1

Introduction

More and more applications require real-time processing of data streams. In oil operations, given an alarm on a well in progress to drown, how long do I have considering the historical behaviour of that well? Or, given this brand of turbine, what is the expected time to failure when the barring gear starts to vibrate as detected in the last 10 minutes? In weather monitoring, which sensors are observing a blizzard within a 20 mile radius of a densely populated area? In Smart Cities, is public transportation where the people are? Is a traffic jam occurring on this highway? How can we reroute travellers? In customer relationship management, who are the best available agents to route all these unexpected contacts from business users about the new tariff plan launched yesterday? In Social Media Analytics, who is driving the discussion about the top 10 emerging topics across all the social networks?

A system able to answer those queries must:

1. handle massive datasets – a typical oil production platform is equipped with about 400,000 sensors; Facebook, as of March 31, 2014, has 1.28 billion of monthly active users [Facebook, 2014], etc.

2. process data streams on the fly – A contact centre for a global telecommunication company may receive thousands of contacts per minutes; an average of 3 million users press the Facebook “I like it” button every minute; etc.

3. cope with heterogeneous datasets – a large variety of static and streaming data sources and data management solutions exists in any domain. For instance, Milano has deployed some 600 traffic light systems equipped with inductive loops in
1. INTRODUCTION

the last 10 years: they use five different data formats, have different operational conditions, etc. Similarly, in social media, each network has its own data model and APIs.

4. cope with incomplete data – sensors can have run out of battery or networking links can be broken, in social media part of the conversation may occur outside the social network, or the APIs that are used to access the social stream may have got a limited sampling rate.

5. cope with noisy data – sensors can be faulty or observation can be performed in a situation where the sensor is out of its operational ranges, in social media sentences can be formulated in an ironic way and a sentiment mining solution may be unable to detect it with 100% correctness.

6. provide reactive answers – answers should be generated while meeting operational deadlines from event occurrence to system response. The time available to answer depends on the application domain: in contact centres the routing decision is constrained to be sub-seconds, in oil operations the detection of dangerous situations must occur within minutes.

7. support fine-grained information access – the issued query may require to locate exactly a turbine, a means of public transportation, an agent in a contact centre among thousands of similar ones.

8. integrate complex domain models – social media analytics may require topic models to make sense of a conversation, a control system for oil production may require to model operational and control processes, etc.

Indeed, systems capable of scalable stream processing exist. Specialised Data-Stream Management Systems (DSMS) [Garofalakis et al., 2007] and Complex Event Processing (CEP) [Luckham, 2001] have been well investigated in the last decade. They can provide reactive fine-grained information access even in the presence of noisy data. Similarly, recent research on Semantic Web, an in particular to scalable Ontology Based Data Access (OBDA) [Calvanese et al., 2011], showed that complex domain models can be used to offer fine grained information access to heterogenous and incomplete datasets.
Table 1.1: The requirements Stream Reasoning aims at covering and how DSMS, CEP and OBDA cover them.

<table>
<thead>
<tr>
<th>Requirement</th>
<th>DSMS/CEP</th>
<th>OBDA</th>
<th>Stream Reasoning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Massive data</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Streaming data</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Heterogenous data</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Incomplete data</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Noisy data</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Reactive answers</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Fine-grain information access</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Complex domain models</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

As illustrated in Table 1.1, none of those solutions covers the entire spectrum of requirements. Therefore, the research question that we address in this thesis is:

Q.1 is it possible to make sense in real time of multiple, heterogeneous, gigantic and inevitably noisy and incomplete data streams in order to support the decision process of extremely large numbers of concurrent users?

In [j.1, 2009], Stefano Ceri, Frank van Harmelen, Dieter Fensel and I define *Stream Reasoner* as a system capable to address all those requirements at once and *Stream Reasoning* the approach to address them.

Only a decade ago, proposing a system to answer stream-reasoning questions would have sounded like science fiction, mainly due to the lack of: a) data, b) data management solutions able to cope with data streams, and c) scalable reasoning solutions. Nowadays, the situation has radically changed: data is widely available, DSMS and CEP matured in industrial products for streaming data management [Cugola & Margara, 2012], the Semantic Web standards (RDF, SPARQL and OWL) are widely accepted [Bizer et al., 2009], reasoning has shown to be possible at very large data sizes [Urbani et al., 2012] and OBDA is rapidly gaining momentum.

