STREAM REASONING

Alessandra Mileo[1], Minh Dao-Tran[2], Thomas Eiter[2], and Michael Fink[2]

[1] Insight Centre for Data Analytics
Dublin City University, Dublin
alessandra.mileo@insight-centre.org
http://www.insight-centre.org
[2] Institute of Information Systems
Vienna University of Technology
{dao,eiter,fink}@kr.tuwien.ac.at
http://www.kr.tuwien.ac.at

SYNONYMS

Continuous Reasoning
Reactive Reasoning

DEFINITION

Stream Reasoning refers to inference approaches and deduction mechanisms which are concerned with providing continuous inference capabilities over dynamic data. The paradigm shift from current batch-like approaches towards timely and scalable stream reasoning leverages the natural temporal order in data streams and applies windows-based processing to complex deduction tasks that go beyond continuous query processing such as those involving preferential reasoning, constraint optimization, planning, uncertainty, non-monotonicity, non-determinism, and solution enumeration.

HISTORICAL BACKGROUND

We are witnessing an unprecedented shift in the available quantity and quality of data drawn from all aspects of our lives, opening tremendous new opportunities but also significant challenges for scalable decision analytics due to its dynamicity. This makes it harder to go from data to *insight* and support effective decision making. Such transformation of these enormous quantities of information into actionable knowledge requires new reasoning methods which are able to remove the common assumption in scalable reasoning that knowledge bases are static or evolving slowly.

Stream Reasoning [**?**] emerged in the last few years as a new research area that aims at bridging the gap between reasoning and stream processing. Different communities have focused on complementary aspects of processing dynamic information, referred to as *stream processing* when closer to the data, and as *reasoning* when closer to knowledge and event management.

In Data Processing, the stream processing approach materializes in the ability to answer continuous queries over low-level, high rate input data such as those produced by (virtual or physical) sensors or social networks. Data Stream Management Systems (DSMSs) [**?**] considered requirements of applications where input data are unbounded and arrive in a streaming manner. Continuous queries are registered for execution when new input arrives, and various window operators are defined to target a deterministic query answer [**?**]. Corresponding solutions referred to as *Semantic Stream Processing* have been investigated for the Web of Data and RDF stream processing. Languages and solutions for processing Linked Stream Data (LSD) [**?**] are mostly based on extensions of the SPARQL query language with timestamps and operators for window-based query processing. The most widely used of such approaches are CQELS [**?**] and C-SPARQL [**?**]. In continuous query processing, incomplete information is rarely considered and not available in scalable implementations. Negation is also an important yet "expensive" construct and only rudimentary, simple forms of negation have been considered, which limits the expressive power of stream query processing approaches.

Complex Event Processing (CEP) emerged from publish-subscribe systems with the additional ability to express high-level events as complex patterns of incoming single events. Dealing with uncertainty is important in CEP, as information sources (usually sensors) are unreliable in general: data can be noisy or lost in communication, values can be rounded, etc. However, few works deal with this, and usually use probabilistic methods [**?**]. A survey on CEP approaches and tools can be found in [**?**].

In Knowledge Representation and Reasoning, recent works deal with streaming input, but lack scalability for high data rate. Nomonotonic Reasoning formalisms serve advanced reasoning from a knowledge base that deals with incomplete and/or inconsistent information, and reasoning under changing bases such as belief revision, reasoning about actions, logic programs update, agent systems etc., have been widely considered; however, streamed data arriving at high rate was not an issue. Recently, this has changed and initial works that study streaming have triggered investigations in this area, including ontology streams, streaming Answer Set Programming (ASP), incremental and reactive ASP and DSMSs with Datalog [**?**].

The performance of these approaches is closer to program updates than to high-speed stream reasoning. This has boosted interest in *stream reasoning* as the issue of providing models, theories and systems to bridge the gap between continuous query processing, CEP and reasoning, paying particular attention to the aspect of scalability.

