. Seven commandments for benchmarking semantic flow processing systems. In Philipp Cimiano, Óscar Corcho, Valentina Presutti, Laura Hollink, and Sebastian Rudolph, editors, *The Semantic Web: Semantics and Big Data, 10th International Conference, ESWC 2013, Montpellier, France, May 26-30, 2013. Proceedings*, volume 7882 of *Lecture Notes in Computer Science*, pages 305–319. Springer, 2013. doi:10.1007/978-3-642-38288-8_21.
- 197 Patrik Schneider, Daniel Alvarez-Coello, Anh Le-Tuan, Manh Nguyen Duc, and Danh Le Phuoc. Stream reasoning playground. In Paul Groth, Maria-Esther Vidal, Fabian M. Suchanek, Pedro A. Szekely, Pavan Kapanipathi, Catia Pesquita, Hala Skaf-Molli, and Minna Tamper, editors, *The Semantic Web - 19th International Conference, ESWC 2022, Hersonissos, Crete, Greece, May 29 - June 2, 2022, Proceedings*, volume 13261 of *Lecture Notes in Computer Science*, pages 406–424. Springer, 2022. doi:10.1007/978-3-031-06981-9_24.
- 198 Jonathan Schwarz, Wojciech Czarnecki, Jelena Luketina, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress & Compress: A scalable framework for continual learning. In *ICML*, volume 80 of *Proceedings of Machine Learning Research*, pages 4535–4544. PMLR, 2018. URL: <http://proceedings.mlr.press/v80/schwarz18a.html>.
- 199 Mario Scrocca, Riccardo Tommasini, Alessandro Margara, Emanuele Della Valle, and Sherif Sakr. The kaiju project: enabling event-driven observability. In Julien Gascon-Samson, Kaiwen Zhang, Khuzaima Daudjee, and Bettina Kemme, editors, *14th ACM International Conference on Distributed and Event-based Systems, DEBS 2020, Montreal, Quebec, Canada, July 13-17, 2020*, pages 85–96. ACM, 2020. doi:10.1145/3401025.3401740.
- 200 Joan Serrà, Didac Suris, Marius Miron, and Alexandros Karatzoglou. Overcoming Catastrophic Forgetting with Hard Attention to the Task. In *ICML*, volume 80 of *Proceedings of Machine Learning Research*, pages 4555–4564. PMLR, 2018. URL: <http://proceedings.mlr.press/v80/serra18a.html>.
- 201 Qi She, Fan Feng, Xinyue Hao, Qihan Yang, Chuanlin Lan, Vincenzo Lomonaco, Xuesong Shi, Zhengwei Wang, Yao Guo, Yimin Zhang, Fei Qiao, and Rosa H M Chan. OpenLORIS-Object: A Robotic Vision Dataset and Benchmark for Lifelong Deep Learning. *arXiv*, pages 1–8, 2019-11. doi:10.48550/arXiv.1911.06487.
- 202 Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual Learning with Deep Generative Replay. In *NIPS*, pages 2990–2999, 2017. doi:10.5555/3294996.3295059.
- 203 Utkarsh Srivastava and Jennifer Widom. Flexible time management in data stream systems. In *PODS*, pages 263–274. ACM, 2004. doi:10.1145/1055558.1055596.
- 204 Heiner Stuckenschmidt, Stefano Ceri, Emanuele Della Valle, and Frank Van Harmelen. Towards expressive stream reasoning. In *Dagstuhl Seminar Proceedings*. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2010. doi:10.4230/DagSemProc.10042.4.
- 205 Jakob Suchan, Mehul Bhatt, and Srikrishna Varadarajan. Out of sight but not out of mind: An answer set programming based online abduction framework for visual sensemaking in autonomous driving. In Sarit Kraus, editor, *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 1879–1885. ijcai.org, 2019. doi:10.24963/IJCAI.2019/260.
- 206 Jakob Suchan, Mehul Bhatt, and Srikrishna Varadarajan. Commonsense visual sensemaking

- for autonomous driving - on generalised neurosymbolic online abduction integrating vision and semantics. *Artif. Intell.*, 299:103522, 2021. doi:10.1016/j.artint.2021.103522.
