Editor: **Steffen Staab,** University of Koblenz-Landau, staab@uni-koblenz.de

It's a Streaming World! Reasoning upon Rapidly Changing Information

Emanuele Della Valle and Stefano Ceri, *Politecnico di Milano*
Frank van Harmelen, *Vrije Universiteit Amsterdam*
Dieter Fensel, *University of Innsbruck*

Will there be a traffic jam on this highway? Can we reroute travelers on the basis of the forecast? By examining the clickstream from a given IP, can we discover shifts in interest of the person behind the computer? Which content on the news Web portal is attracting the most attention? Which navigation pattern would lead readers to other news related to that content? Do trends in medical records indicate any new disease spreading in a given part of the world? Where are all my friends meeting? Can we detect any intra-day correlation clusters among stock exchanges? What are the top 10 emerging topics under discussion in the blogosphere, and who is driving the discussions?

Although the information required to answer these questions is becoming increasingly available on the (Semantic) Web, there's currently no software system capable of computing the answers—indeed, no system even lets users issue such queries. The reason is straightforward: answering such queries requires systems that can manage rapidly changing worlds at the semantic level.

Of course, rapidly changing data can be analyzed on the fly by specialized *data-stream management systems,* but such systems can't perform complex reasoning tasks, and they lack a protocol to publish widely and to provide access to the rapidly changing data.

Reasoners, on the other hand, can perform such complex reasoning tasks, and the Semantic Web is providing the tools and methods to publish data widely on the Web. These technologies, however, don't really manage changing worlds: accessing and reasoning with rapidly changing information have been neglected or forgotten by their development communities.

The state of the art in reasoning over changing worlds is based on temporal logic and belief revision; these are heavyweight tools, suitable for data that changes in low volumes at low frequency. Similarly, the problem of changing vocabularies and evolving ontologies has undergone thorough investigation, but here the standard practice relies on configuration management techniques taken from software engineering, such as vocabulary and ontology versioning. These are suitable for ontologies that change on a weekly or monthly basis, but not for high-change-rate, high-frequency domains.

Moreover, the typical (Semantic) Web architecture, which caches all the information, can hardly be applied to rapidly changing information, because the crawled data would be obsolete at the time of querying.

We therefore suggest a completely different approach. *Stream reasoning,* an unexplored yet high-impact research area, is a new multidisciplinary approach that can provide the abstractions, foundations, methods, and tools required to integrate data streams, the Semantic Web, and reasoning systems, thus providing a way to answer our initial questions and many others.

In this column, we describe two concrete examples of stream-reasoning applications and introduce stream-reasoning research problems. We also present a list of research areas that we believe should be investigated to turn stream reasoning into a reality.

Stream-Reasoning Applications: Concrete Examples

Stream reasoning can benefit numerous areas: Traffic monitoring and traffic pattern detection appear to provide a natural application area; several independent works have already addressed this topic.[1–4] Other areas of interest are financial-transaction auditing,[5] wind power plant monitoring,[6] situation-aware mobile services,[7] and patient monitoring systems.[8] All the works we've just cited have come out of the Semantic Web community; researchers have been struggling with "classical" tools to solve streaming problems.

But if researchers turn their attention to stream reasoning, they will be able to create the methods and tools for easily answering the queries with which we began this column. Now let's look at two concrete applications for stream reasoning.

Reasoning for Mobile Applications

Mobility is one of the defining characteristics of modern life. Technology can support and accompany our mobility in several ways, both for business and entertainment. Mobile phones are popular and widespread; they provide good territory for challenging the stream-reasoning concept. To be immersed in our everyday life, mobile applications must fulfill real-time requirements, especially if we are to use them to make short-term decisions. Using data from sensors, which is likely to come in streams, mobile applications must find answers to the problems of reasoning with streams: coping with noisy data, dealing with errors, moving heavy reasoning computations to the server rather than doing them on the mobile devices, and so on.

Dealing with users' *stream of experience,* mobile applications must reason on what part of the streaming information is relevant and what its meaning is. For example, they have to abstract from quantitative information about latitude and longitude to qualitative information about places such as home, office, and gym. Then they have to raise the level of abstraction to reason about concrete problems, such as how to reach those places: means of transportation, a route that avoids traffic jams, and so on. By treating mobile phone users themselves as sensors, mobile applications could provide an understanding of the urban environment and its structure.