In the spirit of a PhD thesis as collection of papers, this thesis presents a number papers that envision, elaborate, evaluate and discuss seven years of research in Stream Reasoning. The papers document that a) the Semantic Web stack can be extended
so to incorporate streaming data as a first class citizen, b) the Stream Reasoning task is feasible, c) the very nature of streaming data offers opportunities to optimise reasoning tasks where data are ordered by recency and can be forgotten after a while, d) a combination of deductive and inductive stream reasoning techniques can cope with incomplete and noisy data, and e) there are application domains where Stream Reasoning offers an adequate solution. Moreover, from the lessons learnt in these years stems the research question: can orders (beyond the specific order in time) be harnessed to optimise the reasoning tasks? Answering this question is the main future work pointed out in this thesis.

Before presenting the papers that form this thesis, Section 1.1 refines the scope of Stream Reasoning research by stating some simplifying assumptions and formulating the sub-questions answered in the papers. After this, Section 1.2 explains how the papers presented in this thesis are grouped, briefly describing the content of each paper to ease the understanding and the exploration of this document. Finally, Section 1.3 acknowledges the external support that was given in conducting the research presented in this thesis.

1.1 Scope of research

The Database (DB) and The Knowledge Representation (KR) communities often use different terms to indicate similar concepts. The DB community distinguishes among schema and data, whereas the KR community distinguishes among factual, and terminological knowledge. The notion of data is close to the notion of factual knowledge, and similarly the notion of schema is close to the notion of terminological knowledge. Schema are normative and define what data must conform to. Terminological knowledge can be use to define schema (e.g., modelling the relations Person and hasChild), but it also allows for establishing relationships between elements in the schema (e.g., a Parent is a Person who is related to at least another person through the hasChild relation), thus for formally describing an application domain.

For the purpose of this thesis, knowledge denotes terminological knowledge whereas data is used as a synonymous of factual knowledge. As in OBDA, the knowledge forms the common vocabulary that can be used to describe and access the data.
1.1 Scope of research

Knowledge and data can change over time. For instance, in Smart Cities, names of streets, landmarks, events, etc. change slowly (between years and hours), whereas peaks of hundreds of geo-localised micro-posts per minutes may be observed around a stadium during a big event. In the context of this thesis, knowledge is assumed to be invariable while the system is subject to querying; only data can change. Of course, knowledge is subject to change, but then the mutating part of the terminology must be not object of querying.\footnote{This is normal in databases where change of schemas occur by means of create or alter table command. While, for instance, the alter table command is executed all query processing relative to that table is suspended.}

Traditional databases are suitable for capturing relatively small quantities of knowledge in their schema and huge datasets that do not change or change slowly (e.g., a bank account is updated at most few times per hour). Changing data can be modelled by means of triggers that perform updates; for example a trigger may update the balance of a bank account when a transaction is registered.

Semantic Web techniques are suitable for capturing large and relatively complex knowledge and, in the last decade, have been shown to scale to large datasets [Urbani et al., 2012]. In particular, OBDA approaches allow to logically integrate heterogeneous data sources using an integrated model expressed in an ontological language. When the ontology language is light-weight [Calvanese et al., 2007] (e.g., the QL profile of OWL 2 [Calvanese et al., 2009]) the queries issued using the vocabulary defined in integrated conceptual model can be rewritten in queries that a traditional database can answer efficiently without further processing. When the application domain requires more complex knowledge (e.g., the closure of a transitive property), materialisation techniques (a.k.a., forward reasoning) can be applied to the data in the database, inferred facts can be stored back into the database [Bishop et al., 2011] and the query can be evaluated on the expanded databases. However, reasoners risk being unable to meet the needs for reactive query-answering when the mean time between changes in the facts decreases and the number of facts increases. They lack the notion of continuous queries and continuous reasoning tasks.