SCIENTIFIC FUNDAMENTALS

In stream reasoning, input data dynamically arrive at the processing systems in form of possibly infinite streams. To deal with unboundedness of data, such systems typically apply *window operators* to obtain snapshots of recent data. Common windows operators (also used in window-based query processing and CEP) are:

- time-based windows: contain input data arriving within a fixed amount of time,
- tuple-based windows: contain the latest fixed number of input data,
- partition-based windows: split the input data based on different attributes and apply tuple-based windows on each substream.

Based on the finite data after the application of windows, the user specifies different reasoning tasks, typically continuous queries, to get output also in terms of streams. The key aspects of Stream Reasoning can be expressed as a set of requirements and related methods and approaches required for stream reasoning to fulfill its promise, as outlined in the remainder of this section.

Expressivity. Stream processing systems mainly adopt operational and monotonic semantics at the logical core. The latter is less suited to produce results when data is missing, and in particular not geared to deal with incorrect conclusions that must be retracted when more data is available. Such non-monotonic behavior is a key aspect in expressivity that is needed for Stream Reasoning.

Advanced reasoning techniques offer declarative means and model-based semantics to deal with incomplete information by (i) supplying different methods to complete data in a qualitative way and to provide different outcomes to the users, in terms of default or alternative answers; (ii) allowing to retract incorrect conclusions that were based on wrong assumptions.

Using designated, expressive language constructs, a developer can easier formulate, understand, and change the behavior of the reasoning component. Such features are extremely beneficial for reasoning on streaming data in a distributed setting, where the data can continuously change and get missing or is imprecise due to inaccurate computational models of the real world; default or rational conclusions are more useful than silence due to incomplete information.

Formal approaches of expressive reasoning to handle streaming data should comply with the following requirements:

- ability to handle time-changing data through models that cater for abstractions of temporally annotated data and efficient implementations of such abstractions;
- mechanisms to represent and reason about uncertainty associated to both data (missing information, noisy data) and reasoning processes (inconsistencies in deduction process, uncertain observations, propagation of uncertainty), leveraging declarative approaches to problem solving and combining quantitative and qualitative decision processes when needed;

– conflict handling techniques for both hard and soft constraints (also referred to as preferences);
– non-monotonic behavior in order to retract conclusions that are no longer valid.

Distribution and Heterogeneity. Challenges for the integration of distributed and heterogeneous data include issues such as data provenance and trust, as well as management and combination of multiple data formats in a single formalism. The need for integration of information that comes from multiple sources motivates the usage of data models like RDF that allow semantic integration, but this is not sufficient.

Handling streaming data requires the coordination of various processing/reasoning powers through physically distributed and arbitrarily connected reasoning nodes. Each node should be capable to offer a particular form of reasoning, including, but not limited to database querying, ontological reasoning, event pattern matching, continuous queries to stream processing systems, and model-building. Nodes should communicate with each other (with cycles allowed) in push- or pull-based manner, depending on the underlying supported processing/reasoning modes. Users place continuous queries at nodes providing this feature, and get live results via output streams. This offers various benefits to the users, as the ability to combine potentially different reasoning mechanisms to perform sophisticated tasks that a single stream processing formalism can not host.

For example, consider the following query posed by a tourist exploring a new city (which is similar to scenarios that stream processing approaches look at): "recommend a close DVD-shop offering a movie as a birthday gift for a boy aged 12, at a reasonable price." To answer this query, realistic applications must access distributed information sources in different formalisms, including: the city map, the traffic status, the user's GPS position, an ontology of the objects on the street, the websites of DVD-shops with available offers, a movie ontology. Moreover, different reasoning capabilities are needed: ontological reasoning to identify the shop from the map and to find suitable movies; reasoning about routes to find a close shop, respecting also accessibility via bike while avoiding congestion; and especially, reasoning about movie quality from price and information in the movie ontology, e.g., director, actors, etc.