Monitoring Public-Health Risks

Early detection of potentially threatening public-health events such as outbreaks and epidemics is a major priority for national and international health-related organizations. Recent examples are infections such as SARS, the H5N1 avian flu, and the H1N1 virus. Dealing with this priority requires improving early detection capabilities—by enabling more timely and thorough relevant data acquisition and by advancing technologies for near real-time reporting and automated outbreak identification.

The challenge is to make the transition from handcrafted systems to automatic reasoning over data streams of similar magnitudes.

This requires an integrated public-health event detection platform that monitors a large variety of data to detect events and situations that might, when interpreted in the appropriate context, signify a potential threat to public health. Such a dynamic platform must identify, integrate, and interpret heterogeneous, distributed data streams, with information flowing from these data sources automatically analyzed and expressed on the basis of rich background knowledge. When this platform estimates an increased threat probability, it will have to streamline notifications to public-health bodies over various communication channels and deliver traces of the reasoning process and data that led to the calculation, so that authorities can evaluate and use the information appropriately.

Existing systems, such as Google's now classical Flu Trends, do indeed process high-volume streams of data, but all semantic processing of this data takes place in a predefined, hard-coded manner—a priori (when integrating streams) or a posteriori (when interpreting the results). The challenge is to make the transition from such handcrafted systems to automatic reasoning over data streams of similar magnitudes.

Problems to Be Solved

Several years ago, proposing a system to answer stream-reasoning questions would have sounded like science fiction, mainly because of the lack of data. Nowadays, a large amount of the required information is already available in digital format. For example, the mobile application we described earlier could use data such as maps with commercial activities and meeting places, events scheduled in the city and their locations, average speed in highways, positions and speed of public transportation vehicles, parking availabilities in specific parking areas, geo-positioned Twitter posts, user-generated media of any kind, blogs, clickstreams, and so on. Sources that are less subject to change can be made available through (Semantic) Web technologies, but for rapidly changing sources, the Semantic Web does not offer ready-to-use solutions.

To answer the questions we posed at the beginning of this column, a stream-reasoning system must be able to cope with several issues.

Lack of Theory for Stream Reasoning

Several theoretical aspects of stream reasoning have never been formalized. Missing elements include a model theory for stream reasoning; the formal properties of inference problems in stream reasoning beyond belief revision; a definition of soundness and completeness that takes into account the transient nature of streams; the general strategies for incremental reasoning beyond techniques known in the database area; a family of logic representation languages for streaming information; and a corresponding family of stream-reasoning algorithms.

Heterogeneous Formats and Access Protocols

Streams can appear in multiple forms, ranging from relation data over binary messaging protocols, such as data streams originated by sensor networks, to text streams over Web protocols, such as blogs and microblogs. Conversion and wrapping solutions already available for large-scale but static data sets (such as DB2RDF) will have to be developed for dynamic data streams.

Semantic Modeling

Semantic modeling of data streams involves several difficulties.

Window dependencies. A stream is observed through a window, which can be a span in time or a number of elements. Thus, such windows can contain incomplete information (for example, because some sensors did not provide data) or over-constrained information (for example, because different sensors observe the same event) about individuals.

Time dependencies. By their very nature, streams of data can be inspected only while they flow. If information is not captured and immediately summarized (aggregated, for example), then information reconstruction may be impossible.

Relationships between summarization and inference. Given that aggregation can perform lossy data compression, stream reasoning will require methods to determine which inferences are possible even after summarization and which must be performed before summarization.

If information is not captured and immediately summarized, information reconstruction may be impossible.

Merging with static information sources. Data streams are naturally timestamped, but the time validity of static information sources is normally not stated. Thus, merging data streams with static information can create hybrid data that must be carefully managed. Also needed are vocabularies to state future validity of information.

Learning from stream. If analyzed, the information that flows through the windows opened over streams, especially text streams, can determine changes in the static information sources. For example, new terms can emerge, and the number and nature of attributes describing an object can vary.

