Undefined 0 (0) 1
IOS Press

Order Matters! Harnessing a World of Orderings for Reasoning over Massive Data

Editor(s): Pascal Hitzler, Kno.e.sis Center, Wright State University, Dayton, OH, USA; Krzysztof Janowicz, University of California, Santa Barbara, USA
Solicited review(s): Alessandra Mileo, DERI, National University of Ireland, Galway; Denny Vrandečić, Wikimedia Deutschland e.V., Germany; David Carral Martínez, Kno.e.sis Center, Wright State University, Dayton, OH, USA

Emanuele Della Valle [a], Stefan Schlobach [b], Markus Krötzsch [c], Alessandro Bozzon [a], Stefano Ceri [a], Ian Horrocks [c]
[a] *DEI, Politecnico di Milano*
[b] *Vrije Universiteit Amsterdam*
[c] *Univerity of Oxford*

Abstract. More and more applications require real-time processing of massive, dynamically generated, ordered data; order is an essential factor as it reflects recency or relevance. Semantic technologies risk being unable to meet the needs of such applications, as they are not equipped with the appropriate instruments for answering queries over massive, highly dynamic, ordered data sets. In this vision paper, we argue that some data management techniques should be exported to the context of semantic technologies, by integrating ordering with reasoning, and by using methods which are inspired by stream and rank-aware data management. We systematically explore the problem space, and point both to problems which have been successfully approached and to problems which still need fundamental research, in an attempt to stimulate and guide a paradigm shift in semantic technologies.

Keywords: Massive data, inference, ordering, streaming algorithms

1. Introduction

Data is massively produced and published at a speed which exceeds by far our current methods and infrastructure for processing it. Science and Engineering have become more and more data-driven: an environmental study of the earth atmosphere using digital telescopes requires collecting streams of measurements; the smooth pathway of satellites through space critically depends on the availability and analysis of detailed information about tiny objects in a flight path through hundreds of kilometers of space; a single simulation of an airplane engine easily produces terabytes of simulation results.

These examples have a common feature: *their data is ordered*. In some cases, data is naturally ordered by recency. Other data is intrinsically ordered, e.g., by precision, popularity, provenance, certainty, trust. In any case, data is explicitly sortable through attribute values such as latitude, longitude, object size, user-provided ratings, or frequency. Multiple orderings are simultaneously present for almost every available piece of information. Most answers are also required to come in an ordered fashion; for instance, engineers surveying a satellite orbit need to know the largest pieces of debris in closest proximity with maximal certainty, measured with highest precision; social scientists studying the Web want to study the most influential blogs, or the most recent tweets closest to a particular point of interest.

Another common property of the described problems is their *time-critical character*, requiring immediate answers at runtime: scientists and engineers want to adapt their expensive and complex experiments and simulations while running them based on analysis of incoming results, e.g., flight paths have to be adapted

once an object in collision course is detected, and companies need to know the effects of their commercial campaigns immediately. There is an immense need for reactive tools that critically depend on such runtime solutions.

Finally, all these problems require *inference*. For instance, engineers need to identify complex modelling errors in simulations, and classify them by severity, subjects in the flight path of a satellite need to be sorted according to their type, material, size, etc. Semantic applications must deal at the same time with rich ontological models describing complex domain knowledge, and highly dynamic data representing recent or relevant information, as produced by streaming or search-enabled data sources. State-of-the-art semantic technologies do not consider ordering as an essential property. Ranking results is often seen as an "added task," performed after inference, without affecting the inference process which is order-agnostic; as a result, semantic technologies cannot provide reactive and reliable query answering over such massive datasets, integrating highly dynamic sources; they don't scale in front of massive ordered data, and fail to be used in these problems and contexts.

However, the data management community has shown that the intrinsic "sorted" nature of data sources can be considered as an opportunity for building efficient data processing techniques, by harnessing ordering *before* processing, or by exploiting ordering *within* processing. Harnessing and exploiting orderings in reasoning gives the opportunity for significantly scaling up inferencing. We need a foundational theory, new generic methods and concrete algorithms for reasoning in the presence of orderings. This vision paper offers a systematic study of the solution space and of the challenges, both solved and unsolved, that the semantic community should face, taking advantage of methods which were defined in the data management context. We will identify classes of problems, describe prototypical examples, and indicate applicable approaches for each class. We will also identify the open challenges for the most difficult problems.

In principle the methods we propose are complete, i.e., return the correct answer. However, streaming algorithms lend themselves very naturally to approaches returning partial and approximate answers, with increasing quality over time. Progress in the context of reasoning over massive dynamic data can thus mean two things: runtime performance and quality performance. To make this more concrete, let *real-time* indicate the minimal time required to meet operational deadlines from event occurrence to system response [10].

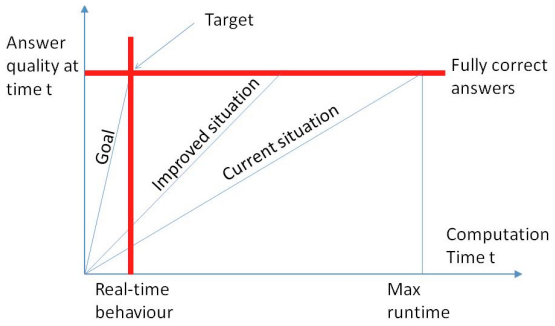

Fig. 1. Runtime performance and answer quality

Then, the behaviour of a system should be targeted towards exploiting all the inferences that can be performed in real-time, and the target is to reduce the runtime to real-time while retaining an acceptable level of answer quality. These concepts are visualized in Fig. 1.

In summary, the class of problems that can benefit from a tight integration of orderings and reasoning have the following properties:

- Data is massive.
- Data is ordered.
- Data can be incomplete, heterogeneous and noisy.
- Applications are time sensitive.
- Applications require inference.
- The analytical tasks require ordered answers.

In the remainder of the paper, we describe four concrete examples of time-sensitive applications that need to process massive sets of highly dynamic data (Section 2). We identify the problem and solution space that need to be investigated to implement our vision, and we argue that several areas in the problem space can only be addressed by making ordering a first class citizen in reasoning (Section 3). The core of the paper is (1) a systematic exploration of the solution space that points both to problems that have been successfully approached and to problems that still need fundamental research (Section 4), and (2) a discussion about the important role of approximation and parallelism in such space (Section 5).

2. Examples of applications

Hereafter, we present four concrete application problems for which order-aware reasoning can significantly boost scalability: three of them have their origin

in specific projects with industrial partners, the fourth is more general and will be used as a running example throughout the paper to explain our vision and ideas.

