Stream and Complex Event Processing

The realm of stream reasoning

G. Cugola    E. Della Valle

Politecnico di Milano

cugola@elet.polimi.it
dellavalle@elet.polimi.it

A. Margara

Vrije Universiteit Amsterdam

a.margara@vu.nl
Course outline

• History and principles of stream computing and complex event processing
  • Description of the area
  • Typical applications
  • Challenges
• A modeling framework for IFP systems
  • Functional model
  • Processing model
  • Deployment model
  • Interaction model
  • Data model
  • Time model
  • Rule model
  • Language model
• The realm of stream reasoning
  • A brief introduction to the semantic Web technologies
  • From stream processing to stream reasoning

• Distributing to survive: The "operator placement" problem
  • Theory
  • Algorithms
• On managing uncertainty in data and rules
  • A model of uncertainty for information flow processing systems
• Discovering existing systems
  • Complex event processing systems in practice
  • Data streaming systems in practice
  • Stream reasoning systems in practice
• On benchmarking Information Flow Processing Systems
  • The problem
  • Possible solutions
• Putting it all together
  • A practical scenario to test IFP systems
• Experience report
It’s the Information Society, baby

Oil operations

Traffic

Financial markets

Social networks

...generate data!
It’s the Information Society, baby

...and have to analyze data in real time

In a well in progress to drown, how long time do I have given its historical behavior?

Is public transportation where the people are?

Can we detect any intra-day correlation clusters among stock exchanges?

Who is driving the discussion about the top 10 emerging topics?
Motivation

New Requirements → New Challenges

Typical Requirements

- Processing Streams
- Large datasets
- Reactivity
- Fine-grained information access
- Modeling complex application domains

- Continuous semantics
- Scalable processing
- Real-time systems
- Powerful query languages
- Rich ontology languages
Motivation

Are DSMS/CEP ready to address them?

Typical Requirements

- Processing Streams
- Large datasets
- Reactivity
- Fine-grained information access
- Modeling complex application domains

DSMS/CEP

- Continuous semantics
- Scalable processing
- Real-time systems
- Powerful query languages
- Rich ontology languages
Motivation

Is Semantic Web ready to address them?

- The Semantic Web, the Web of Data is doing fine
  - RDF, RDF Schema, SPARQL, OWL, RIF
  - well understood theory,
  - rapid increase in scalability

- BUT it pretends that the world is static
  or at best a low change rate
  both in change-volume and change-frequency
  - ontology versioning
  - belief revision
  - time stamps on named graphs

- It sticks to the traditional one-time semantics
Motivation

New Requirements → New Challenges

Typical Requirements

• Processing Streams
• Large datasets
• Reactivity
• Fine-grained information access
• Modeling complex application domains

Semantic Web

- Continuous semantics
- Scalable processing
- Real-time systems
- Powerful query languages
- Rich ontology languages

Stream & Complex Event Processing - Stream Reasoning
Motivation

New Requirements call for Stream Reasoning

Typical Requirements

- Processing Streams
- Large datasets
- Reactivity
- Fine-grained information access
- Modeling complex application domains

- Continuous semantics
- Scalable processing
- Real-time systems
- Powerful query languages
- Rich ontology languages

Stream Reasoning
Stream Reasoning Definition [IEEE-IS2010]

- Making sense
  - in real time
  - of multiple, heterogeneous, gigantic and inevitably noisy data streams
  - in order to support the decision process of extremely large numbers of concurrent user

Note: making sense of streams necessarily requires processing them against rich background knowledge, an unsolved problem in database
Fitting data streams in the Semantic Web architecture

Virtual RDF stream

RDF-DSMS/CEP mapping

DSMS/CEP data model

Existing DSMS/CEP

adapter

Continuous SPARQL service

source http://www.w3.org/DesignIssues/diagrams/sw-double-bus.png
Concept

Research Challenges

• Relation with DSMSs and CEPs
  • Just as RDF relates to data-base systems?
• Data types and query languages for semantic streams
  • Just RDF and SPARQL but with continuous semantics?
• Reasoning on Streams
  • Theory
  • Efficiency
  • Scalability
• Dealing with incomplete & noisy data
  • Even more than on the current Web of Data
• Distributed and parallel processing
  • Streams are parallel in nature, ...
• Engineering Stream Reasoning Applications
  • Development Environment
  • Integration with other technologies
  • Benchmarks
Running Example