Scale

Scale is an issue for stream reasoning due both to the presence of huge data throughputs and to the need to link streaming data with large static knowledge bases. In many applications, a limited amount of data and knowledge is sufficient for a given stream-reasoning task. In these cases, the data should be sampled, abstracted, and approximated.

Continuous Processing

Stream reasoning requires continuous processing, because queries are normally registered and remain continuously active while data streams into the stream-reasoning system.

Real-Time Constraints

Stream-reasoning systems must provide answers before they become useless. Thus, the system must be able to provide incremental query-answering or reasoning, and it must also be able to tell the user whether it will deliver the answer to her continuous query on time.

Parallelization and Distribution

Stream-reasoning systems will require adaptation to novel hardware and software architectures that assign the reasoning tasks to independent computational units. This also requires controlling the reasoning process by modularizing reasoning and minimizing data transmission among the units.

Research Areas

We systematically analyzed the problems we presented in the previous section and have divided stream-reasoning research into five areas. We illustrate these new challenges by pointing to publications, particularly works presented at the First Stream Reasoning Workshop (SR 09), which took place in conjunction with the European Semantic Web Conference in May 2009.

```
 1. REGISTER QUERY TotalAmountPerBroker COMPUTE EVERY 10m AS
 2. PREFIX ex: <http://example/>
 3. SELECT DISTINCT ?broker ?total
 4. FROM <http://brokerscentral.org/brokers.rdf>
 5. FROM STREAM <http://stockex.org/market.trdf> [RANGE 1h STEP 10m]
 6. WHERE {
 7.   ?broker ex:from ?country .
 8.   ?broker ex:does ?tx .
 9.   ?tx ex:with ?amount .
10.   FILTER (?country = "CH" )
11. }
12. AGGREGATE { (?total, SUM(?amount), ?broker) }
```

Figure 1. C-SPARQL example. Continuous SPARQL (C-SPARQL) is an extension of the SPARQL protocol to support continuous queries by handling streams of RDF triples. In this example, given a static description of brokers and a stream of financial transactions for all Swiss brokers, the C-SPARQL engine computes the total of their transactions within the last hour.

Theory for Stream Reasoning

Stream-reasoning research definitely needs new theoretical investigations that go beyond data-stream management systems,[9] event-based systems,[10] and complex event processing.[11] Similarly, we must go beyond current existing theoretical frameworks such as belief revision[12] and temporal logic.[13] Existing theoretical frameworks either give a good basis for formal and explicit semantics, or they are appropriate for high-frequency, high-volume change rates. No current framework deals with both aspects simultaneously.

Examples of important theoretical problems that need investigation beyond these existing foundations are the following:

- dealing with incomplete or overconstrained information about individuals;[5]
- revising a notion of symbol grounding for temporal data;[2] and
- revising the traditional notions of soundness and completeness for stream reasoning.

Logic Language for Stream Reasoning

The investigation about whether a logic language is appropriate for stream reasoning is an important theoretical question. At the moment there is no agreement on the ways in which logic languages can integrate stream reasoning; the articles submitted to SR 09 adopted a variety of different logics. A *constructive description logic*[14] is at the core of Michael Mendler's and Stephan Scheele's work.[5] Matteo Palmonari and Davide Bogni propose a *commonsense spatial hybrid logic*.[1,15] *Metric temporal logic*[16] is the logical language of the DyKnow middleware.[2] Several other logics also appear to be valid starting points—for example, *temporal action logic*,[17] *step logic*,[18] and *active logic*.[19]

Stream Data Management for the Semantic Web

A first step toward stream reasoning is certainly trying to combine the power of existing data-stream management systems and the Semantic Web. The key idea is to keep streaming data in relational format as long as possible and bring it to the semantic level as aggregated events.[4] Existing data models, access protocols, and query languages for data-stream management systems and the Semantic Web are not sufficient to do so; they must be combined.

Streaming SPARQL[6] and Continuous SPARQL (C-SPARQL)[20] are two proposals for extending SPARQL to stream data management. Both of them introduce the notion of RDF streams as the natural extension of the RDF data model to this scenario, and then extend SPARQL to query RDF streams.