- 207 Dan Suci, Dan Olteanu, Christopher Ré, and Christoph Koch. *Probabilistic databases*. Springer Nature, 2022. doi:10.1007/978-3-031-01879-4.
- 208 Gábor Szárnyas, Brad Bebee, Altan Birlir, Alin Deutsch, George Fletcher, Henry A. Gabb, Denise Gosnell, Alastair Green, Zhihui Guo, Keith W. Hare, Jan Hidders, Alexandru Iosup, Atanas Kiryakov, Tomas Kovatchev, Xinsheng Li, Leonid Libkin, Heng Lin, Xiaojian Luo, Arnau Prat-Pérez, David Püroja, Shipeng Qi, Oskar van Rest, Benjamin A. Steer, Dávid Szakállas, Bing Tong, Jack Waudby, Mingxi Wu, Bin Yang, Wenyuan Yu, Chen Zhang, Jason Zhang, Yan Zhou, and Peter Boncz. The linked data benchmark council (ldbc): Driving competition and collaboration in the graph data management space. In *TPCTC*, 2023. doi:10.48550/arXiv.2307.04350.
- 209 Gözde Ayşe Tataroğlu Özbek. Decentralized stream reasoning agents. In *Proceedings of the 17th ACM International Conference on Distributed and Event-based Systems*, pages 203–206, 2023. doi:10.1145/3583678.3603286.
- 210 Mahbod Tavallae, Ebrahim Bagheri, Wei Lu, and Ali A. Ghorbani. A detailed analysis of the KDD CUP 99 data set. In *CISDA*, pages 1–6. IEEE, 2009. doi:10.1109/CISDA.2009.5356528.
- 211 Douglas B. Terry, David Goldberg, David A. Nichols, and Brian M. Oki. Continuous queries over append-only databases. In Michael Stonebraker, editor, *Proceedings of the 1992 ACM SIGMOD International Conference on Management of Data, San Diego, California, USA, June 2-5, 1992*, pages 321–330. ACM Press, 1992. doi:10.1145/130283.130333.
- 212 Artem Thofimov, Igor E. Kuralenok, Nikiga Marshalkin, and Boris Novikov. Delivery, consistency, and determinism: rethinking guarantees in distributed stream processing. *CoRR*, abs/1907.06250, 2019. doi:10.48550/arXiv.1907.06250.
- 213 Edward Thomas, Jeff Z. Pan, and Yuan Ren. Trowl: Tractable OWL 2 reasoning infrastructure. In Lora Aroyo, Grigoris Antoniou, Eero Hyvönen, Annette ten Teije, Heiner Stuckenschmidt, Liliana Cabral, and Tania Tudorache, editors, *The Semantic Web: Research and Applications, 7th Extended Semantic Web Conference, ESWC 2010, Heraklion, Crete, Greece, May 30 - June 3, 2010, Proceedings, Part II*, volume 6089 of *Lecture Notes in Computer Science*, pages 431–435. Springer, 2010. doi:10.1007/978-3-642-13489-0_38.
- 214 Mattias Tiger and Fredrik Heintz. Stream reasoning using temporal logic and predictive probabilistic state models. In *TIME*, pages 196–205, 2016. doi:10.1109/TIME.2016.28.
- 215 Mattias Tiger and Fredrik Heintz. Incremental reasoning in probabilistic signal temporal logic. *Int. J. Approx. Reason.*, 119:325–352, 2020. doi:10.1016/j.ijar.2020.01.009.
- 216 Riccardo Tommasini, Pieter Bonte, Femke Ongena, and Emanuele Della Valle. Rsp4j: an api for rdf stream processing. In *The Semantic Web: 18th International Conference, ESWC 2021, Virtual Event, June 6–10, 2021, Proceedings 18*, pages 565–581. Springer, 2021. doi:10.1007/978-3-030-77385-4_34.
- 217 Riccardo Tommasini, Pieter Bonte, Fabiano Spiga, and Emanuele Della Valle. *Streaming Linked Data: From Vision to Practice*. Springer Nature, 2023. doi:10.1007/978-3-031-15371-6.