Social Media Analytics in BOTTARI [JWS2012]

http://streamreasoning.org/demos/bottari
Running Example

Data Model Used in BOTTAII

sioc:UserAccount
  sioc:id(xsd:string)

sioc:CreatorOf
  sioc:has_creator

sioc:Post
  sioc:content(xsd:string)

sr:TwitterUser
  sr:screenName(xsd:string)

sr:NamedPlace

geo:SpatialThing
  geo:lat (xsd:float)
  geo:long(xsd:float)

sr:Tweet
  sr:messageID(xsd:string)
  sr:messageTimeTimestamp(xsd:string)

sr:talksAbout
  sr:reply
  sr:retweet
  sr:talksAboutNeutrally
  sr:talksAboutPositively
  sr:talksAboutNegatively

sr:TalkAbout

sr:following

sr:twitterUser

sr:hasCreator

sr:twd:discuss

sr:twd:post
Running Example

Streaming vs. Background Information

User related background knowledge

Point of Interest related background knowledge

Data stream
Achievements

• **RDF Streams**
  • Notion defined

• **C-SPARQL**
  • *Syntax and semantics* defined as a SPARQL extension
  • *Engine* designed and implemented

• **Experiments with C-SPARQL** under simple RDF entailment regimes
  • *window based selection* of C-SPARQL *outperforms* the standard *FILTER based selection*
  • *algebraic optimizations* of C-SPARQL queries are possible
  • *high throughputs*
  • *Complex event* can be *detected* using a network of C-SPARQL queries *at high throughputs*

• **Experiment with C-SPARQL under RDFS++ entailment regimes**
  • efficient incremental updates of deductive closures investigated
  • our approach outperform state-of-the-art when updates comes as stream
Achievements

Outline

• RDF Streams
  • Notion defined

• C-SPARQL
  • Syntax and semantics defined as a SPARQL extension
  • Engine designed and implemented

• Experiments with C-SPARQL under simple RDF entailment regimes
  • window based selection of C-SPARQL outperforms the standard FILTER based selection
  • algebraic optimizations of C-SPARQL queries are possible
  • high throughputs
  • Complex event can be detected using a network of C-SPARQL queries at high throughputs

• Experiment with C-SPARQL under RDFS++ entailment regimes
  • efficient incremental updates of deductive closures investigated
  • our approach outperform state-of-the-art when updates comes as stream
RDF Stream Data Type

- Ordered sequence of pairs, where each pair is made of an RDF triple and its timestamp

\[ ((\text{subj}_i, \text{pred}_i, \text{obj}_i), \tau_i) \]
\[ ((\text{subj}_{i+1}, \text{pred}_{i+1}, \text{obj}_{i+1}), \tau_{i+1}) \]

- Timestamps are not required to be unique, they must be non-decreasing.

E.g.,

\[ (A: Alice \text{ :posts } :post1 >, \ 2010-02-12T13:34:41) \]
\[ (A: post1 \text{ :talksAboutPositively } :LaScala>, \ 2010-02-12T13:34:41) \]
\[ (A: Bob \text{ :posts } :post2 >, \ 2010-02-12T13:36:28) \]
\[ (A: post2 \text{ :talksAboutNegatively } :Duomo>, \ 2010-02-12T13:36:28) \]
Achievements

Outline

• RDF Streams
  • Notion defined

• C-SPARQL
  • Syntax and semantics defined as a SPARQL extension
  • Engine designed and implemented

• Experiments with C-SPARQL under simple RDF entailment regimes
  • window based selection of C-SPARQL outperforms the standard FILTER based selection
  • algebraic optimizations of C-SPARQL queries are possible
  • high throughputs
  • Complex event can be detected using a network of C-SPARQL queries at high throughputs

• Experiment with C-SPARQL under RDFS++ entailment regimes
  • efficient incremental updates of deductive closures investigated
  • our approach outperform state-of-the-art when updates comes as stream
Achievements

Where C-SPARQL Extends SPARQL
Achievements

An Example of C-SPARQL Query

Who are the opinion makers? i.e., the users who are likely to influence the behavior of other users who follow them

REGISTER STREAM OpinionMakers COMPUTED EVERY 5m AS
CONSTRUCT { ?opinionMaker sd:about ?resource }
FROM STREAM <http://streamingsocialdata.org/interactions> [RANGE 30m STEP 5m]
WHERE {
FILTER ( cs:timestamp(?follower) > cs:timestamp(?opinionMaker) && ?opinion != sd:accesses )
}
HAVING ( COUNT(DISTINCT ?follower) > 3 )
An Example of C-SPARQL Query

Who are opinion makers? i.e., the users who are likely to influence behavior of other users who follow them.