This situation has been observed in the database in the late ’90s [Babcock et al., 2002] when DSMS introduced a paradigm changes from persistent relations to transient streams – with the innovative assumption that streams can be consumed on the fly.
1. INTRODUCTION

(rather then stored forever) – and from user-invoked queries to continuous queries – queries which are persistently monitoring streams and are able to produce their answers even in the absence of invocation. DSMSs can support parallel query answering over data originating in real time [Shah et al., 2004] and can cope with bursts of data by adapting their behaviour and gracefully degrading answer accuracy by introducing approximations [Tatbul et al., 2003]. Stream Reasoning advocates the need for a similar paradigmatic change in the Semantic Web community.

In the last seven years, we formulated and answered the following sub-questions that operationalise the overall research question Q.1:

SQ.1 Is it possible to (syntactically and semantically) extend the Semantic Web stack in order to represent heterogenous data streams, continuous queries, and continuous reasoning tasks?
Answer: yes, introducing the RDF stream data model, extending SPARQL with continuous semantics, namely Continuous-SPARQL (see Chapter 3).

SQ.2 Is it possible to optimise continuous querying and continuous reasoning tasks so to provide reactive answers to large number of concurrent users ?
Answer: yes, by exploiting the ordered nature of data streams and the possibility to forget old enough information (see Chapter 4).

SQ.3 Is it possible to cope with the noisy and incomplete nature of data streams?
Answer: yes, by combining RDF streams and Continuous-SPARQL with Machine Learning technologies (see Chapter 5).

The answers and the case studies, where we showed that processing data streams at semantic level is an effective choice, guarantee for the ability of a Stream Reasoner to provide adequate answers to the queries illustrated in the beginning of this chapter and many others.

1.2 Summary of the chapters

This thesis consists of 21 scientific published papers grouped in 7 chapters. This section provides a brief description of each of these chapters explaining which papers are included in each chapter and why. The numbering of the papers follows the schema
1.2 Summary of the chapters

(publication type).(progressive number), where journal (abbreviated as p), conference (c), workshop (w), and other (o) are the admitted types. I will use this numbering schema in Chapter 8 to make it easier to spot the papers that form this thesis among all the referenced ones.

Chapter 2: Vision

In 2008, we perceived Stream Reasoning as an unexplored, yet high impact research area. We felt it was worth to share concrete examples of Stream Reasoning applications, problems to be solved, conceptual architecture of a Stream Reasoner, so to set a research agenda for the community and to create a shared agreement on how to measure progress.

These contents are presented in the following three papers:

http://doi.ieeecomputersociety.org/10.1109/MIS.2009.125

c.1 Emanuele Della Valle, Stefano Ceri, Davide Francesco Barbieri, Daniele Braga, Alessandro Campi: A First Step Towards Stream Reasoning. FIS 2008: 72-81
http://dx.doi.org/10.1007/978-3-642-00985-3_6


Chapter 3: Stream Data Management for the Semantic Web

Sub-question SQ.1 – is it possible to (syntactically and semantically) extend the Semantic Web stack in order to represent heterogenous data streams, continuous queries and continuous reasoning tasks? – is positively answered in the following group of papers:
1. INTRODUCTION

o.1 Davide Francesco Barbieri, Daniele Braga, Stefano Ceri, Emanuele Della Valle, Michael Grossniklaus: C-SPARQL: SPARQL for continuous querying. WWW 2009: 1061-1062
http://doi.acm.org/10.1145/1526709.1526856

http://dx.doi.org/10.1142/S1793351X10000936

http://doi.acm.org/10.1145/1860702.1860705

w.2 Davide Francesco Barbieri, Emanuele Della Valle: A Proposal for Publishing Data Streams as Linked Data - A Position Paper. LDOW 2010

In [o.1, 2009], we propose the notion of an RDF Stream as an unbound list of tuples \(< t, \tau >\) where \(t\) is an RDF triple and \(\tau\) is a non-decreasing timestamp. Triples with the same timestamp are said to be contemporaneous or simultaneous. This extension is at the logical level and it does not impose any specific syntax. In [w.2, 2010], we propose a way to publish RDF streams as Linked Data using two types of RDF named graphs: instantaneous Graphs (iGraphs) to group all the triples with the same timestamp, and streaming Graphs (sGraphs) to represent a portion of an RDF stream as a list of iGraphs.

