Towards Efficient Semantically Enriched Complex Event Processing and Pattern Matching

Syed Gillani[1,2] Gauthier Picard[1] Frédérique Laforest[2]
Antoine Zimmermann[1]

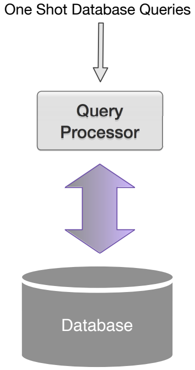
Institute Henri Fayol, EMSE, Saint-Etienne, France [1]

Telecom Saint Etienne, Université Jean Monnet, Saint-Etienne, France [2]

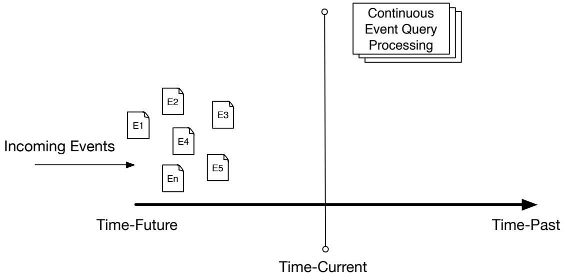
Overview

Traditional Vs Real-Time Data Processing

Traditional Data Processing

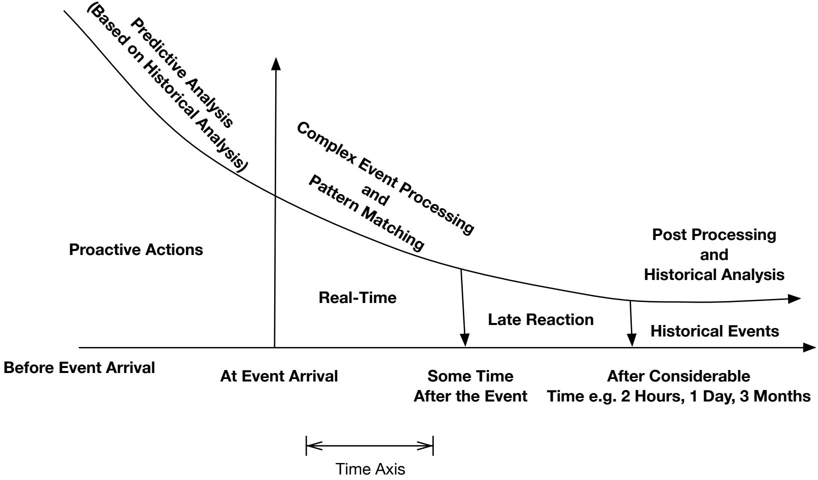


Real-Time Data Processing



Event Arrival Time

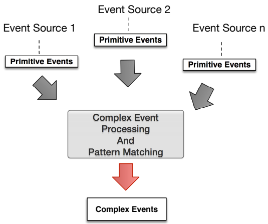
Event Processing Vs Time Axis



*Dr. Adrian Paschke, DemAAL Summer school 2013

Complex Event Processing

- Aggregation, derivation of Primitive Events
- Occurrence and non-occurrence of certain events
- Imposing Temporal Constraints (application of certain rules)
- For Instance
  - Detection of state changes based on observations (If total consumed electricity > 10MWatt)
  - Matching sequence of events that describes a scenario (If A<10 AND B>40 OR B<80 AND C>90)



Overview

SCEP

- Complex Event Processing +Stream Reasoning+ Semantic Technologies (rules & ontologies) + Heterogeneous Data Handling?
- Incoming Stream Reasoning + Background Knowledge
- Distributed into TWO flavours
  - Stream Reasoning (Real Time + Background Information + Aggregation through Windows) (C-SPARQL, CQELS....)
  - Pattern Matching (Sequence, Optional, Negation) (EP-SPARQL)

