Towards Efficient Schema-Enhanced Pattern Matching over RDF Data Streams

Srdjan Komazec and Davide Cerri
• Data streams are becoming common on the Web
  – Stock exchange tickers
  – Weather information
  – Sensor readings
  – Social networking activity notifications

• Collecting, integrating and sharing data streams
  – Pachube platform

• Interpreting data streams as event sources
  – Vision of the “Web of Events”
Introduction
Technologies for stream/event processing

- **(Complex) Event Processing (CEP)**
  - Established computing paradigm providing approaches and techniques to **timely** process streams of events
  - Built around the notion of **event patterns**
  - E.g. Tibco Business Events, Oracle CEP, Esper, Streambase, etc.

- **Data Stream Management Systems (DSMS)**
  - A system which enables querying and maintenance of data in data streams
  - Built around the notion of **continuous queries** against data streams
  - E.g., Aurora and Borealis, STREAM, TelegraphCQ, NiagaraST, etc.
Stream/event processing on the Web needs to cope with openness, heterogeneity and required scalability.

**Semantic Web technologies**
- Facilitating data integration by using machine processable descriptions to reconcile heterogeneities (e.g., Semantic Sensor Web)

**Impact of SW technologies over stream processing and vice versa**
- Entailments and data/event stream processing
- Data volatility (temporality) and Semantic Web technologies
Related Work

ETALIS

- Event-driven Transaction Logic Inference System
  - A CEP system bringing together the power of logic rule inference and event processing

- Feature rich system
  - Out-of-order events evaluation, aggregate functions computation, dynamic pattern insertion and retraction, several garbage collection policies and consumption strategies, event operators, count-based sliding windows, …

- Relies on a Prolog engine

- Support for RDF data through EP-SPARQL
  - Extends SPARQL with temporal operators at the level of RDF graphs
  - RDFS ontologies and graph patterns are transformed into Prolog statements
Related Work

C-SPARQL

- **Continuous SPARQL**
  - A SPARQL-based DSMS representative enhanced with reasoning capabilities

- **Features**
  - Extends SPARQL with the notions of RDF streams, time windows, aggregation functions, handling of multiple streams, …
  - Suitable for situations when a significant amount of static knowledge needs to be combined with data streams to enable time-constrained reasoning

- Performs incremental RDF Schema-based materialisation of RDF snapshots on top of a triple repository

- Limited support for temporal operators

- Doesn’t provide truly continuous querying (snapshotting)
Introduction

SparkWeave position

ETALIS

SparkWeave

C-SPARQL

Semantic Web Technologies
SparkWeave

Introduction

• Providing efficient pattern matching functionalities against RDF streams
  – Enabling expression of temporal RDF graph dependencies
  – Taking into account RDF Schema entailments
• Uses **Rete algorithm** as the foundation
• Extends Rete with **networking structures** necessary to compute entailments
• Supports **rich set of time-based features**
SparkWeave Constraints

- **SparkWeave operates over a fixed schema**
  - Schema definitions are rarely transported over stream (e.g., sensor readings)

- **SparkWeave does not support background knowledge**
  - The system exploits a memory-intensive approach

- **SparkWeave does not provide generic reasoning capabilities**
  - The system is built for RDF Schema (and few OWL constructs)
• Pattern matching algorithm for implementing production rule systems in many object/many pattern situations.
  - The basis for many popular expert system shells, including CLIPS, Jess, Drools, BizTalk, Rules Engine and Soar.

• Classical Rete architecture is extended with an additional network of entailment nodes called $\varepsilon$ – network

• $\varepsilon$ – network follows Rete principles
  – Dataflow organization
  – Usage of tokens to keep state of computation

• $\varepsilon$ – network is built on top of schema and optimized in accordance to RDF pattern descriptions
  – The network topology is governed by the property hierarchies (with domain and range definitions) and class hierarchies

• Outputs of $\varepsilon$ – network are connected to inputs of $\alpha$ – network
## SparkWeave