In [o.1, 2009; j.2, 2010; j.3, 2010], we propose Continuous SPARQL (C-SPARQL) as a way to express SPARQL queries on multiple RDF streams as well as static information stored in RDF graphs. [o.1, 2009] contains the first syntactic sketch of C-SPARQL based on the CQL [Arasu et al., 2006] query language proposed by Jennifer Widom et al. for the DSMS (namely, Stream) investigated at Stanford university in the early 2000s [Arasu et al., 2003a]. In particular, C-SPARQL inherits the notion of registered queries and windows through which RDF stream are observed. [j.2, 2010] formalises
the semantics of C-SPARQL and [j.3, 2010] updates C-SPARQL to SPARQL 1.1 syntax and semantics. Specifically, [j.3, 2010] records the decision for adopting SPARQL 1.1 aggregates instead of those proposed in [o.1, 2009;j.2, 2010].

The representation of continuous reasoning tasks, at least those explored in the papers listed in Chapter 4, does not require to extend the semantics of the ontological languages proposed for the Semantic Web [Calvanese et al., 2009].

Chapter 4: Continuous querying and reasoning optimisation

Sub-question SQ.2 – is it possible to optimise continuous querying and continuous reasoning tasks so to provide reactive answers to large number of concurrent users? – is positively answered in the papers selected for this chapter:

w.3 Davide Francesco Barbieri, Daniele Braga, Stefano Ceri, Emanuele Della Valle, Michael Grossniklaus: Continuous Queries and Real-time Analysis of Social Semantic Data with C-SPARQL. SDoW 2009.

c.2 Davide Francesco Barbieri, Daniele Braga, Stefano Ceri, Emanuele Della Valle, Michael Grossniklaus: Incremental Reasoning on Streams and Rich Background Knowledge. ESWC (1) 2010: 1-15
http://dx.doi.org/10.1007/978-3-642-13486-9_1

o.2 Daniele DellAglio, Emanuele Della Valle: Incremental Reasoning on RDF streams. Linked Data Management (Andreas Harth, Katja Hose, and Ralf Schenkel, Eds.). CRC Press, 2014: 413-436
http://www.crcpress.com/product/isbn/9781466582408

[w.3, 2009] is the first paper that records how window based selection in our implementation of the C-SPARQL engine\(^1\) outperforms the standard FILTER based selection of SPARQL under simple RDF entailment regimes. The reason is straightforward, triples in an RDF stream are ordered by timestamp, the window operator that selects a continuous portion of the stream can be implemented with a linked list and does not require the classical B-trees.

\(^1\)The engine is available for downloading at http://www.streamreasoning.org/download
1. INTRODUCTION

A similar intuition guided the research on C-SPARQL query optimisation under OWL2RL entailment regime (more specially the fragment of OWL2RL that includes the rules of RDFS covered by OWL2RL and those capturing the semantics of owl:transitiveProperty and owl:inverseOf). In [c.2, 2010], we present a new algorithm for efficient incremental updates of deductive closures when changes to the knowledge base only come in as a stream observed through a sliding window. The algorithm was empirically shown to outperform state of the art approaches by order of magnitudes. In [o.2, 2014], we took the chance to write a book chapter to go back to the work presented in [c.2, 2010] and to better present it together with an up-to-date comparison with state-of-the-art. For this reason, in this thesis I only include [o.2, 2014].

More evaluations w.r.t. input throughput and scalability of the C-SPARQL Engine are presented in the collection of papers grouped in Chapter 6.