2.1. Space Situational Awareness

Since the start of human excursions into space, the amount of debris left in the atmosphere has exponentially increased. Typical examples of such debris are decommissioned satellites, parts of transportation rockets, but also include tools lost by astronauts, and most often wreckage from explosions or collisions in orbit. For satellites circling earth, collision with space debris has turned into a serious risk.[1] Space situational awareness is the problem of detecting these objects by monitoring the space using networks of radar and telescopic installations, and combining those results with existing debris databases. Those installations provide massive streams of measurements that are time and space bound, and can be ordered by reliability of the tools and precision of the observations.

Given that the task consists in finding debris in a restricted space and time window, and that human decisions strongly depend on a system's confidence, this is a typical target problem where order-aware reasoning makes a difference. For example, those streams can provide information whose real-time analysis could allow to adapt the flight path of a satellite to avoid collision, or even to shoot down debris.

2.2. Jet Engine Design

The construction of jet engines heavily depends on simulation. During those simulations, terabytes of data about flow fields, pressure, etc. are produced. In the current routine, such a simulation is performed in high performance computer centres and can last up to months. In a separate step, the data is analysed for design-errors, e.g., regarding the distance between rotor and engine boundary. Visualisation is used to detect deformations to rotor or stator, and to derive novel and more favourable design parameters. Given the complexity of running the simulations reactive adaptation of those parameters according to real-time analytics would be desirable.

Both analysis and visualisation are done offline and decoupled from the simulation, as analytic inference and data selection cannot yet be done efficiently enough for real-time processing. However, engineers perceive the need for effective analysis and visualisation of intermediate simulation results as soon as they are produced, and inference becomes a critical computational bottleneck. More concretely, tools are required that order intermediate results according to their importance for the visualisation process and their analytic importance (e.g., mesh quality problems). By dealing with simulation results at the time they are produced, jet engine design will be turned into a reactive process, which will save critical experimental time and efforts.

2.3. Intelligent Surveillance

Surveillance is the monitoring of behaviour, activities, or other information about groups of people. It is, e.g., employed to ensure the safety of workers on the factory floor, to detect crimes occurring in indoor or outdoor settings, or to monitor the flow of large crowds through public spaces. Current systems still require full involvement of human operators, which implies an high labour costs, limited capability for multiple screens, inconsistency in long-duration, etc. Most surveillance products on the market are based on vision and pattern recognition techniques, but industry and governments call for next generation intelligent surveillance systems that integrate such sensor data with social data streams. For larger cities, such as Seoul, the size of data generated each day from sensor networks exceed 500GB and 3 million tweets are posted each day. To make sense of all this information, extremely large amounts of geo-spatial data are also considered. In the workflow of real-time city surveillance scalable inferencing has been identified as an insurmountable obstacle.

Again, data naturally contains orderings. For instance, tweets can be ordered by popularity that can be estimated by the ratio of tweets and re-tweets, trustworthiness of information gathered from sensor networks can be ordered by precision of the sensor. Moreover, the final analytics returned to the interested party is necessarily based on choices that are related to those orderings; for instance, aggregating social sensing over a recent time window, a district of the city, age groups, or based on the most reliable information. Making use of the orderings in both data and information need typically calls for order-aware reasoning.

2.4. Social Media Analysis

The final example is about social media analysis, and will be used throughout the paper as running ex-

[1] http://www.space.com/11314-space-junk-satellite-collision-air-force.html

ample to motivate and explain our vision and ideas. Each second, thousands of tweets are produced worldwide, forming a rich body of information for companies and governments alike. Imagine a system which listens to all micro-posts that are published (on Twitter, Facebook, Google+, etc.), knows the geographic location of social media users, has the ability of detecting the topic of each micro-post, and has modelled relationships between topics in an expressive ontological language. Such system would be capable of serving a variety of information needs, e.g.:

> *Which users of social media, currently leading popular discussions on fashion-related topics, are closest to my current location? What are they saying about the shopping district nearby?*

Such a query describes a complex information need heavily depending on orderings in the data, and is thus a prototypical example for a problem requiring order-aware reasoning.

3. Inferencing with streaming algorithms

As mentioned in the introduction, research in efficient database querying indicates that many actual applications require order-aware queries, and that answering those queries can be highly efficient. Translated to the problem of inference over large-scale data, this means the following: whenever our applications require only those parts of the solution space that are optimal according to one or more ordering criteria, there is the opportunity to speed-up traditional inference methods by focusing on that part of the data that contributes to this significantly smaller solution space. For instance, when dealing with fast-changing data, efficiency can be gained through window-aware main memory processing of streaming data and indexing of static data, which allow efficient random access. Alternatively, when the information need relates to an ordering criterion, there is the opportunity to speed-up traditional inference methods by focusing on the ordered data and using order-aware operators.

Figure 2 shows a bird's-eye view of the order-aware reasoning vision. Data on the left is considered to be sorted – or sortable – according to some ordering. Such orderings can be *natural*, i.e., already present in the data (e.g., the recency in the micro-post stream), or they can be *enforced* for the purpose of a given application. Among the enforced orders, we further distinguish *cheap* from *expensive* ones. Cheap orderings

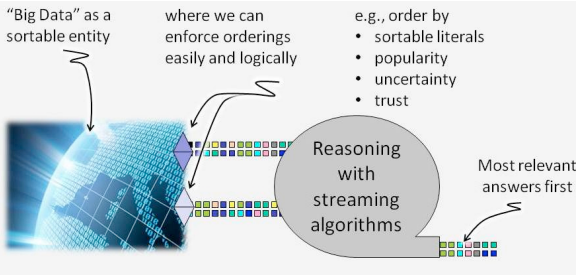

Fig. 2. A bird's-eye view on order-aware reasoning

can be obtained by means of simple operations, such as column sorts, that can be supported efficiently. For instance, indexes for sorted access can be created for any database column of an ordinal type in relational databases or for any geometry column in geo-spatial databases. Expensive orderings, in contrast, can only be produced as the result of complicated operations, such as joining intermediate results (e.g., the reciprocal distance of two flying objects in a given moment in time) or invoking complex custom functions (e.g., the impact of an opinion maker in the last hour as a function of the number of replies and re-tweets). Many forms of cheap or natural sorted data access can be leveraged by algorithms to obtain a core selection criterion, thus reducing the impact of evaluating expensive orders.