At a global level, distribution requires to pay attention to synchronization and distributed processing by considering the issues of communication between nodes, global semantics of the system, and coordination of nodes to compute such a semantics. Some inspiration comes from Multi-Context Systems [**?**], which is an abstract framework to interlink possibly heterogeneous, non-monotonic knowledge bases via bridge rules for belief exchange (possibly in cycles). Their semantics is given by equilibria, which are stably interlinked local models. Distributed algorithms, optimizations techniques, and a prototype system DMCS[3] to compute equilibria top-down are available. As for data streaming, a timing concept can be attached to the belief exchange, with special management for continuous model building or query answering. The DMCS algorithms can serve as a starting point for bottom-up algorithms in a distributed streaming environment.

Scalability. There are several factors which challenge the performance of a stream processing/reasoning system. At the local level we have that: (1) the high input stream rate that exceed even high computation rate, (2) the input is dynamic, that is, selectivity and distribution of incoming data change from time to time, (3) when a window slides on a stream, consecutive window snapshots may overlap, (4) the reasoning tasks on top of the input data can have high complexity (see Expressivity above).

Current solutions in stream processing concentrate on addressing (1)–(3) on top of lightweight tasks (identified by, for example, the number of joins in queries). Traditional stream processing has efficient methods to deal with these issues. For example, adaptive evaluation adjusts computation strategies to change in dynamic input, incremental evaluation [**?**] avoids wasteful recomputation, and load shedding [**?**] reduces input while tolerating some Quality of Service.

Pushing for (4) faces the obstacle that solving complex reasoning tasks may take considerable time. Two observations, however, shows that advanced reasoning is still feasible:

– often expressive queries and complex reasoning tasks need only a high-level, abstract view of data instead of low-level input (e.g., GPS positions to regions, temperature to values like warm, cold, etc.).
– at the abstract level, changes often occur much less frequently than at the low level. A high-level reasoning algorithm has thus much more time available than a low-level algorithm.

[3] http://www.kr.tuwien.ac.at/research/systems/dmcs/

Abstraction techniques like data filtering or summarizing are available and can be readily used.

At the global level in a distributed setting, the main challenge is that global information such as the whole routing strategy or communication branches are invisible to nodes in the system. Each node merely knows parent and child nodes, and none has a clear picture of the system status to decide about the routing strategy, or to drop input/intermediate results at bottlenecks to improve global performance. To overcome this, the nodes need sophisticated methods to communicate enough information such that they can realize the need to adjust the global strategy and inform other nodes. Furthermore, a number of parameters needs a fine-grained investigation, namely, (i) the system size, i.e., the number of distributed nodes, (ii) the density and topology of the communication network, and (iii) the size of the knowledge base at each local node. The key aspect is to understand how these parameters scale while performance is still met; what is the limitation in increasing one parameter while fixing the others.

At the local and the global level, several reasoning tasks such as inconsistency checking, query answering, etc., and restrictions such as approximate semantics, syntactic restrictions, etc. are of interest. Investigating their computational complexity for various query languages will give hints on the theoretical scalability border. For example, with Datalog, depending on the version, the complexity ranges from AC_0 to Σ_2^P under well-founded and answer set semantics [**?**].

Benchmarking and comparison. The main stream processing/reasoning approaches have been independently proposed and few works compare them more extensively.

Linear Road [**?**] is currently the only complete benchmarking system for DSMSs, simulating an ad-hoc toll system for motor vehicle expressways. It uses L-Rating as a single metric to measure performance, i.e., the number of expressways a system can process while meeting time and correctness constraints. Other interesting aspects for comparison are still open, e.g., expressiveness, manageable query patterns, scalability under varying static data size, number of simultaneous queries, etc.

A preliminary work [**?**] proposed methods to cross-compare LSD processing engines on aspects such as functionality, correctness, and performance. They capture more basic evaluation tasks such as projection, filter, join, etc., rather than reasoning aspects, but can serve as a starting point for a more general comparison.

The lack of extensive comparison is due to several obstacles. On the practical side, comparing stream engines is non-trivial for several reasons: (i) they are based on different semantics which a benchmarking system must respect; (ii) there is pushed- vs. pull-based evaluation (also called data- vs. time-driven, eager vs. periodical evaluation): the former triggers computation when new input arrives, while the latter runs it periodically independent of input arrival; thus output may be missed (resp., repeated) when the input rate is too high (resp., slow) compared to the processing rate; (iii) the engines are fragile as any runtime deviation (e.g. processing delay) can lead to inaccurate output, especially for queries with aggregates; (iv) some engines are black boxes.