State-of-the-art SCEP

	Continuous Query	Background Knowledge	Data Model	Event Processing (Per Query)	Historical Data (No Dedicated Management)	Underlying Engine	Parallel and Distributed Multi-Query Processing	Temporal Operators (Pattern Matching)
C-SPARQL	✔	✔	Triple Based	Centralised	✖	ESPER	✔	✖
CQLES	✔	✔	Triple Based	Centralised	✖	ESPER	✔	✖
EP-SPARQL	✔	✔	Triple Based	Centralised	✔	ETAILS	✖	✔
Streaming SPARQL	✔	✔	Triple Based	Centralised	✖	DYNAQUEST	✖	✖
TA-SPARQL	✖	✔	Triple Based	Centralised	✔	TUPELO	✖	✖

*Streaming the Web: Reasoning over Dynamic Data: *Alessandro Margara, Jacopo Urbani, Frank van Harmelen, Henri Bal*

State-of-the-art SCEP

- Complex Pattern Matching (Approaches)
  - Relational Community
    - NFA, EDG, RETE algorithm, Rule based system
  - Semantic Web Community
    - RETE algorithm, Logical Rule based system
  - How about NFA and EDG in SCEP context?
  - NFA and EDG are proven to be the most efficient for Pattern Matching in relational community

*Non-Deterministic Finite Automata

*Event Detection Graphs

Foundational Challenges for SCEP

- *Distributed Event Processing (per Query):* Moving from centralised push based event processing
- *Distributed Temporal Pattern Matching:* Dedicated language for Pattern Matching (Implementation of Kleene Closure, Negation in distributed manner)
- *Historical Management of Events:* Storing and Partitioning of events
- *Defining Event Boundaries:* Triple based to Graph based streaming, preserving graph model to implement Event boundaries
- *Predictive Event Processing:* A new paradigm for SCEP
- *Stream Reasoning + CEP:* Combing two different worlds

Overview

Introduction

Semantic Complex Event Processing

Proposed Approach

- Event and Stream Data Model
- Query Model and Language Specification

Conclusion

Event and Stream Data Model

- Considering RDF as first class citizen (even for temporal reasoning, instead relying on external engines)
- Temporally Annotated RDF Named Graph ($< NG, [ts, te] >$)

```
<http://www.streaminginfo.com/ElecGen> [st1,et1]
        :gen1  :hasName 'PowGen-Sect1'.
        :gen1 :hasLocation 'St-Etienne'.
        :gen1 :hasCurrentPower '60'.
```

Proposed Data Model

- Data Partitioning ==> Optimises query time
- Summarisation ==> Merging of similar NG
- Event Boundaries ==> With NG
- Access Control ==> With NG
- Provenance Tracking ==> With NG
- Fact Assignment ==> With Time Interval

Query Model and Language Specification

- Former Query Models
  - Reliance on Triple-Based Data Model
  - Uses black-box approach (delegation to external Engines)
  - Overhead in query and data translation
  - Query Semantics not suitable for distributed processing per query (SPARQL Extensions...)

PROPOSED QUERY MODEL

Pattern Duration

Stream Source Selection, Temporal Operators

Temporal Pattern Description

KB Integration

```
PREFIX sm: <http:example.com/sm>
PREFIX lv: <http:example.com/lv>
Select *
Within 12 hours
From Stream S1 <http://example.org/streams/
  powersource> Window From Now 10 mins
From Stream S2 <http://example.org/streams/
  weathersource> Window From Now 10 mins
From Stream S3 <http://example.org/streams/
  elecappliance> Window From Now 10 mins
Where {
SEQ ( EVENTPATT A, (EVENTPATT B)+, (EVENTPATT C AND
  EVENTPATT B))

DEFINE EVENTPATT A ON S1 { ?event rdfs:subclassof owl:
  thing; sm:events[ sm:eventType sm:powersource; sm:
  id ?id; sm:power ?genpow]. GRAPH <http://example.
  org/streams/sourcelocation> {?id lv:name ?locName}
  FILTER( ?id = 'gen1', ?pow = '60') }

DEFINE EVENTPATT B ON S2 { ?event rdfs:subclassof owl:
  thing; sm:events[ sm:eventType sm:weathersoruce; sm
  :id ?id; sm:temp ?temp; sm:pressure ?pres]. FILTER(
   ?id = 'Wsource1', ?temp = '20', ?pres='10') }

DEFINE EVENTPATT C ON S3 { ?event rdfs:subclassof owl:
  thing; sm:events[ sm:eventType sm:electricappliance
  ; sm:name ?name; sm:usagepower ?pow; sm:loadclass ?
  load]. FILTER( ?id = 'heater', ?pow <?genpow ?load=
  '10-100Watt') }

 }
```