Data of the given size, frequently changing and intrinsically ordered, calls for streaming algorithms. These algorithms completely avoid random access to data, i.e., require only one pass or a small number of passes over the data, while using a workspace that is much smaller than the size of the data. Examples include many algorithms that perform computations by splitting a problem into the two problems of sorting and solving. Typical streaming algorithms feature low space complexity upper bounds that are polylog in the size of the input (i.e., their memory use is estimated by $O(log^k(n))$, where n is the size of the input and k is a constant). Moreover, although exact time bounds are often not known for streaming algorithms, the majority of these algorithms are also very fast in practice.

Let us consider the running example of Section 2.4 to show how streaming algorithms can make use of the additional complexity of the orderings in the data to effectively speed up inferencing and overcome the scalability bottleneck. As topic detection can be computationally difficult, this becomes expensive for datasets of the size considered in intelligent surveillance. However, we are looking for the most recent contributions

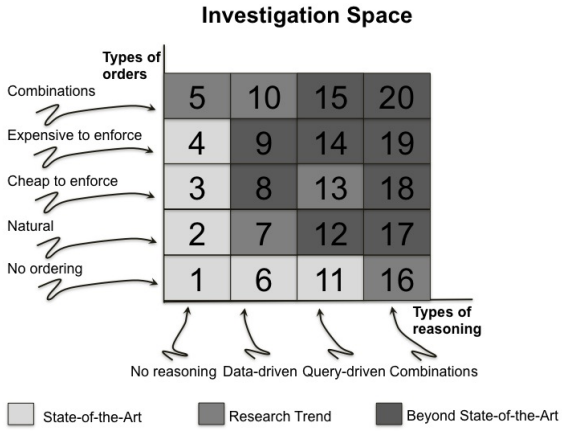

Fig. 3. Investigation space for order-aware reasoning.

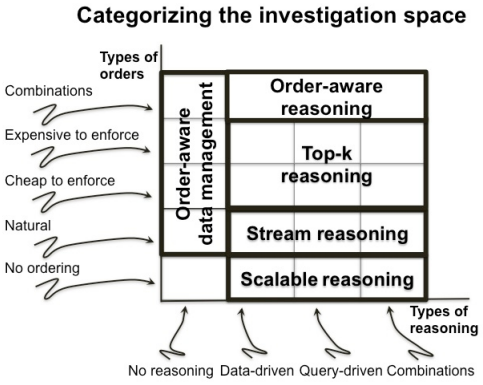

Fig. 4. A proposal for naming categories of areas in the investigation space for order-aware reasoning.

at physical proximity which are expressed by trusted users; thus inference becomes easier, as we can now filter out tweets from distant or untrusted contributors, and those that have been published longer time ago. Assuming that such ordering can be effectively harnessed, the inference space gets smaller, as we can iteratively evaluate all users by distance and popularity for the topic they posted on, and stop once we have enough good answers. We can stream answers by computing the *top-k* answers according to some metrics, or else we can stream answers as they are discovered, without ordering them, in an *any-k* approach.

The basic approach used in this paper is to consider how streaming algorithms apply to different types of orderings – natural or enforced – and to different types of reasonings, such as data- or query-driven inferencing. The investigation space covers two dimensions, as illustrated in Fig. 3.

The *vertical dimension* is structured by types of orders of increasing complexity. The baseline is a set of scalable data management solutions that do not consider order as first class citizen (i.e., the large majority of those available on the market). On top of that, three different types of orderings can be considered:

- Natural orders
- Cheap orders
- Expensive orders
- Combinations of the three

The *horizontal dimension* is defined by type of reasoning. Once again, the baseline is a set of scalable data management solutions that do not offer reasoning (i.e., the large majority of those available on the market). On top of that, three different types of reasoning methods can be considered:

- Data-driven: those that reduce query latency by materializing inferences at loading time
- Query-driven: those that rewrite the ontological query into one or several simpler queries
- Combined: those that explore mixed approaches

Light grey squares indicate problems for which well-established methods exist; grey squares indicate problems that are currently hot topics in research; dark grey squares denote progress areas in which most innovative techniques can lead to major advances, well beyond the state of the art.

A third dimension, which is not shown in Fig. 3, concerns additional features such as parallelisation and approximation. The former is fundamental for scalability. The latter trades completeness or correctness (or both) for improving performance, and as such is closely related to any-time reasoning, where imperfect results are returned as early as possible and continuously improved if more time is available.

As our framework adds orderings as an extra dimension of inference, new appropriate quality metrics are required. Classical metrics such as soundness, completeness or computational complexity do not take the special requirements of order-based processing into account. For example, with regards to a query answering task, answers could be sound and complete as usual, but now correctness of ordering has to be established. New qualitative and quantitative measures of appropriateness are needed to characterise the quality of order-based methods.

4. Investigation space

In the previous section, we introduced the investigation space of order-aware reasoning with respect to the two dimensions of types of orders and of reasoning. Next, we use this classification to group the areas of investigation into six categories illustrated in Figure 4: data management solutions available on the market (Area 1), order-aware data management (Areas 2 to 5), scalable reasoning for ontology-based information integration (Areas 6, 11 and 16), stream reasoning as defined in [20] (Areas 7, 12 and 17), top-k reasoning (Areas 8, 9, 13, 14, 18 and 19), and full-fledged order-aware reasoning (Areas 10, 15 and 20). In each category, we draw the line between the established state of the art, current research trends, and open challenges for future investigations. We leave the discussion of approximation and parallelisation for Section 5.

4.1. Solutions currently on the market

The large majority of scalable data management solutions available on the market fall into the area of methods and tools that are (almost) completely ignoring orderings in the data, and that do not perform any inference. Existing approaches are based on parallel programming models such as BSP (Bulk Synchronous Parallel), PRAM (Parallel Random Access Machine), PGAS (Partitioned Global Access Space), or Map-Reduce. The latter has been implemented in several frameworks: MapReduce [19], Hadoop [30], SkyNet [58], Disco [21] are examples of data-centric workflow systems (based on the Map-Reduce paradigm) that ease the parallel execution of data-intensive processes on a large cluster of commodity machines. Other frameworks based on the MapReduce paradigm such as Hive [61] or Pig [47] allow the specification of ordering constraints for user queries over massive data collections, but no specific optimisation is provided for top-k or any-k queries.

4.2. Order-aware data management

Area 2 is the area of Stream Data Management Systems (DSMS) [24] and Complex Event Processors (CEP) [43], which is about naturally ordered data without reasoning. This type of system has been largely investigated in the end of the 1990s and in the beginning of the 2000s. A number of start-ups (e.g., Mike Stonebraker's StreamBase) were founded and major data management solution vendors have extended their offer in this direction (e.g., Microsoft's StreamInsight, IBM's InfoSphere Streams).