Figure 1 shows an example of a C-SPARQL query that, given a static description of brokers and a stream of financial transactions for all Swiss brokers, computes the total of their transactions within the last hour. At line 1, the `REGISTER` clause tells the C-SPARQL engine that it should register a continuous query—that is, a query that will continuously compute answers to the query. The `COMPUTE EVERY` clause states the frequency of every new computation. In line 5, the `FROM STREAM` clause defines the RDF stream of financial transactions used in the query. Next, line 6 defines the window of observation of the RDF stream. Streams, by their very nature, are volatile and so should be consumed on the fly; thus, the C-SPARQL engine observes them through a window that contains the stream's most recent elements and that changes over time. In the example, the window comprises RDF triples produced in the last hour, and the window slides every 10 minutes. The `WHERE` clause is standard; it includes a set of matching patterns and `FILTER` clauses. Finally, at line 12, the `AGGREGATE` function asks the C-SPARQL engine to include in the result set a new variable, `?total`, which is bound to the sum of the transaction amounts for each broker.

However, more investigation into stream-data management is needed. Following are some of the interesting research topics:

- semantic modeling of streams, extending the notion of RDF streams;
- protocols providing real-time access over the Web to streams;
- vocabularies to assert future validity of information;
- query language for RDF streams, extending approaches such as C-SPARQL and Streaming SPARQL;
- cost metrics to measure query plan cost and to predict whether enough time is available to provide an answer;

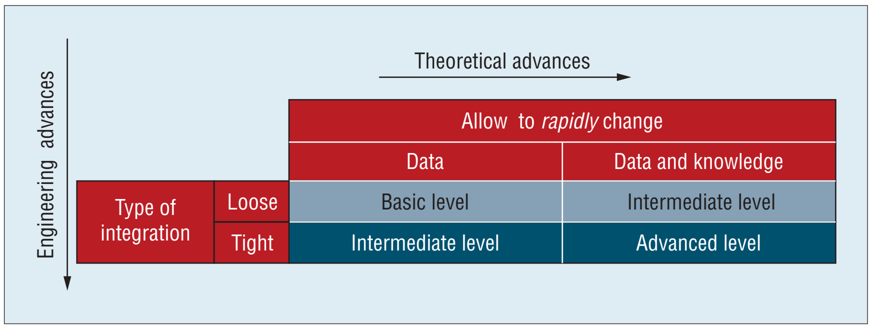

Figure 2. Basic schema for measuring progress in stream reasoning.

- query-rewriting techniques to leverage existing data-stream management systems;
- continuous query plan adaptation to the bursty nature of data streams;
- parallel processing of multiple queries to exploit interquery optimization opportunities; and
- distributed query processing, extending current ongoing efforts in SPARQL.

Stream Reasoning for the Semantic Web

Combining stream data management and reasoning at the data-model and query-language level is only a first step toward stream reasoning. From different viewpoints, part of this research has been conducted in AI under the name belief revision.[21] However, a well-developed notion of stream reasoning has not yet been proposed.

The central research question is this: can the idea of continuous semantics as introduced in data-stream management systems be extended to reasoners such as those currently being developed for the Semantic Web? For instance, does the current approach to materialization of ontological entailments still hold? Can an updated materialization be incrementally computed before it will be outdated? Last, but not least, can such techniques benefit from distribution and parallelization?

The state of the art in this area has its roots in deductive database research that considers how changes to facts should lead to changes in the extensions of logical predicates.[22] Pioneering work by Raphael Volz, Steffen Staab, and Boris Motik extends database techniques to changes to the ontology manifesting themselves in changes to the rules of the logic program.[23] Moreover, work presented at SR 09 includes very interesting attempts to answer these questions by the incremental evaluation of complex temporal formulas[2] and the incremental answering of reachability queries on streaming graphs.[3]

Engineering and Implementations

The three preceding sections of this article constitute our proposal for a research agenda for basic and applied research in stream reasoning. However, because the engineering of stream reasoning is clearly in its infancy, a large amount of engineering work will also be necessary to make concrete progress toward the stream-reasoning vision.