(a)

Rewritten Subqueries (Stream Processing)

Sub-Query 1 (Event Pattern A)

```
Select *
From Stream S1 <http://example.org/streams/powersource>
    Window From Now 10 mins
Where {
(?event rdfs:subclassof owl:thing;
sm:events[ sm:eventType sm:powersource;
    sm:id ?id;
    sm:power ?pow]. )
FILTER( ?id = 'gen1', ?pow = '60')
}
```

Sub-Query 2 (Event Pattern B)

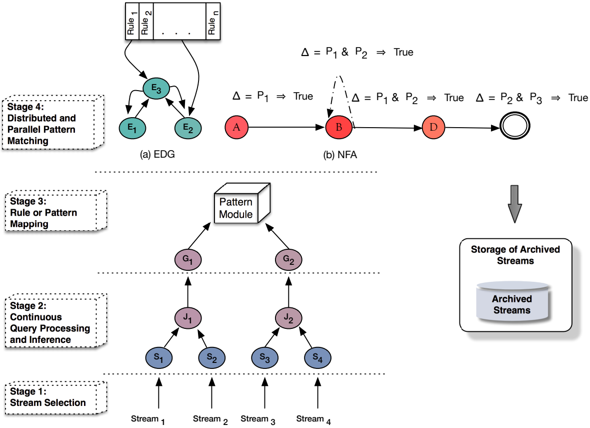
```
Select *
From Stream S2 <http://example.org/streams/
   weathersource> Window From Now 10 mins
Where {
(?event rdfs:subclassof owl:thing;
sm:events[ sm:eventType sm:weathersoruce;
    sm:id ?id;
    sm:temp ?temp;
    sm:pressure ?pres].)
FILTER( ?id = 'Wsource1', ?temp = '20', ?pres='10')
}
```

Sub-Query 3 (Event Pattern C)

```
Select *
From Stream S3 <http://example.org/streams/
   elecappliance> Window From Now 10 mins
Where {
(?event rdfs:subclassof owl:thing; sm:events[ sm:
   eventType sm:electricappliance; sm:name ?name; sm:
   usagepower ?pow; sm:loadclass ?load]. )
FILTER( ?id = 'heater', ?pow <?genpow ?load='10-100Watt
   ')
}
```

(b)

System Overview



$Stream_i$: Incoming Streams
J_i : Join Operations
E_n : Event Nodes
S_k : Select Operators
G_t : Generated Events
A/B/D : NFA States

Proposed Model

- Supports Triple based and NG based data model
- Offers event source based Filtering
- Historical management of events through summarisation (Facts Assignments)
- Provide dedicated design for SCEP (No Data or Query Translation unlike EP-SPARQL and other systems)
- Distributed and parallel sub-query processing with query rewriting

Proposed Model

- Integrating stream processing and CEP
- Offers various new operators including, Sequencing, Kleene Closure and Negation for RDF Graph patterns
- Allows NFA and EDG to be used in the context of SCEP through query rewriting (from Rule based to State based system)

Overview

Introduction

Semantic Complex Event Processing

Proposed Approach

Conclusion

Conclusion

- Annotated RDF NG enables temporal reasoning at RDF level
- Our data/query model and query rewriting allows
  - Annotated NG based event data model
  - Historical management of stream data
  - Integration of various new operators for RDF Graphs (Kleene Closure, Negation)
  - Integration of NFA and EDG in the context of SCEP
  - Parallel and distributed event processing (per query)

Questions?