### Schema entailment

<table>
<thead>
<tr>
<th>Rule</th>
<th>If</th>
<th>Then add</th>
</tr>
</thead>
<tbody>
<tr>
<td>rdf1</td>
<td>(x p y)</td>
<td>(p rdf:type rdf:Property)</td>
</tr>
<tr>
<td>rdfs2</td>
<td>(p rdfs:domain c) (x p y)</td>
<td>(x rdf:type c)</td>
</tr>
<tr>
<td>rdfs3</td>
<td>(p rdfs:range c) (x p y)</td>
<td>(y rdf:type c)</td>
</tr>
<tr>
<td>rdfs4a</td>
<td>(x p y)</td>
<td>(x rdf:type rdfs:Resource)</td>
</tr>
<tr>
<td>rdfs4b</td>
<td>(x p y)</td>
<td>(y rdf:type rdfs:Resource)</td>
</tr>
<tr>
<td>rdfs5</td>
<td>(p rdfs:subPropertyOf q) (q rdfs:subPropertyOfOf r)</td>
<td>(p rdfs:subPropertyOf r)</td>
</tr>
<tr>
<td>rdfs6</td>
<td>(p rdf:type rdf:Property)</td>
<td>(P rdfs:subPropertyOf p)</td>
</tr>
<tr>
<td>rdfs7</td>
<td>(p rdfs:subPropertyOfOf q) (x p y)</td>
<td>(x q y)</td>
</tr>
<tr>
<td>rdfs8</td>
<td>(c rdf:type rdfs:Class)</td>
<td>(c rdfs:subClassOf rdfs:Resource)</td>
</tr>
<tr>
<td>rdfs9</td>
<td>(c rdfs:subClassOf d) (x rdf:type c)</td>
<td>(x rdf:type d)</td>
</tr>
<tr>
<td>rdfs10</td>
<td>(c rdf:type rdfs:Class)</td>
<td>(c rdfs:subClassOf c)</td>
</tr>
<tr>
<td>rdfs11</td>
<td>(c rdfs:subClassOf d) (d rdfs:subClassOf e)</td>
<td>(c rdfs:subClassOf e)</td>
</tr>
<tr>
<td>rdfs12</td>
<td>(p rdf:type rdfs:ContainerMembershipProperty)</td>
<td>(p rdfs:subPropertyOf rdfs:Member)</td>
</tr>
<tr>
<td>rdfs13</td>
<td>(x rdf:type rdfs:Datatype)</td>
<td>(x rdfs:subClassOf rdfs:Literal)</td>
</tr>
<tr>
<td></td>
<td>(p owl:inverseOf q)</td>
<td>(q owl:inverseOf p)</td>
</tr>
<tr>
<td></td>
<td>(p owl:inverseOf q) (x p y)</td>
<td>(y q x)</td>
</tr>
<tr>
<td></td>
<td>(p rdf:type owl:SymmetricProperty) (x p y)</td>
<td>(y p x)</td>
</tr>
</tbody>
</table>
:UserAccount a rdfs:Class .

:Post a rdfs:Class .

:Tweet a rdfs:Class ;
    rdfs:subClassOf :Post .

:FacebookPost a rdfs:Class ;
    rdfs:subClassOf :Post .

:Place a rdfs:Class .

:hasCreator a rdf:Property ;
    rdfs:domain :Post ;
    rdfs:range :UserAccount .

:creatorOf a rdf:Property ;
    owl:inverseOf :hasCreator .

:relatedTo a rdf:Property ;
    rdfs:domain :Post ;
    rdfs:range :Place .

:talksAbout a rdf:Property ;
    rdfs:subPropertyOf :relatedTo .

:postedFrom a rdf:Property ;
    rdfs:subPropertyOf :relatedTo .
### SparkWeave

#### Time-based features

- **Time window** support
  - Extensions of β-node behaviour for executing time-window boundary checks

- **Temporal operators**
  - Focusing on the interval-based semantics
  - Extensions of join-node behaviour

- **Garbage collection** of discarded triples/tokens
  - Calculation of an object lifetime
  - Incremental and parallel process

- **Event consumption strategies**
  - Extensions of join-node behaviour
  - Currently supports UNRESTRICTED strategy
Conclusions and Further Work

- **SparkWeave**
  - Schema-enhanced pattern matcher operating against RDF data streams
  - Built on top of Rete algorithm extended with an entailment (ε) network
  - Supports RDFS + a few OWL constructs

- **Future work**
  - Integration and implementation of temporal extensions
  - Performance evaluation