Several implemented stream-reasoning systems exist.[1,2,8] However, there have been only two systematic approaches attempted: DyKnow, which introduces the notion of primitive streams, stream generator, stream consumer, and stream processor;[2] and the work by Emanuele Della Valle and colleagues,[4] which applies to data streams the concepts of identification, selection, abstraction, and reasoning proposed in the LarKC approach.[24] Investigations into a conceptual architecture, concrete implementations, design environments, and an evaluation framework to compare and contrast different implementations are clearly still needed.

Measuring Progress

Although the problem of measuring progress in the stream-reasoning field may appear intractable at this stage, we believe that it is possible to assess the degree to which a stream reasoner meets different technical and application requirements.

On a general level, currently there are solutions for reasoning about static knowledge and solutions for handling streaming data. Therefore a basic requirement for a stream-reasoning system is to integrate these two aspects in a common approach that can perform reasoning on semantic streams. We can define progress relative to this integration qualitatively, as shown in Figure 2.

The simplest way of integrating the two aspects is in terms of a loose integration of reasoning and stream-data management: in a first step, a suitable data-stream management infrastructure acquires streaming data; and in a second step, reasoning is performed on the result of a view on this data. Although such integration doesn't exist today, the ambition of stream-reasoning research is to go beyond this simple case and create progress in two directions. On the engineering side, the goal of stream-reasoning research is to achieve a tight integration of reasoning and stream data management, where reasoning and stream querying are intertwined to improve performance. On the theoretical side, a goal of stream-reasoning research is to extend investigation to cases in which not only the instance data is provided in a streaming way, but also knowledge on the conceptual level is subject to frequent change.

Because it is not clear how to quantify progress in these two directions, we suggest indirectly measuring the degree of integration and the coverage of knowledge change by systematically evaluating stream-reasoning implementations against several well-defined quality criteria. As an initial set, we suggest the following:

- number of data streams handled simultaneously;
- update speed of the data streams (for example, in assertions per second);
- number of subscribed queries handled in parallel;
- number of query subscribers that must be notified; and
- time between event occurrence and notification of all subscribers.

Although the work we have discussed in this column grounds stream reasoning and gives it credence as an attainable goal, a huge amount of innovation will be necessary to make stream reasoning a reality.

Starting from lessons learned from databases (such as the ability to efficiently abstract and aggregate information out of multiple, high-throughput streams) and the Semantic Web (standard tools and methods for publishing and accessing structured data on the Web) we need a new foundational theory of stream reasoning, capable of associating reasoning tasks to time windows that delineate data validity and therefore produce time-varying inferences. From these foundations, we must derive new paradigms for knowledge representation—for example, extending RDF streams, query languages, and C-SPARQL. We will also need to deploy the consequent computational frameworks for stream-reasoning-oriented software architectures and their instrumentation.

This shift from static to highly dynamic data domains is not an incremental step. It will require substantial revisions to the very cornerstones on which the current Semantic Web is built.■