> *The simplest portion of the query in our running example, i.e., the request to retrieve the microposts that have been posted recently, is the typical query a DSMS is optimised for.*

DSMS and CEP share two ideas: a) processing "on the fly" on data streams while they pass by, and b) exploiting the temporal order of the data stream to optimise the computation. These two ideas are the basis for a well-known class of algorithms: the streaming algorithms, that we already discussed in Section 3. In 1998, "Computing on Data Streams" [31] was the first publication formalising streaming algorithms, but early works on this class of algorithms have their roots in the late 1970s, when, for instance, the first query optimisation technique based on estimation of order statistics in the data was presented [54].

For instance, streaming algorithms that compute graph statistics, matchings in a graph, and random walks [65] can cope with massive graphs that can only be stored in high capacity storage devices where random access is extremely slow (as compared to primary memory devices). Notably, to use streaming algorithms, data does not need to be naturally ordered (e.g., by recency as in DSMS/CEP); it has to be sortable by some criteria, e.g., popularity or physical distance, required by the streaming algorithm that will read it.

Area 3 is about data on which orderings are cheap to enforce, again without reasoning. It is the area of top-k query answering, which has been the subject of research since the 1990s. One of the earliest works in the area is the famous rank aggregation algorithm by Fagin [23], which allows merging multiple lists of results returned from different databases with one pass on both lists. More recent works (e.g., [34]) focused on rank-aware join algorithms (see [35] for a survey), i.e., return the top-k results of a join of a set of ordered relationships scanning only a minimal part of each relation and avoiding random access.[2] Area 4 is about efficient evaluation of top-k queries that include expensive to enforce orderings. Some important theoretical results and efficient algorithms (e.g., on the minimal number of probes absolutely required to return correct results [17]) are available in this area, but no existing work tackles the problem of optimal planning of top-k

[2] If random access is possible, some rank-join algorithms perform a minimum amount of random accesses after the sequential scans.

queries considering both predicate correlation and selectivity estimation.

Regarding the running example, solutions studied in Areas 3 and 4 allow to retrieve nearby shops that are discussed by popular social media users.

Area 5 is about combining data stream processing with top-k query answering. This is where most of the current research efforts concentrate. "Continuous monitoring of top-k queries over sliding windows" [45] is the leading work in this area. It shows how to efficiently combine sliding windows, which harvest the natural orders of a data streams, with top-k query answering.

Using the methods of Area 5, it is possible to retrieve the shops nearby that popular social media users are currently positively posting about.

However, two parts of the query in the running example remain difficult to express: knowing which topics are related to fashion and computing which recent discussions on social media are popular. Both are difficult to model without an expressive ontological language (such as OWL 2) and both require complex algorithms that an ontology reasoner can handle natively. Moreover, these techniques do not cope with heterogeneity, i.e., data should be translated in one common representation before order-aware data management techniques can be applied.

4.3. Scalable reasoning for ontology-based information integration

Areas 6, 11, and 16 consider ontological background information as a basis for inferring implicit information from the given data. In our target applications, this is particularly useful for ontology-based information integration, i.e., for handling heterogeneity in the input data [39].

In the running example, ontological background knowledge can be used to model relationships between more specific and more general topics of interest, which can be used to infer which concrete topics are related to fashion.

Area 6 covers reasoning methods that draw ontological inferences based on the available data (possibly including ontological information). A prime example of this "bottom-up" approach is materialisation in relational databases [1,29], which is closely related to forward chaining in logic programming. This approach has successfully been applied to ontologies, in particular in the lightweight ontology language OWL RL and fragments thereof. Commercial implementations of this idea include OWLIM, Virtuoso, AllegroGraph, and OntoBroker. Similar methods have been applied with great success to more expressive fragments of OWL under the label consequence-based reasoning [36,57] as implemented in the ELK reasoner for OWL EL [37]. The common advantage of bottom-up techniques is that the computation is driven by the inferences that are possible based on the data. This guides the search for logical consequences and reduces overall computational effort. The major disadvantage of data-driven approaches, however, is that they do not take our actual information-need (query) into account, i.e., they are usually not goal-directed.

Area 11 includes approaches that search for logical consequences that lead to the answer of a particular query. Typical examples from relational databases are the numerous techniques for query rewriting [1], which closely relate to backward chaining in logic programming. In the context of ontologies, this approach was mainly applied to OWL QL and related logics [16,50,27]. Implementations include QuOnto, Owlgres, and Requiem. Query rewriting has also been used to implement reasoning capabilities in commercial RDF databases, e.g., in Virtuoso (configurable alternative to materialisation) and 4Store (4sr plugin). The theoretical foundations of query rewriting have also been studied for more expressive ontology languages that are based on existential rules (a.k.a. Datalog$\pm$) [27,4]. The advantage of these methods is that they limit the search space by considering the actual information-need (query) instead of computing all possible inferences. On the other hand, rewriting a query in a way that is not depending on the given data may require a very high number of rewritings (exponentially many for OWL QL, possibly infinitely many for existential rules).

Area 16 therefore aims to combine the advantages of bottom-up and top-down approaches. A classical example is the Magic Sets technique known in deductive databases, which achieves a goal-directed behaviour in bottom-up computations [1]. For more expressive ontology languages, however, this combination is not clear and subject to on-going research. Initial proposals for combined approaches have been made for a limited fragment of OWL EL [44] and, on a purely theoretical level, for existential rules [4]. We note that there are a number of reasoning methods that do not specifically relate to query answering, e.g., tableau meth-

ods, that check satisfiability of a logical theory by constructing models. Such general approaches can be a basis for data- or query-driven approaches, but are not combined approaches in our sense since they do not usually combine the advantages of data- and query-driven procedures.

4.4. Stream reasoning

Investigations in Areas 7, 12 and 17 deal with reasoning on rapidly changing information, namely *stream reasoning* [20]. This new reasoning method removes the common assumption in scalable reasoning that knowledge bases are static or evolving slowly. By harvesting the natural temporal order in data streams, stream reasoning addresses the requirements of a number of modern applications, ranging from sensor networks to social media analysis.

> *In terms of our example, stream reasoning methods are the most appropriate to compute which recent discussions on social media are popular.*

In the last three years, several independent groups elaborated stream reasoning techniques [9,8,22,2,26] applied to sensor networks, healthcare, financial fraud detection and social media analysis. These approaches exploit different stream processing and reasoning techniques, but they share an homogeneous theoretical framework.