Chapter 5: Inductive Stream Reasoning

The paper, proposed in this chapter, provide an answer (in the Social Media Analytics domain) to sub-question SQ.3 – Is it possible to cope with the noisy and incomplete nature of data streams?.


http://doi.ieeecomputersociety.org/10.1109/MIS.2010.142

The paper proposes an architecture that combines deductive and inductive stream reasoning modules for social media analytics. Specifically, we analysed the stream of interactions of Glue’s\(^1\) users with webpages. The paper experimentally shows that it is possible and effective to combine deductive and inductive stream reasoning method to predict user interaction with webpages. The predictions can be used to recommend

\(^{1}\)Glue (http://getglue.com) is a social network that lets users connect to each other and share Web navigation experiences. In addition, Glue uses semantic recognition techniques to identify books, movies, and other similar topics and publishes them in the form of data streams. Users can observe the streams and receive recommendations on interesting findings from their friends.
webpages to explore to Glue’s users. Data incompleteness is handled both in the
deductive stream reasoning, where inferable facts are added to the stream, and in the
inductive stream reasoning part, where the probability of interaction between a user
and a webpage is inductively materialised. The probability, with which the triples are
annotated, also accounts for some level of noise introduced by the semantic recognition
techniques used by Glue. It also worth to note that multiple inductive materialisation
are computed on different time window. While the one on a long-term time windows
allows to predict long-lasting patterns, the short-term one allows to predict hypes.

Chapter 6: Deployments

The following group of papers document practical cases where processing data streams
at semantic level is an effective choice:

w.4 Supporting Environmental Information Systems and Services Realization with
the Geo-Spatial and Streaming Dimensions of the Semantic Web. Proceedings of
the Workshop Environmental Information Systems and Services - Infrastructures
and Platforms. Bonn, Germany, October 6-8, 2010.

w.5 Irene Celino, Daniele Dell’Aglio, Emanuele Della Valle, Yi Huang, Tony Kyung-il
Lee, Seon-Ho Kim, Volker Tresp: Towards BOTTARI: Using Stream Reasoning
to Make Sense of Location-Based Micro-posts. ESWC Workshops 2011: 80-87
http://dx.doi.org/10.1007/978-3-642-25953-1_7

j.5 Marco Balduini, Irene Celino, Daniele Dell’Aglio, Emanuele Della Valle, Yi Huang,
Tony Kyung-il Lee, Seon-Ho Kim, Volker Tresp: BOTTARI: An augmented rea-
ality mobile application to deliver personalized and location-based recommenda-
tions by continuous analysis of social media streams. J. Web Sem. 16: 33-41
(2012)
http://dx.doi.org/10.1016/j.websem.2012.06.004

w.6 Marco Balduini, Emanuele Della Valle, Daniele Dell’Aglio, Mikalai Tsytsarau,
Themis Palpanas, Cristian Confalonieri: Twindex Fuorisalone: Social Listening
of Milano during Fuorisalone 2013. ESWC (Satellite Events) 2013: 327-336
1. INTRODUCTION

http://dx.doi.org/10.1007/978-3-642-41242-4_59


http://dx.doi.org/10.1007/978-3-642-41338-4_1


http://dx.doi.org/10.3233/SW-130107

As we wrote in the vision papers grouped in Chapter 2, Stream Reasoners, if shown to be effective, have the potential to be deployed in many fields. Specific requirements may largely vary, but we can mainly distinguish between two types of data streams: those originating from sensor networks (e.g., traffic sensors, weather sensors, operation control sensors, etc.) and those from social networks. The former are mostly numerical. The need for processing them at semantic level arises from the heterogeneity of sensors and the need to contextualise the streamed information using a rich description of the sensors and their surroundings. The latter may appear mostly textual, but a micro-post has a graph structure. Consider, for instance, a tweet: it is a short text posted by a user that may contain multiple hashtags, may point to webpages, may retweet another tweet, may reply to a user, may be directed to a number of users, may be geo-taged, and it may be semantically enriched with links to named entities (published as linked data on the Web) decorated with the sentiment (positive/neutral/negative) expressed by the user. This graph structure calls for an RDF representation and the short natural language content requires ontological models to be interpreted.

In the early exploratory papers, where the data sets were synthetic, we often targeted use cases centred on Sensor Networks: [c.1, 2008:o.1, 2009:j.2, 2010:j.3, 2010:c.2, 2010] envision the application of Stream Reasoning in traffic monitoring, whereas [w.1, 2010:j.3, 2010:w.4, 2010] do it in oil production and weather monitoring. The more we investigated the Sensor Network domain, the less we found it appropriate for our
1.2 Summary of the chapters

C-SPARQL Engine, which assumes to operate on RDF streams. In that domain, the data has little structure and it is naturally represented in relational format. On the contrary, deployments targeting Social Media Analytics were progressively more successful, because Social Media have rich and variable structure and, as argued above, micro-posts are naturally represented as small graphs in the larger social graph.