References

1. M. Palmonari and D. Bogni, "Commonsense Spatial Reasoning about Heterogeneous Events in Urban Computing," *Proc. 1st Int'l Workshop Stream Reasoning,* CEUR, 2009, http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-466.
2. F. Heintz, J. Kvarnstrom, and P. Doherty, "Stream Reasoning in DyKnow: A Knowledge Processing Middleware System," *Proc. 1st Int'l Workshop Stream Reasoning,* CEUR, 2009, http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-466.
3. G. Unel, F. Fischer, and B. Bishop, "Answering Reachability Queries on Streaming Graphs," *Proc. 1st Int'l Workshop Stream Reasoning,* CEUR, 2009, http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-466.
4. E. Della Valle et al., "A First Step towards Stream Reasoning," *Proc. Future Internet Symp.* (FIS 08), Springer, 2008, pp. 72–81.
5. M. Mendler and S. Scheele, "Towards a Type System for Semantic Streams," *Proc. 1st Int'l Workshop Stream Reasoning,* CEUR, 2009, http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-466.
6. A. Bolles, M. Grawunder, and J. Jacobi, "Streaming SPARQL—Extending SPARQL to Process Data Streams," *The Semantic Web: Research and Applications,* LNCS 5021, Springer, 2008, pp. 448–462.
7. M. Luther and S. Böhm, "Situation-Aware Mobility: An Application for Stream Reasoning," *Proc. 1st Int'l Workshop Stream Reasoning,* CEUR, 2009, http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-466.
8. P. Ordonez et al., "A Ubiquitous Context-Aware Environment for Surgical Training," *Proc. 4th Ann. Int'l Conf. Mobile and Ubiquitous Systems* (MobiQuitous 07), IEEE CS Press, 2007, pp. 1–6.
9. M. Garofalakis, J. Gehrke, and R. Rastogi, "Data Stream Management: Processing High-Speed Data Streams," *Data-Centric Systems and Applications,* Springer, 2007.
10. G. Mühl, L. Fiege, and P. Pietzuch, *Distributed Event-Based Systems,* Springer, 2006.
11. D. Luckham, *The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems,* Springer, 2008.
12. Peter Gärdenfors, ed., *Belief Revision,* Cambridge Univ. Press, 2003.
13. M. Risher, D.M. Gabbay, and Lluis Vila, eds., *Handbook of Temporal Reasoning in Artificial Intelligence,* Elsevier Science, 2005.
14. M. Mendler and S. Scheele, "Towards Constructive Description Logics for Abstraction and Refinement," *Proc. 21st Int'l Workshop Description Logics* (DL 08), CEUR, 2008, http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-353.
15. S. Bandini, A. Mosca, and M. Palmonari, "Common-Sense Spatial Reasoning for Information Correlation in Pervasive Computing," *Applied Artificial Intelligence,* vol. 21, nos. 4–5, 2007, pp. 405–425.
16. J. Ouaknine and J. Worrell, "Some Recent Results in Metric Temporal Logic," *Proc. 6th Int'l Conf. Formal Modeling and Analysis of Timed Systems* (Formats 08), LNCS 5215, Springer, 2008, pp. 1–13.
17. P. Doherty et al., "TAL: Temporal Action Logics Language Specification and Tutorial," *Electronic Trans. Artificial Intelligence,* vol. 2, nos. 3–4, 1998, pp. 273–306.
18. J.J. Elgot-Drapkin, *Step-Logic: Reasoning Situated in Time,* doctoral thesis,

Dept. of Computer Science, Univ. of Maryland, 1988.
19. J.J. Elgot-Drapkin et al., *Active Logics: A Unified Formal Approach to Episodic Reasoning,* tech report no. CS-TR-3680, Dept. of Computer Science, Univ. of Maryland, 1999.
20. D.F. Barbieri et al., "C-SPARQL: SPARQL for Continuous Querying," *Proc. 18th Int'l World Wide Web Conf.* (WWW 09), ACM Press, 2009, pp. 1061–1062.
21. A. Darwiche and J. Pearl, "On the Logic of Iterated Belief Revision," *Artificial Intelligence,* vol. 89, nos. 1–2, 1997, pp. 1–29.
22. M. Staudt and M. Jarke, "Incremental Maintenance of Externally Materialized Views," *Proc. 22th Int'l Conf. Very Large Data Bases,* Morgan Kaufmann, 1996, 75–86.
23. R. Volz, S. Staab, and B. Motik, "Incrementally Maintaining Materializations of Ontologies Stored in Logic Databases," *J. Data Semantics II,* vol. 3360, 2005, pp. 1–34.
24. D. Fensel et al., "Towards LarKC: A Platform for Web-Scale Reasoning," *Proc. IEEE Int'l Conf. Semantic Computing* (ICSC 08), IEEE Press, 2008, pp. 524–529.

Emanuele Della Valle is an assistant professor in the Department of Electronics and Information at Politecnico di Milano. Contact him at emanuele.dellavalle@polimi.it.

Stefano Ceri is a professor in the Department of Electronics and Information at Politecnico di Milano. Contact him at stefano.ceri@polimi.it.

Frank van Harmelen is a professor in the AI Department at Vrije Universiteit Amsterdam. Contact him at frank.van.harmelen@cs.vu.nl.

Dieter Fensel holds a professorship in the Semantic Technology Institute at the University of Innsbruck. Contact him at dieter.fensel@sti2.at.