All of them, but [26], share the notion of RDF stream [9], which logically models a stream of triples annotated with a non-decreasing timestamp. However, other types of RDF stream can be explored. For instance, each triple could be annotated with two time stamps that describe the time interval in which the triple is valid (this is commonly done in CEP). The granularity of the streamed data element can also be rethought. Choosing a triple as streamed data element is appropriate when a single triple carries enough information. For instance, [7] experimentally proved that effective social media analysis can be based on a stream of triples such as `Alice likes Wonderland`. In semantic sensor networks [56], one observation requires a minimum of ten triples, thus choosing a named graph containing a set of triples as streamed data element can be more appropriate. This is similar to the time-decaying logic programs used in [26]. Punctuation [63] – a mark that identifies substreams allowing to view an infinite stream as a mixture of finite streams – is a flexible alternative approach still unexplored in stream reasoning.

Moreover, little effort has been dedicated so far to the formal definition of stream reasoning inference problems. Preliminary work has been done in [8], which formally defines continuous query answering under RDFS++ entailment regime, and in [26], which gives the formal semantics of stream reasoning with answer set programming in terms of a dedicated module theory [46]. However, a general formal definition of soundness and completeness for continuous query answering under expressive OWL2 entailment regime remains an open problem. Also the notion of inconsistency in stream reasoning deserves further investigation; while in a static domain an individual cannot belong to two disjoint classes (i.e., `Alice` can either belong to `Tall` or to `Short`), when we consider a time frame (i.e., a window in DSMS terms) two inconsistent facts can be present, but the content of the window should not be considered inconsistent, only the most recent statement should be considered. However, this is not as simple as it may appear at a first look and theoretically framing this problem, so that it can be efficiently treated in practice, is an open issue (i.e., well-known AI techniques, such as belief revision [18], provide a sound theoretical framework, but, in practice, naive implementations do not scale).

Algorithmically, [8,22,2] are all data-driven approaches (Area 7). Area 12, which is about integrating natural orders into query-driven reasoning, has not been explored, yet. It may deserve exploration given that in stream reasoning queries are registered, thus the cost of rewriting can be paid only once (at query registration time) and inter-query optimization (e.g., sharing of sub-plans) may show to be able to handle rewritten queries that would be practically too complex for a DBMS. The biggest challenge for stream reasoning is in applying combined data- and query-driven inferencing techniques to data naturally ordered (Area 17).

4.5. Top-k reasoning

Investigations in Areas 3 and 4 are typically referred to as top-k data processing. Likewise, we refer to the research space that deals with reasoning in presence of both cheap to enforce (Areas 8, 13, and 18) and expensive to enforce (Areas 9, 14, and 19) orders as *top-k reasoning*. In traditional reasoning, ranking of results is normally considered a task that increase the *hopelessness* of scaling inference to massive data set; our proposal, instead, is to overcome such a common practice and interleave order and reasoning.

In terms of the running example, top-k reasoning methods are the most appropriate to compute which are the top-k social media users, who are well-known to lead discussions on fashion-related topics and are closest to the requester current location.

Some investigations have been conducted in Areas 13 and 14, where several works addressed the problem of top-k query answering in presence of orders using query rewriting: [60] studies SoftFacts – an ontology-mediated top-k information retrieval system over relational databases; [13] adds order to SPARQL as a first class citizen; the authors of [41] take a different angle by extending SPARQL to querying RDFS annotated by bounded lattice (and thus comes with a partial ordering).

The theoretical framework for top-k reasoning is unexplored. Progressing in this direction calls for a sound identification of the type of data to be managed, and the classes of queries that can be answered with the given data. [13] made a first step in this direction by defining the notion of ranked sets of mappings, and an order-aware SPARQL algebra that embodies rank-aware algebraic operators. Once these basic building blocks of query answering under simple RDF entailment regime is in place, appropriate inference problems can be defined: e.g., notions of *exact top-k closure of an ontology w.r.t. a query and a scoring function* (for an attempt see [53]).

From an algorithmic point of view, the only explored areas in top-k reasoning are Area 13 and 14. Area 8 – methods for top-k materialisation on easy to enforce orders – seems the easiest to explore since order-aware joining algorithms [35] could be applied to rule-base reasoning; Area 9 could benefit from existing works in databases [50]. Perhaps, the biggest challenge in top-k reasoning is in interleaving combined data- and query-driven techniques with techniques that harvest easy (Area 18) and expensive to enforce orders (Area 19).

4.6. Order-aware reasoning

With *full-fledged order-aware reasoning*, we refer to Areas 10, 15, and 20, where data- and query-driven inference methods have to deal with combinations of natural, cheap to enforce and expensive to enforce type of orders. In such a context, the naive assumption of independence of orderings would have to be relaxed, thus theories and methods, which exploit mutual relationships between the three type of orders, have to be rethought.

Considering our running example, methods implementing order-aware reasoning are the only ones able to answer to the query we posed in Section 2.4, i.e., Which users of social media, currently leading popular discussions on fashion-related topics, are closest to my current location? What are they saying about the shopping district nearby?

Some promising work is undergoing in the Answer Set Programming (ASP) community. In Area 10, [25] proposes an streaming algorithm for ASP that first ranks the constants referring to domain elements and, then, fetch them increasing the domain sizes until an answer set is found.

The challenge is to define a theoretical framework that unifies and generalises those defined for stream reasoning and top-k reasoning. Such a framework should pave the way for designing scalable data- and query-driven methods that allows for efficient answering of queries that involve all types of ordering (Areas 10 and 15). As for stream and top-k reasoning, perhaps, the biggest challenge is in interleaving combined data- and query-driven techniques with techniques that harvest all types of orders (Area 20).

5. Approximation and parallelisation

While the solution space in Fig. 3 is based on the input and output behaviour of order-aware systems, there are various additional areas of investigation that are relevant in almost every scenario. In this section, we focus on two such orthogonal research dimensions that are of particular importance to order-aware reasoning: approximation and parallelisation.

5.1. Approximate reasoning

Classical approximation tries to reduce the use of computation time and working memory. Item 1 above considers another critical resource, namely the amount of input data that is required for giving useful answers.

In every case, it is desirable that algorithms compute increasingly accurate results when given more resources (time, memory, data). *Any-time algorithms* are approximation procedures that provide preliminary results of increasing quality at (almost) any stage of the computation. Ideally, the correct result should be ap-

proximated arbitrarily close in this process, but such *asymptotic* behaviour might not always be possible.