A use case, in the Social Media Analytics domain, was first conceived in [j.1, 2009], where we advocate the possibility to monitor public health through social media. A first experiment was performed with SIEMENS on 245,860 interactions made by 2,457 users with tens of thousands of Webpage on the Glue social network\(^1\); the ability to recommend Webpages to users is documented in [j.4, 2010]. This drew the attention of Saltlux (a Korean company in the top-10 Semantic technology enterprises worldwide) with which we developed BOTTARI – an augmented reality mobile application to deliver personalised and location-based recommendations by continuous analysis of social media streams.

BOTTARI’s requirements are documented in [w.2, 2010]. BOTTARI won the Semantic Web Challenge in 2011 [j.5, 2011]. For [j.5, 2011], we ran experiments that are an order of magnitude more extensive than those ran for [j.4, 2010]. We considered 200 million tweets posted in three years by 31,369 users originating from the district of Insadong in Seoul where 245 restaurants are located. The dataset is incomplete\(^2\) and noisy\(^3\). The results confirmed those obtained in [j.4, 2010] and allowed us to experimentally assess that forgetting old enough information is convenient: the best recommendations are obtained using a window of 30 days. Further investigations [j.7, 2014] show that this can only be partially explained with seasonal effects. Even after incorporating the seasonal effect the inductive reasoning models, the most effective recommendation method remains the one that combines the inductive reasoning and the top-k restaurants of the last 30 days. Within the scope of the BOTTARI investigations, we also tested the scalability of our solution and, by accelerating the stream of tweets, we were able to process up to 15,000 tweets/sec on a \(€150/\text{month}\) share in the cloud with 2 cores and 8 GB RAM.

\(^1\)http://getglue.com/ (closed down in 2013 and reopened in 2015 with a different content)
\(^2\)Only 41% of the users express a positive opinion for at least one restaurant.
\(^3\)The same user expressed different opinion about the same restaurants in different moments.
1. INTRODUCTION

Those good results pushed us to engineer the Streaming Linked Data framework (SLD) [c.3, 2013] and to further prove the effectiveness of our Stream Reasoning solution with even more data and for even more real-world applications. During the London Olympics 2012, we experimentally deployed SLD for tracking movements and attention of crowds in real time analysing social streams during the opening ceremony [c.4, 2013]. The application, which processed 35812 tweets in 21 hours, was one of the 5 finalists of the Semantic Web Challenge 2012. A more complex application (namely, Twindex Fuorisalone) was realised in spring 2013 to socially listen to Milano Design Week 2013 through an integrated analysis of Twitter and Instagram. Twindex Fuorisalone was used by 12 thousand unique visitors during Milano Design Week 2013 and won the third price of the AI Mashup Challenge 2013 [w.6, 2013:c.3, 2013]. In 2015, SLD-based social listening was launched as a product of Fluxedo s.r.l., a spin-off of Politecnico di Milano, and deployed for monitoring social media about events, cities and brands.

Chapter 7: Benchmarking

Deployments are appropriate as reality checks for innovative technologies, but, now that Stream Reasoning is established, steady improvement requires comparative evaluation among alternative solutions. As demonstrated in the database community, the establishment of adequate benchmarks [Gray, 1992] is a key ingredient in this direction. Those benchmarks can help to foster broader and deeper tests of the four hypothesis formulated in this thesis.