REGISTER STREAM OpinionMakers COMPUTED EVERY 5m AS
CONSTRUCT { ?opinionMaker sd:about ?resource } FROM STREAM <http://streamingsocialdata.org/interactions> [RANGE 30m STEP 5]
WHERE {
?follower sioc:follows ?opinionMaker
FILTER ( cs:timestamp(?follower) > cs:timestamp(?opinionMaker) && ?opinion != sd:accesses )
}
HAVING ( COUNT(DISTINCT ?follower) > 3 )

Query registration (for continuous execution)
RDF Stream added as new output format
FROM STREAM clause
WINDOW
Builtins to access timestamps
Aggregates as in SPARQL 1.1
Achievements

Outline

• RDF Streams
  • Notion defined
• C-SPARQL
  • Syntax and semantics defined as a SPARQL extension
  • Engine designed and implemented
• Experiments with C-SPARQL under simple RDF entailment regimes
  • window based selection of C-SPARQL outperforms the standard FILTER based selection
  • algebraic optimizations of C-SPARQL queries are possible
  • high throughputs
  • Complex event can be detected using a network of C-SPARQL queries at high throughputs
• Experiment with C-SPARQL under RDFS++ entailment regimes
  • efficient incremental updates of deductive closures investigated
  • our approach outperform state-of-the-art when updates comes as stream
FROM STREAM Clause - Types of Window

- **physical**: a given number of triples
- **logical**: a variable number of triples which occur during a given time interval (e.g., 1 hour)
  - **Sliding**: they are progressively advanced of a given STEP (e.g., 5 minutes)
  - **Tumbling**: they are advanced of exactly their time interval
Achievements

Efficiency of Evaluation [IEEE-IS2010]

- window based selection of C-SPARQL outperforms the standard FILTER based selection
Achievements

Outline

• **RDF Streams**
  • Notion defined

• **C-SPARQL**
  • *Syntax* and *semantics* defined as a SPARQL extension
  • *Engine* designed and implemented

• **Experiments with C-SPARQL** under simple RDF entailment regimes
  • *window based selection* of C-SPARQL *outperforms* the standard *FILTER based selection*
  • *algebraic optimizations* of C-SPARQL queries are possible
  • *high throughputs*
  • *Complex event can be detected using a network of C-SPARQL queries at high throughputs*

• **Experiment with C-SPARQL** under RDFS++ entailment regimes
  • efficient incremental updates of deductive closures investigated
  • our approach outperform state-of-the-art when updates comes as stream
Achievements

Algebraic optimizations of C-SPARQL [EDBT2010]

- Several transformations can be applied to algebraic representation of C-SPARQL
- Some recalling well known results from classical relational optimization
  - Push of FILTERs and projections
- Some being more specific to the domain of streams
  - Push of aggregates
Achievements

algebraic optimizations of C-SPARQL [EDBT2010]

- Push of filters and projections

![Graph showing performance metrics for different window sizes and processing types.](Image)
Achievements

Outline

• RDF Streams
  • Notion defined

• C-SPARQL
  • Syntax and semantics defined as a SPARQL extension
  • Engine designed and implemented

• Experiments with C-SPARQL under simple RDF entailment regimes
  • window based selection of C-SPARQL outperforms the standard FILTER based selection
  • algebraic optimizations of C-SPARQL queries are possible
  • high throughputs
  • Complex event can be detected using a network of C-SPARQL queries at high throughputs

• Experiment with C-SPARQL under RDFS++ entailment regimes
  • efficient incremental updates of deductive closures investigated
  • our approach outperform state-of-the-art when updates comes as stream
Achievements

High Throughputs [JWS2012]

![Graph showing high throughputs for different stream reasoning techniques. The x-axis represents the number of tweets in the proportion of an RDF stream used for testing, ranging from 0 to 2000. The y-axis represents the input throughput (tweets/sec), ranging from 0 to 20000. The graph compares SLD using C-SPARQL and recording on TDB.]
Achievements

Outline

• **RDF Streams**
  - Notion defined

• **C-SPARQL**
  - *Syntax* and *semantics* defined as a SPARQL extension
  - *Engine* designed and implemented

• **Experiments with C-SPARQL** under simple RDF entailment regimes
  - *window based selection* of C-SPARQL outperforms the standard *FILTER based selection*
  - *algebraic optimizations* of C-SPARQL queries are possible
  - *high throughputs*
  - *Complex event can be detected using a network of C-SPARQL queries at high throughputs*