By approximate reasoning we mean any approach to reasoning that can produce results that are incomplete (missing answers) or unsound (giving wrong answers), but in a deliberate and controlled way that is accepted as a means of improving performance.[3] Such algorithms might be used to produce quick preliminary results, to give answers when no other method is feasible, or to estimate outputs in situations where only limited accuracy is needed.

Approximation plays a key role in order-aware reasoning, due to the following independent reasons:

1. Restricted access to input data: order-aware reasoning often deals with data that is only partially accessible (in an ordered fashion), making it impossible to guarantee full accuracy in all cases.
2. Soft constraints on output data: order-aware reasoning introduces the order of results as a new output dimension, for which controlled inaccuracy is often more acceptable.
3. Order-aware approximation: reasoning involves computational tasks that are inherently hard to solve, motivating the use of approximation in classical cases; but order provides a new guideline and quality measure for approximation.

Approximation has been considered in some of the investigation areas in Fig. 3. For a classical example that belongs to Area 1, consider the problem of counting the number of triangles in a graph. This is a well-known problem in graph analysis, which is a basic building block for evaluating many practically important graphs, e.g., in social networks, chemical compounds, or networks of Web links. It has been shown that approximate, streaming algorithms can outperform classical, data-bound approaches to this problem by several orders of magnitude [6,14]. Moreover, such approximations can be asymptotic, so that arbitrary accuracy can be achieved [6].

Various forms of approximation have been proposed in areas related to ontology-based data access. Many rule-based systems compute only part of the entailed consequences by employing a set of rules that cannot derive all results. This is the case, e.g., for Jena, Sesame, OWLIM, and Virtuoso, all of which support specific fragments of OWL RL but not the whole language. Incompleteness is mostly well-understood in such cases.

A variety of other approximation methods have been considered in reasoning [51,28]. A typical approach is to approximate the input information by restricting to a simpler ontology language that is then processed with a more efficient, sound and complete algorithm, see, e.g., [48,62]. This is related to the idea of knowledge compilation [55]. Approximate reasoning is also used as a sub-method in many sound and complete reasoners, e.g., the OWL reasoner HermiT first computes the syntactically told class hierarchy before using more complex algorithms for a complete subsumption check. Reasoning approaches that are based on backward-chaining and query rewriting typically suggest any-time approximation, as more and more answers can be found by continuing the (possibly infinite) generation of queries. This is typical for many Prolog implementations and can also be combined with rewriting-based approaches for DL-Lite [16] or existential rules [5].

None of the above systems, however, deal with or take advantage of orderings of any kind. A number of interesting research challenges thus remain open.

5.2. Parallelisation approaches

Parallelisation of reasoning means that the task is split into subproblems that can be solved independently with limited exchange of information. This enables concurrent computation (e.g., by sharing computation among several CPUs of one machine) and distribution (sharing computation among multiple networked machines with independent memory). Both aspects are essential for exploiting state-of-the-art computing systems to go beyond the limits of traditional reasoning.

However, order-based reasoning introduces new challenges to parallelisation, e.g., the following:

1. Ordered data access might be inherently sequential, and thus is harder to distribute.
2. Outputs must be integrated into an overall order, and thus cannot be generated in a fully decentralized fashion.

To bring order-aware reasoning to its full potential, these challenges must be addressed.

In recent years, parallelisation has been successfully employed in various reasoning tasks, especially in rule-based (bottom-up) materialisation approaches. The two main strands of work are multi-machine dis-

[3] Note that this is different from (accurate) reasoning with approximate (imprecise) information, as done, e.g., in fuzzy logic.

tribution and multi-processor concurrency. Distribution is motivated by processing data volumes that are too large for a single machine's working memory. Approaches in that field often target OWL RL [33,65] or a fragment thereof [32,64,66,38], using MapReduce as the main computation paradigm [19]. Another distribution paradigm suggested for this case is the pre-partitioning of inputs [59]. In contrast to these approaches, concurrent processing on a single machine aims at speeding up reasoning for improving reactivity of user-driven applications. The main example of this approach is the ELK reasoner for OWL EL [37] and various preliminary works on parallel reasoning in more expressive ontology languages [40,52,3]. Overall, many of the implemented systems demonstrated performance gains of one or several orders of magnitude, yet none of these systems is applicable to a streaming scenario.

6. Conclusions

Our systematic exploration of the investigation space has shown that a huge amount of foundational and applied research is necessary. Starting from lessons learned from data management, we need:

- A *theory of semantic processing* of massive sets of complex and highly dynamic data. This will include the development of knowledge representation languages, performance metrics and a systematic roadmap about how to process massive, dynamic, ordered data.
- *Methods and techniques* related to such a theoretical framework. This means at least one method for each area of investigation identified in Fig. 3.
- *Implementations* of the above methods according to current software development standards.
- *Rigorous evaluations* of the proposed methods and technology in a comparative way using a purpose-built testing infrastructure.

Putting order as first class citizen in reasoning may start with incremental steps, but a tight integration of ordering and reasoning will require substantial rethinking to the cornerstones on which current semantic technologies are built.

Acknowledgements This research has been supported by the ERC Search Computing (SeCo) project, the Dutch national program COMMIT, the EU FP7 project SEALS, and by the EPSRC projects ConDOR, ExODA and LogMap. Many ideas of this paper stem from results of the LarKC project. We also want to thank Frank van Harmelen for his important contribution, and Tony Lee (Saltlux), Andreas Schreiber (DLR) and Achim Basermann (DLR) for the valuable discussion on concrete examples of problems that require order-aware reasoning.

References

[1] Abiteboul, S., Hull, R., & Vianu, V. (1995). *Foundations of Databases*. Addison-Wesley.

[2] Anicic, D., Fodor, P., Rudolph, S., & Stojanovic, N. (2011). EP-SPARQL: a unified language for event processing and stream reasoning. In S. Srinivasan, K. Ramamritham, A. Kumar, M. P. Ravindra, E. Bertino, and R. Kumar, editors, *WWW*, pages 635–644. ACM.

[3] Aslani, M. & Haarslev, V. (2010). Parallel TBox classification in description logics – first experimental results. In H. Coelho, R. Studer, and M. Wooldridge, editors, *Proc. 19th European Conf. on Artificial Intelligence (ECAI'10)*, volume 215 of *Frontiers in Artificial Intelligence and Applications*, pages 485–490. IOS Press.

[4] Baget, J.-F., Leclère, M., Mugnier, M.-L., & Salvat, E. (2009). Extending decidable cases for rules with existential variables. In [12], pages 677–682.

[5] Baget, J.-F., Leclère, M., Mugnier, M.-L., & Salvat, E. (2011). On rules with existential variables: Walking the decidability line. *Artif. Intell.*, **175**(9-10), 1620–1654.