- 218 Riccardo Tommasini, Davide Calvaresi, and Jean-Paul Calbimonte. Stream reasoning agents: Blue sky ideas track. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1664–1680, 2019. doi:10.5555/3306127.3331894.
- 219 Riccardo Tommasini, Emanuele Della Valle, Andrea Mauri, and Marco Brambilla. Rsplab: Rdf stream processing benchmarking made easy. In *The Semantic Web-ISWC 2017: 16th International Semantic Web Conference, Vienna, Austria, October 21-25, 2017, Proceedings, Part II 16*, pages 202–209. Springer, 2017. doi:10.1007/978-3-319-68204-4_21.
- 220 Transaction Processing Performance Council (TPC). *TPC-H Benchmark Specification*, 2023. URL: <http://www.tpc.org/tpch/>.
- 221 Georgia Troullinou, Haridimos Kondylakis, Matteo Lissandrini, and Davide Mottin. SOFOS: Demonstrating the Challenges of Materialized View Selection on Knowledge Graphs. In *Proceedings of the 2021 International Conference on Management of Data, SIGMOD/PODS '21*, pages 2789–2793, New York, NY, USA, 2021. Association for Computing Machinery. event-place: Virtual Event, China. doi:10.1145/3448016.3452765.
- 222 Efthymia Tsamoura, David Carral, Enrico Malizia, and Jacopo Urbani. Materializing Knowledge Bases via Trigger Graphs. *Proc. VLDB Endow.*, 14(6):943–956, 2021. doi:10.14778/3447689.3447699.
- 223 Jacopo Urbani, Markus Krötzsch, and Thomas Eiter. Chasing streams with existential rules. In Gabriele Kern-Isberner, Gerhard Lakemeyer, and Thomas Meyer, editors, *Proceedings of the 19th International Conference on Principles of Knowledge Representation and Reasoning (KR 2022)*, pages 416–421. IJCAI, 2022. URL: <https://proceedings.kr.org/2022/43/>.
- 224 Nithya N Vijayakumar and Beth Plale. Towards low overhead provenance tracking in near real-time stream filtering. In *International provenance and annotation workshop*, pages 46–54. Springer, 2006. doi:10.1007/11890850_6.
- 225 Raphael Volz, Steffen Staab, and Boris Motik. Incrementally Maintaining Materializations of Ontologies Stored in Logic Databases. *J. Data Semant.*, 2:1–34, 2005. doi:10.1007/978-3-540-30567-5_1.
- 226 Przemysław A. Walega, Mark Kaminski, and Bernardo Cuenca Grau. Reasoning over streaming data in metric temporal datalog. In *AAAI*, pages 3092–3099, 2019. doi:10.1609/aaai.v33i01.33013092.
- 227 Przemysław A Walega, Mark Kaminski, Dingmin Wang, and Bernardo Cuenca Grau. Stream reasoning with DatalogMTL. *Journal of Web Semantics*, 76:100776, 2023. doi:10.1016/j.websem.2023.100776.

- 228 Przemyslaw Andrzej Walega, David J. Tena Cuccala, Bernardo Cuenca Grau, and Egor V. Kostylev. The stable model semantics of datalog with metric temporal operators. *Theory and Practice of Logic Programming*, pages 1–35, 2023. doi:10.1017/S1471068423000315.
- 229 Przemyslaw Andrzej Walega, David J. Tena Cuccala, Egor V. Kostylev, and Bernardo Cuenca Grau. Datalogmtl with negation under stable models semantics. In Meghyn Bienvenu, Gerhard Lakemeyer, and Esra Erdem, editors, *Proceedings of the 18th International Conference on Principles of Knowledge Representation and Reasoning, KR 2021, Online event, November 3-12, 2021*, pages 609–618, 2021. doi:10.24963/kr.2021/58.
- 230 Dingmin Wang, Pan Hu, Przemyslaw Andrzej Walega, and Bernardo Cuenca Grau. MeTeoR: Practical reasoning in datalog with metric temporal operators. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelfth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022*, pages 5906–5913. AAAI Press, 2022. doi:10.1609/AAAI.V36I5.20535.