• **Experiment with C-SPARQL under RDFS++ entailment regimes**
  - efficient incremental updates of deductive closures investigated
  - our approach outperform state-of-the-art when updates comes as stream
Achievements

Complex Event Detection as stream compositions

- e.g., continuous detection of blizzards by analyzing multiple streams of data generated by weather sensors spread across a continental area

Blizzard: a severe snowstorm with high winds and low visibility lasting at least three hours
Achievements

Complex Event Detection as stream compositions

- Stream & Complex Event Processing - Stream Reasoning
Achievements

Complex Event Detection as stream compositions

snowfall \ + \ strong winds \ + \ low temp \ \rightarrow\ blizzards
Achievements

Detecting crowd movements London 2012

An event is starting if in the last 2 hours I saw continuous flow of people exiting Stratford station, funnelling through Stratford walk, entering the stadium

London 2012 Opening Ceremony at the Olympic Stadium

http://www.streamreasoning.org/demos/london2012
Achievements

Outline

- **RDF Streams**
  - Notion defined
- **C-SPARQL**
  - *Syntax* and *semantics* defined as a SPARQL extension
  - *Engine* designed and implemented
- **Experiments with C-SPARQL** under simple RDF entailment regimes
  - *window based selection* of C-SPARQL *outperforms* the standard *FILTER based selection*
  - *algebraic optimizations* of C-SPARQL queries are possible
  - *Complex event* can be *detected* using a network of C-SPARQL queries *at high throughputs*
- **Experiment with C-SPARQL** under RDFS++ entailment regimes
  - efficient incremental updates of deductive closures investigated
  - our approach outperform state-of-the-art when updates comes as stream
Where’s the Reasoning? 1/3

- It’s normal that you don’t see it when looking at the query only
- Memo: two approach for reasoning about query answering
  - Data-driven
  - Query-driven
Achievements

Where’s the Reasoning?

• Query driven reasoning directly applies to data streams
• Memo ...

... in the context of streaming data

Well, things are a bit more complex, because no standard continuous query language exists
Where’s the Reasoning?

At a first look also data-driven should be easy to port.

The problem is that materialization (the result of data-driven processing) are very difficult to decrement efficiently.

- State-of-the-art: DRed algorithm
  - Over delete
  - Re-derive
  - Insert
Achievements

Example of C-SPARQL and Reasoning 1/2

• Query: can we measure the impact of a tweet?
• Twitter allows two traceable ways of discussing a tweet:
  • reply: a user reply to a tweet of another user (it always retweet the original tweet)
  • retweet: a user propagates to his/her followers an interesting tweet
• For example
Example of C-SPARQL and Reasoning 2/2

What impact have I been creating with my tweets in the last hour? Let’s count them …

REGISTER STREAM OpinionSpreading COMPUTED EVERY 30s AS
SELECT ?tweet (count(?tweet) AS ?impact)
FROM STREAM <http://ex.org> [RANGE 60m STEP 10m]
WHERE {
  :t1 sr:discuss ?tweet
}

7!
The Intuition of DRed Algorithm

Let’s assume that we have the following materialized graph:

While inserts are not problematic, deletion are difficult to handle. If we delete $t_2$, we have:

1. **overestimate the impact of the deletion** and mark for deletion $t_4$-$t_1$ that can be derived by $t_4$-$t_2$ and $t_2$-$t_1$.

2. **look for alternative derivation** of $t_4$-$t_1$ and eventually find the chain $t_4$-$t_3$ and $t_3$-$t_1$.

Achievements

Stream & Complex Event Processing - Stream Reasoning
Achievements

our approach [ESWC2010] 1/2

• Assumption
  • Insertions and deletions are triples respectively entering and exiting the window
  • The window size is known

• Therefore
  • The time when each triple will expire is known and determined by the window size
    • E.g. if the window is 10s long a triple entering at time t will exit at time t+10s
  • Note: all knowledge can be annotated with an expiration time
    • i.e., background knowledge is annotated with $+\infty$
Achievements

our approach [ESWC2010] 2/2

• The algorithm
  1. deletes all triples (asserted or inferred) that have just expired
  2. computes the entailments derived by the inserts,
  3. annotates each entailed triple with an expiration time, and
  4. eliminates from the current state all copies of derived triples except the one with the highest timestamp.