[6] Bar-Yossef, Z., Kumar, R., & Sivakumar, D. (2002). Reductions in streaming algorithms, with an application to counting triangles in graphs. In D. Eppstein, editor, *SODA*, pages 623–632. ACM/SIAM.

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

[8] Barbieri, D. F., Braga, D., Ceri, S., Della Valle, E., & Grossniklaus, M. (2010b). Incremental reasoning on streams and rich background knowledge. In [42], pages 1–15.

[9] Barbieri, D. F., Braga, D., Ceri, S., Della Valle, E., & Grossniklaus, M. (2010c). Querying RDF streams with C-SPARQL. *SIGMOD Record*, **39**(1), 20–26.

[10] Ben-Ari, M. (1990). *Principles of concurrent and distributed programming*. PHI Series in computer science. Prentice Hall.

[11] Bernstein, A., Karger, D. R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., & Thirunarayan, K., editors (2009). *Proc. 8th Int. Semantic Web Conf. (ISWC'09)*, volume 5823 of *LNCS*. Springer.

[12] Boutilier, C., editor (2009). *IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA, July 11-17, 2009*.

[13] Bozzon, A., Della Valle, E., & Magliacane, S. (2011). Towards and efficient SPARQL top-k query execution in virtual RDF stores. In S. Chakrabarti and D. Martinenghi, editors, *Proc. 5th International Workshop on Ranking in Databases (DBRANK 2011)*, pages 1–6.

[14] Buriol, L. S., Frahling, G., Leonardi, S., Marchetti-Spaccamela, A., & Sohler, C. (2006). Counting triangles in data streams. In S. Vansummeren, editor, *PODS*, pages 253–262. ACM.

[15] Calvanese, D. & Lausen, G., editors (2008). *Web Reasoning and Rule Systems, Second International Conference, RR 2008, Karlsruhe, germany, October 31-November 1, 2008. Proceedings.*, volume 5341 of *LNCS*. Springer.

[16] Calvanese, D., Giacomo, G. D., Lembo, D., Lenzerini, M., & Rosati, R. (2007). Tractable reasoning and efficient query answering in description logics: The DL-Lite family. *J. Autom. Reasoning*, **39**(3), 385–429.

[17] Chang, K. C.-C. & won Hwang, S. (2002). Minimal probing: supporting expensive predicates for top-k queries. In M. J. Franklin, B. Moon, and A. Ailamaki, editors, *SIGMOD Conference*, pages 346–357. ACM.

[18] Darwiche, A. & Pearl, J. (1996). On the logic of iterated belief revision. *Artificial intelligence*, **89**, 1–29.

[19] Dean, J. & Ghemawat, S. (2008). Mapreduce: simplified data processing on large clusters. *Commun. ACM*, **51**, 107–113.

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

[21] Disco (2012). Disco project - http://discoproject.org.

[22] Do, T., Loke, S., & Liu, F. (2011). Answer set programming for stream reasoning. In C. Butz and P. Lingras, editors, *Advances in Artificial Intelligence*, volume 6657 of *LNCS*, pages 104–109. Springer Berlin / Heidelberg.

[23] Fagin, R. (1996). Combining fuzzy information from multiple systems (extended abstract). In *Proceedings of the fifteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, PODS '96, pages 216–226, New York, NY, USA. ACM.

[24] Garofalakis, M., Gehrke, J., & Rastogi, R. (2007). *Data Stream Management: Processing High-Speed Data Streams*. Springer-Verlag New York, Inc.

[25] Gebser, M., Sabuncu, O., & Schaub, T. (2011). An incremental answer set programming based system for finite model computation. *AI Commun.*, **24**(2), 195–212.

[26] Gebser, M., Grote, T., Kaminski, R., Obermeier, P., Sabuncu, O., & Schaub, T. (2012). Stream Reasoning with Answer Set Programming: Preliminary Report. In T. Eiter and S. McIlraith, editors, *Proceedings of the Thirteenth International Conference on Principles of Knowledge Representation and Reasoning (KR'12)*, pages 613–617. AAAI Press.

[27] Gottlob, G., Orsi, G., & Pieris, A. (2011). Ontological queries: Rewriting and optimization. In S. Abiteboul, K. Böhm, C. Koch, and K.-L. Tan, editors, *ICDE*, pages 2–13. IEEE Computer Society.

[28] Groot, P., Stuckenschmidt, H., & Wache, H. (2005). Approximating Description Logic Classification for Semantic Web Reasoning. In A. Gómez-Pérez and J. Euzenat, editors, *Proc. 2nd European Semantic WebConf. (ESWC'05)*, volume 3532 of *LNCS*, pages 318–332. Springer.

[29] Gupta, A. & Mumick, I. S., editors (1999). *Materialized views: techniques, implementations, and applications*. MIT Press, Cambridge, MA, USA.

[30] Hadoop (2012). Apache hadoop framework - http://hadoop.apache.org/.

[31] Henzinger, M. R., Raghavan, P., & Rajagopalan, S. (1999). Computing on data streams. In *External Memory Algorithms: Dimacs Workshop External Memory and Visualization, May 20-22, 1998*, volume 50, page 107. Amer Mathematical Society.

[32] Hogan, A., Harth, A., & Polleres, A. (2009). Scalable authoritative OWL reasoning for the Web. *Int. J. of Semantic Web Inf. Syst.*, **5**(2), 49–90.

[33] Hogan, A., Pan, J. Z., Polleres, A., & Decker, S. (2010). SAOR: template rule optimisations for distributed reasoning over 1 billion linked data triples. In [49], pages 337–353.

[34] Ilyas, I. F., Aref, W. G., & Elmagarmid, A. K. (2003). Supporting top-k join queries in relational databases. In *VLDB*, pages 754–765.

[35] Ilyas, I. F., Beskales, G., & Soliman, M. A. (2008). A survey of top-k query processing techniques in relational database systems. *ACM Comput. Surv.*, **40**(4), 1–58.

[36] Kazakov, Y. (2009). Consequence-driven reasoning for Horn $\mathcal{SHIQ}$ ontologies. In [12], pages 2040–2045.

[37] Kazakov, Y., Krötzsch, M., & Simancík, F. (2011). Concurrent classification of EL ontologies. In *Proceedings of the 10th international conference on The semantic web - Volume Part I*, ISWC'11, pages 305–320, Berlin, Heidelberg. Springer-Verlag.