On the theoretical side, there is no formal foundations on which existing stream processing/reasoning approaches can be captured and thus compared systematically. On both the theoretical and the practical sides, the main issues are sufficiently generic measurements and metrics, along with methods to realize them for a fair comparison.

KEY APPLICATIONS

With increasing numbers of data streams becoming available, the amount of observable events to be captured and processed increases dramatically, resulting in new opportunities and challenges in sensor-rich and knowledge intensive environments. Some of the key application areas are listed in what follows.

Web of Things. The Web of Things is extremely dynamic: information is frequently changed and updated, and new data is continuously generated from a huge number of sources, often at high rate. Physical objects are made accessible as virtual sources of data streams, but the ability to select and combine them is crucial for exploiting their potential. Stream reasoning techniques that can cater for optimization and preferences help selecting and combining the best services and event sources that best contribute to a specific decision task.

Smart Cities. A smart city faces a huge volume of data that is coming from sensors monitoring the physical world in real time as well as from citizens through social streams. Processing such data streams in real time using prior knowledge of the domain (e.g. traffic, public safety) and the city (e.g. road topology) calls for stream reasoning. Traffic, public safety, healthcare, water, power grid, etc., can greatly benefit from stream reasoning providing real time insights into collective interactions within and between each of these systems.

Social Media Analysis. The large data volume produced by users in social networks and mobile applications is part of what is called *social sensing*. The latter contains information about user contexts, including the needs, interests, relations and activities of the users. Social Media Analysis aims at capturing this information and provide a characterization of the user context for context-aware applications. Social Media Contexts are highly dynamic but also characterized by noise and possible inconsistencies, and context properties are linked by complex relations. Stream reasoning techniques can provide mechanisms to analyze such changes, relate them with historical data and use them to discover interesting patterns.

Location-Aware Services and Mobile Applications. Location is an important aspect of contextual knowledge that highly characterize mobile applications. Utilizing physical and virtual sensors for reasoning about the location of a user is often perturbed by noise, by imprecise or missing sensor readings as well as inconsistent information coming from different sources. Stream reasoning can help in the combination of such sources with background knowledge and provide continuous location estimation based on semantic sensor fusion.

Smart Building and Resource Optimization. When smart building are equipped with sensors for water and energy consumption, usage and occupancy, stream reasoning techniques dealing with constraint optimization and preferential reasoning can substantially reduce cost associated with improper use of natural resources.

Emergency Management. Stream reasoning applied to dynamic planning can help providing real-time solutions to complex problems such as actuation of lights and door locks to guidance of people in evacuation plans. This can be done by reasoning about dynamic data describing the status of an emergency and its evolution in real-time (including the state of the environment and people location), combined with static knowledge about the floor plan.

FUTURE DIRECTIONS

Stream Reasoning is a very young topic of research for which the requirements listed in this entry has only partially been addressed.

The lack of a unified formal foundation for advanced reasoning with streaming data hinders the potential for expressive formalisms to be used in concrete frameworks, and investigation on multi-processing models to coordinate various reasoning posers in an advanced framework needs to be further investigated.

In terms of benchmarking, a direct cross-comparison is only possible for few engines.

A formal foundation for stream reasoning does not exist yet and is a mean for methods beyond those in [**?**] and a key to give a reference from which existing approaches can be captured as different ways of approximation, aiding their theoretical comparison. Based on the theoretical results, properties to assess stream reasoning solutions need to be proposed and measured for practical comparison.

Efficient approaches to stream reasoning should also explore the interplay between statistical analysis and knowledge-driven inference methods to have both a quantitative perspective on streaming data patterns and a qualitative perspective on complex structural properties of events, context of validity and logical correlations in decision processes.

CROSS REFERENCES

Window-based query processing
Complex Event Processing

RECOMMENDED READING