• learn more
  • http://www.slideshare.net/emanueledellavalle/incremental-reasoning-on-streams-and-rich-background-knowledge
### Achievements

#### Intuition

<table>
<thead>
<tr>
<th>TS</th>
<th>Triples in the Window</th>
<th>Entailments in the Window</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A [11] → B</td>
<td>A [11] \rightarrow C</td>
</tr>
<tr>
<td>2</td>
<td>A [11] → B [12] → C [13]</td>
<td>A [11] \rightarrow C [12]</td>
</tr>
<tr>
<td>3</td>
<td>A [11] → B [12] → C [13] → D [14]</td>
<td>A [11] \rightarrow C [12] \rightarrow D [14]</td>
</tr>
<tr>
<td>4</td>
<td>[14] → E [14]</td>
<td>A [11] → B [12] → C [13] → D [14]</td>
</tr>
</tbody>
</table>

**Multiple Derivation**

If a longer lasting derivation of a triple already entailed is found, the expiration time of the entailed triple is updated.

**Efficient Deletion**

Deletion is achieved by lookups of triples in an hashmap. No over deletion occurs.
Comparative Evaluation on Materialization

- **base-line**: re-computing the materialization from scratch
- **state-of-the-art** [Ceri1994, Volz2005]
- **our approach** [ESWC2010]
Achievements

Comparative Evaluation on Query Answering

- comparison of the average time needed to answer a C-SPARQL query using
  - backward reasoner
  - the naive approach of re-computing the materialization
  - our approach

<table>
<thead>
<tr>
<th></th>
<th>Backward reasoning</th>
<th>naive approach</th>
<th>incremental-stream</th>
</tr>
</thead>
<tbody>
<tr>
<td>query</td>
<td>5,82</td>
<td>1,61</td>
<td>1,61</td>
</tr>
<tr>
<td>materialization</td>
<td>0</td>
<td>15,91</td>
<td>0,28</td>
</tr>
</tbody>
</table>
Research Challenges vs. Achievements

- Relation with DSMSs and CEPs
  - Notion of RDF stream: alternative solutions can be investigated
- Data types and query languages for semantic streams
  - C-SPARQL: work in progress in FZI&AIFB [1,2] DERI [3], UPM [4]
- Reasoning on Streams
  - Theory: work in progress in Potsdam&DERI [9,10]
  - Efficiency: work in progress in ISTI-Innsbruck [5]
  - Scalability: work in progress in IBM&VUA [6]
- Dealing with incomplete & noisy data
  - Even more than on the current Web of Data: some initial joint work with SIEMENS only [IEEE-IS2010]
- Distributed and parallel processing
  - Streams are parallel in nature, ...: work in progress in IBM&VUA [6]
- Engineering Stream Reasoning Applications
  - Demonstrative applications in Social Media and Sensor Networks
  - Development Environment: work in progress in UPM [7]
  - Benchmarks: work in progress in CWI&UPM [8], DERI [12]
Conclusions

• The **Semantic Web community positively answered to** the call at investigating **stream reasoning**

• A number of **work in progress** are rapidly developing this field, but we are only at the very beginning
Credits

• Politecnico di Milano’s colleagues
  • Prof. Stefano Ceri who had the initial intuition about the value of introducing data streams to the semantic Web community
  • Marco Balduini, Davide Barbieri, Daniele Braga, Stefano Ceri and Michael Grossniklaus who helped concieving the C-SPARQL Engine and the Streaming Linked Data Framework

• People I directly worked with on the topic
  • CEFRIEL: Irene Celino, and Danile Dell’Aglio
  • Saltlux: Seonho Kim, and Tony Lee
  • SIEMENS: Yi Huang, and Volker Tresp
  • STI-Innsbruck: prof. Dieter Fensel and Srdjan Komazec
  • UO: prof. Ian Horrocks and Markus Krötzsch
  • VUA: prof. Frank van Harmelen and Stefan Schlobach

• The broader research community that showed interest in stream reasoning
Downloads

• C-SPARQL Engine (no reasoning support)
  • A ready to go pack for eclipse
    • http://streamreasoning.org/download
  • Source code available on request
References

My papers

- [JWS2012] M. Balduini; I.Celino; E. Della Valle; D.Dell'Aglio; Y. Huang; T. Lee; S. Kim; V. Tresp: *BOTTARI: an Augmented Reality Mobile Application to deliver Personalized and Location-based Recommendations by Continuous Analysis of Social Media Streams.* JWS. 2012. IN PRESS.
References

Other groups’ papers


[8] Ying Zhang, Minh-Duc Pham, Oscar Corcho and Jean Paul Calbimonte. SRBench: A Streaming RDF/SPARQL Benchmark ISWC 2012: IN PRESS


References

Background papers

Thank You! Questions?

Much More to Come!
Keep an eye on
http://www.streamreasoning.org