- 231 Guozhang Wang, Lei Chen, Ayusman Dikshit, Jason Gustafson, Boyang Chen, Matthias J. Sax, John Roesler, Sophie Blee-Goldman, Bruno Cadonna, Apurva Mehta, Varun Madan, and Jun Rao. Consistency and completeness: Rethinking distributed stream processing in apache kafka. In Guoliang Li, Zhanhuai Li, Stratos Idreos, and Divesh Srivastava, editors, *SIGMOD '21: International Conference on Management of Data, Virtual Event, China, June 20-25, 2021*, pages 2602–2613. ACM, 2021. doi:10.1145/3448016.3457556.
- 232 Min Wang, Marion Blount, John Davis, Archan Misra, and Daby Sow. A time-and-value centric provenance model and architecture for medical event streams. In *Proceedings of the 1st ACM SIGMOBILE international workshop on Systems and networking support for healthcare and assisted living environments*, pages 95–100, 2007. doi:10.1145/1248054.1248082.
- 233 Yi Wang and Joohyung Lee. Handling uncertainty in answer set programming. In Blai Bonet and Sven Koenig, editors, *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*, pages 4218–4219. AAAI Press, 2015. doi:10.1609/aaai.v29i1.9726.
- 234 Zhun Yang, Adam Ishay, and Joohyung Lee. Neurasp: Embracing neural networks into answer set programming. In Christian Bessiere, editor, *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 1755–1762. ijcai.org, 2020. doi:10.24963/ijcai.2020/243.
- 235 Huaxiu Yao, Caroline Choi, Bochuan Cao, Yoonho Lee, Pang Wei Koh, and Chelsea Finn. Wildtime: A benchmark of in-the-wild distribution shift over time. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL: http://papers.nips.cc/paper_files/paper/2022/hash/43119db5d59f07cc08fca7ba6820179a-Abstract-Datasets_and_Benchmarks.html.
- 236 Carlo Zaniolo. Logical foundations of continuous query languages for data streams. In *Datalog*, pages 177–189, 2012. doi:10.1007/978-3-642-32925-8_18.
- 237 Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual Learning Through Synaptic Intelligence. In *International Conference on Machine Learning*, pages 3987–3995, 2017-07-17. doi:10.5555/3305890.3306093.
- 238 Shuhao Zhang, Juan Soto, and Volker Markl. A survey on transactional stream processing. *CoRR*, abs/2208.09827, 2022. doi:10.48550/arXiv.2208.09827.
- 239 Ying Zhang, Pham Minh Duc, Oscar Corcho, and Jean-Paul Calbimonte. Srbench: a streaming rdf/sparql benchmark. In *The Semantic Web—ISWC 2012: 11th International Semantic Web Conference, Boston, MA, USA, November 11-15, 2012, Proceedings, Part I 11*, pages 641–657. Springer, 2012. doi:10.1007/978-3-642-35176-1_40.
- 240 Giacomo Ziffer, Alessio Bernardo, Emanuele Della Valle, and Albert Bifet. Kalman filtering for learning with evolving data streams. In Yixin Chen, Heiko Ludwig, Yicheng Tu, Usama M. Fayyad, Xingquan Zhu, Xiaohua Hu, Suren Byna, Xiong Liu, Jianping Zhang, Shirui Pan, Vagelis Papalexakis, Jianwu Wang, Alfredo Cuzzocrea, and Carlos Ordonez, editors, *2021 IEEE International Conference on Big Data (Big Data)*, Orlando, FL, USA, December 15-18, 2021, pages 5337–5346. IEEE, 2021. doi:10.1109/BIGDATA52589.2021.9671365.
- 241 Giacomo Ziffer, Alessio Bernardo, Emanuele Della Valle, Vitor Cerqueira, and Albert Bifet. Towards time-evolving analytics: Online learning for time-dependent evolving data streams. *Data Science*, 6(1-2):1–16, 2022.
- 242 Indre Zliobaite, Albert Bifet, Jesse Read, Bernhard Pfahringer, and Geoff Holmes. Evaluation methods and decision theory for classification of streaming data with temporal dependence. *Mach. Learn.*, 98(3):455–482, 2015. doi:10.1007/s10994-014-5441-4.