[38] Kotoulas, S., Oren, E., & van Harmelen, F. (2010). Mind the data skew: distributed inferencing by speeddating in elastic regions. In *Proc. 19th Int. Conf. on World Wide Web (WWW'10)*, WWW'10, pages 531–540. ACM.

[39] Lenzerini, M. (2002). Data integration: A theoretical perspective. In L. Popa, editor, *PODS*, pages 233–246. ACM.

[40] Liebig, T. & Müller, F. (2007). Parallelizing tableaux-based description logic reasoning. In R. Meersman, Z. Tari, and P. Herrero, editors, *Proceedings of OTM Workshops 2007, Part II*, volume 4806 of *LNCS*, pages 1135–1144. Springer.

[41] Lopes, N., Polleres, A., Straccia, U., & Zimmermann, A. (2010). AnQL: SPARQLing up annotated RDFS. In [49], pages 518–533.

[42] Lora Aroyo et al., editor (2010). *The Semantic Web: Research and Applications, 7th Extended Semantic Web Conference, ESWC 2010, Heraklion, Crete, Greece, May 30 - June 3, 2010, Proceedings, Part I*, volume 6088 of *LNCS*. Springer.

[43] Luckham, D. (2008). The power of events: An introduction to complex event processing in distributed enterprise systems. In N. Bassiliades, G. Governatori, and A. Paschke, editors, *Rule Representation, Interchange and Reasoning on the Web*, volume 5321 of *Lecture Notes in Computer Science*, pages 3–3. Springer Berlin / Heidelberg.

[44] Lutz, C., Toman, D., & Wolter, F. (2009). Conjunctive query answering in the description logic EL using a relational database system. In [12], pages 2070–2075.

[45] Mouratidis, K., Bakiras, S., & Papadias, D. (2006). Continuous monitoring of top-k queries over sliding windows. In S. Chaudhuri, V. Hristidis, and N. Polyzotis, editors, *SIGMOD Conference*, pages 635–646. ACM.

[46] Oikarinen, E. & Janhunen, T. (2006). Modular equivalence for normal logic programs. In G. Brewka, S. Coradeschi, A. Perini, and P. Traverso, editors, *ECAI*, volume 141 of *Frontiers in Artificial Intelligence and Applications*, pages 412–416. IOS Press.

[47] Olston, C., Reed, B., Srivastava, U., Kumar, R., & Tomkins, A. (2008). Pig latin: a not-so-foreign language for data processing. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, SIGMOD '08, pages 1099–1110, New York, NY, USA. ACM.

[48] Pan, J. Z. & Thomas, E. (2007). Approximating OWL-DL ontologies. In *Proc. 22nd AAAI Conf. on Artificial Intelligence*

(AAAI'07), pages 1434–1439. AAAI Press.

[49] Patel-Schneider, P. F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J. Z., Horrocks, I., & Glimm, B., editors (2010). *The Semantic Web - ISWC 2010 - 9th International Semantic Web Conference, ISWC 2010, Shanghai, China, November 7-11, 2010, Revised Selected Papers, Part I*, volume 6496 of *LNCS*. Springer.

[50] Pérez-Urbina, H., Horrocks, I., & Motik, B. (2009). Efficient query answering for OWL 2. In A. Bernstein, D. R. Karger, T. Heath, L. Feigenbaum, D. Maynard, E. Motta, and K. Thirunarayan, editors, *International Semantic Web Conference*, volume 5823 of *LNCS*, pages 489–504. Springer.

[51] Rudolph, S., Tserendorj, T., & Hitzler, P. (2008). What is approximate reasoning? In [15], pages 150–164.

[52] Schlicht, A. & Stuckenschmidt, H. (2009). Distributed resolution for expressive ontology networks. In A. Polleres and T. Swift, editors, *Proc. 3rd Int. Conf. on Web Reasoning and Rule Systems (RR 2009)*, volume 5837 of *LNCS*, pages 87–101. Springer.

[53] Schlobach, S. (2011). Top-k reasoning for the semantic web. *Proceedings of the 11th Interational Semantic Web Conference ISWC2011*, pages 55–59.

[54] Selinger, P. G., Astrahan, M. M., Chamberlin, D. D., Lorie, R. A., & Price, T. G. (1979). Access path selection in a relational database management system. In P. A. Bernstein, editor, *SIGMOD Conference*, pages 23–34. ACM.

[55] Selman, B. & Kautz, H. A. (1996). Knowledge compilation and theory approximation. *J. ACM*, **43**(2), 193–224.

[56] Sheth, A. P., Henson, C. A., & Sahoo, S. S. (2008). Semantic Sensor Web. *IEEE Internet Computing*, **12**(4), 78–83.

[57] Simancik, F., Kazakov, Y., & Horrocks, I. (2011). Consequence-based reasoning beyond horn ontologies. In T. Walsh, editor, *IJCAI*, pages 1093–1098. IJCAI/AAAI.

[58] Skynet (2012). Skynet ruby project - http://skynet.rubyforge.org/.

[59] Soma, R. & Prasanna, V. K. (2008). Parallel inferencing for OWL knowledge bases. In *Proc. Int. Conf. on Parallel Processing (ICPP'08)*, pages 75–82. IEEE Computer Society.

[60] Straccia, U. (2010). Softfacts: A top-k retrieval engine for ontology mediated access to relational databases. In *SMC*, pages 4115–4122. IEEE.

[61] Thusoo, A., Sarma, J. S., Jain, N., Shao, Z., Chakka, P., Anthony, S., Liu, H., Wyckoff, P., & Murthy, R. (2009). Hive: a warehousing solution over a map-reduce framework. *Proc. VLDB Endow.*, **2**, 1626–1629.

[62] Tserendorj, T., Rudolph, S., Krötzsch, M., & Hitzler, P. (2008). Approximate OWL-reasoning with screech. In [15], pages 165–180.

[63] Tucker, P. A., Maier, D., Sheard, T., & Fegaras, L. (2003). Exploiting punctuation semantics in continuous data streams. *IEEE Trans. Knowl. Data Eng.*, **15**(3), 555–568.

[64] Urbani, J., Kotoulas, S., Oren, E., & van Harmelen, F. (2009). Scalable distributed reasoning using MapReduce. In [11], pages 634–649.

[65] Urbani, J., Kotoulas, S., Maassen, J., van Harmelen, F., & Bal, H. E. (2010). OWL reasoning with WebPIE: calculating the closure of 100 billion triples. In [42], pages 213–227.

[66] Weaver, J. & Hendler, J. A. (2009). Parallel materialization of the finite RDFS closure for hundreds of millions of triples. In [11], pages 682–697.