The papers grouped in this chapter are contributions to the creation of a community-shared benchmark for Stream Reasoning:


http://dx.doi.org/10.1007/978-3-642-38288-8_21


1.2 Summary of the chapters

c.5 Daniele DellAglio, Jean-Paul Calbimonte, Marco Balduini, Oscar Corcho and Emanuele Della Valle: On Correctness in RDF stream processor benchmarking. International Semantic Web Conference (2) 2013: 326-342

http://dx.doi.org/10.1007/978-3-642-41338-4_21

As of the summer 2014, when this thesis is written, two benchmarks exist for Stream Reasoning (SRBench [Zhang et al., 2012] and LSBench [Le-Phuoc et al., 2012]). In [c.4, 2013] we take a critical view on them as well as on those proposed in the DSMS (Linear Road Benchmark [Arasu et al., 2004]) and CEP (Fast Flower Delivery [Luckham, 2001]) communities. We point out which lessons can be learned from DSMS and CEP benchmarks, and highlight how to extend existing Stream Reasoning benchmarks to peck Stream Reasoners where it really hurts. [w.8, 2013] and [c.5, 2013] report on our steps into the direction indicated by [c.4, 2013].

Chapter 8: Outlook

This final chapter contains only one paper:


http://dx.doi.org/10.3233/SW-2012-0085

In the paper, we observe that more and more applications require real-time processing of massive, dynamically generated, ordered data and that order is an essential factor as it reflects recency or relevance. Learning from the effectiveness of exploiting order in reasoning on streaming data, we hypothesise that integrating ordering with reasoning may allow semantic technologies to meet the needs of such applications. 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.
Chapter 9: Conclusions

This final chapter wraps up the results of my seven years of research on Stream Reasoning. For each sub-research question illustrated in Section 1.1, I provide a positive answer based on my work, I compare the results documented in this thesis with those obtained in investigations carried out by other groups, and I briefly discuss open issues and future research directions.

1.3 Collaborations

The scientific publications presented in this thesis were written in collaboration with other researchers. I’d like to acknowledge their contribution and to thank them:

1. Politecnico di Milans colleagues:
   
   (a) Prof. Stefano Ceri who had the initial intuition about the value of introducing data streams to the Semantic Web community,
   
   (b) Marco Balduini, Davide Barbieri, Daniele Braga, Michael Grossniklaus and, again, Prof. Stefano Ceri who helped conceiving the C-SPARQL Engine,
   
   (c) once again to Davide Barbieri who engineered most of the C-SPARQL Engine as part of his PhD Thesis,
   
   (d) once again to Marco Balduini who engineered most of Streaming Linked Data framework and recently started a PhD on Stream Reasoning related topics,
   
   (e) Daniele Dell’Aglio who supported the design and realisation of BOTTARI and has almost finished a PhD on Stream Reasoning related topics,
   
   (f) Andrea Pozzetti and Christian Marazzi who daily work on real-world deployments of Stream Reasoning technologies, and
   
   (g) Mirko Bratomi, and Marco Regaldo – Politecnico di Milano’s Master students that assisted in the design and development of the prototypes.

2. SIEMENS’s colleagues:

   (a) Volker Tresp who had the initial intuition about the value of inductive stream reasoning, and
1.3 Collaborations

(b) Yi Huang who conceived and realised the SUNS inductive reasoner used as inductive Stream Reasoner in our joint work on Social Media Analytics.

3. Saltlux’s colleagues:

(a) Tony Lee who imagined the business value of an application like BOTTARI and selected our Deductive and Inductive Stream Reasoning technologies to prototype it, and

(b) Seonho Kim that designed and realised the augmented reality interface of BOTTARI.

4. CEFRIEL’s colleagues:

(a) Irene Celino who elicited the requirements of BOTTARI, helped design it and coordinated its realisation, and

(b) Cesare Colombo who has been constantly supporting the Semantic Web activities in CEFRIEL.

5. other colleagues:

(a) all the other co-authors of the papers presented in this thesis,

(b) prof. Frank van Harmelen, prof. Dieter Fensel, prof. Heiner Stuckenschmidt who co-developed with me the Stream Reasoning vision, and

(c) Stefan Schlobach, Markus Krtzsch, Alessandro Bozzon and prof. Ian Horrocks who co-developed with me the Order Aware Reasoning vision.

The work presented in this thesis was partially funded by the EU Project FP7-215535 Large Knowledge Collider (LarKC) and by the Search Computing project funded by the European Research Council (ERC) granted to prof. Stefano Ceri.