Semantic Processing of Sensor Event Stream by using External Knowledge Bases

Kia Teymourian and Adrian Paschke
Freie Universität Berlin

Presenter: **Kia Teymourian**

Workshop SSN 2012

ISWC2012 - Boston 9-14. November 2012

Motivation

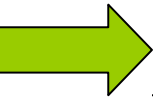
- In some of the use cases **huge amount of Background Knowledge** about **Sensor Events** are available.

- *Fusion of **external Knowledge Bases** with the event stream can improve the **expressiveness**, **agility** and **flexibility** of event processing systems.*

Example – Semantic Enrichment of Events

Query:
Select food products, which include substances capable of causing cancer (**Carcinogen**) and
are *produced in the **Europe***.

Stream of Product IDs

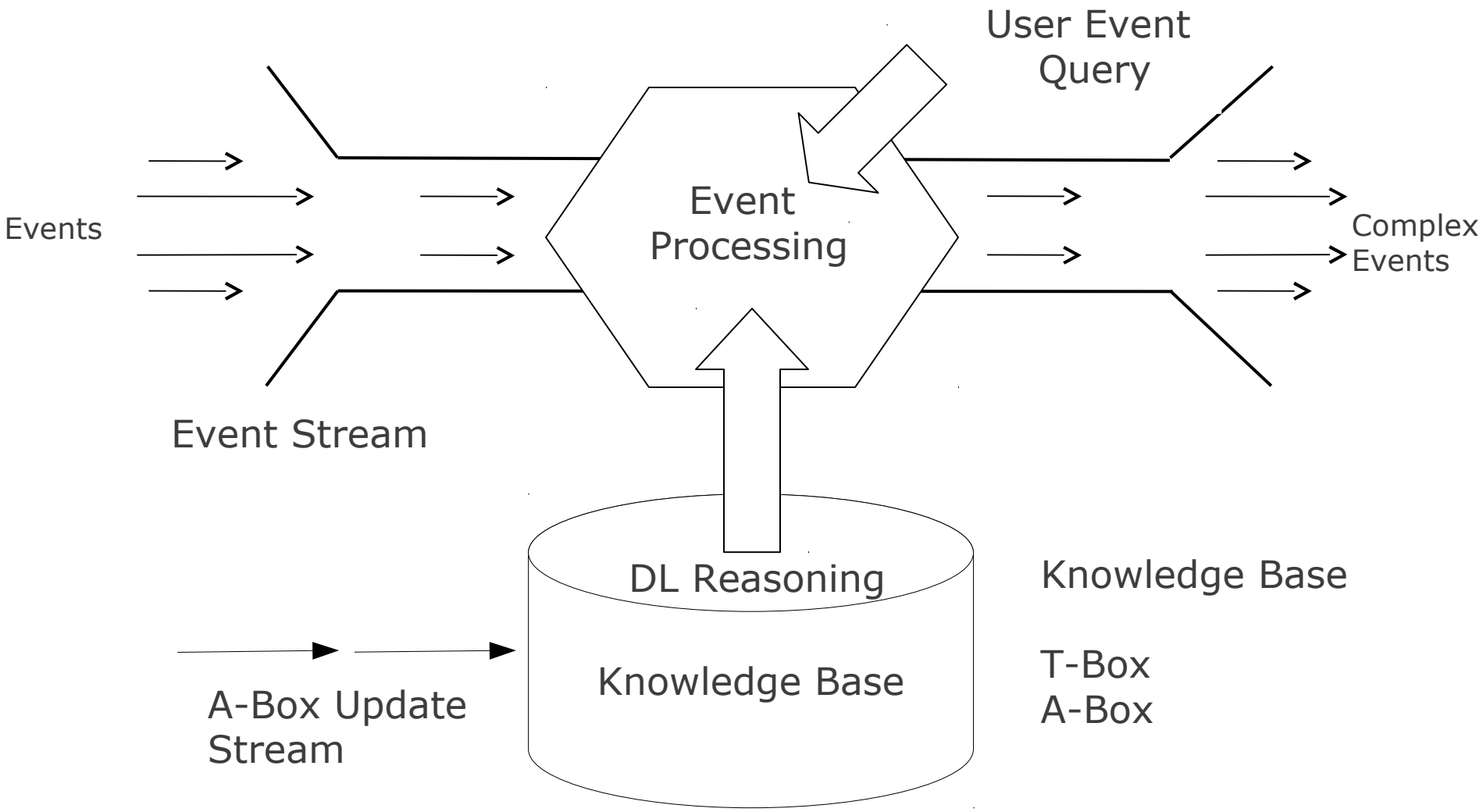


{ (Product_id, "X1234") , (ProductionDate, 5/1/2012) }
{ (Product_id, "X1235") , (ProductionDate, 6/1/2012)}

Knowledge Base

{(:***Company_1***, produce, :**Product_X1**),
(:**Product_X1**, inludes, ***:Tartrazine (E102)***),
(:**Product_X1**, inludes, ***:Erythrosine (E127)***),
(:***Company_1***, production_facilities_in, :**Berlin**),
(:**Berlin,** is_in, :**Germany**),
(:**Germany,** is_in, **:Europe**)}

Knowledge-based Event Processing

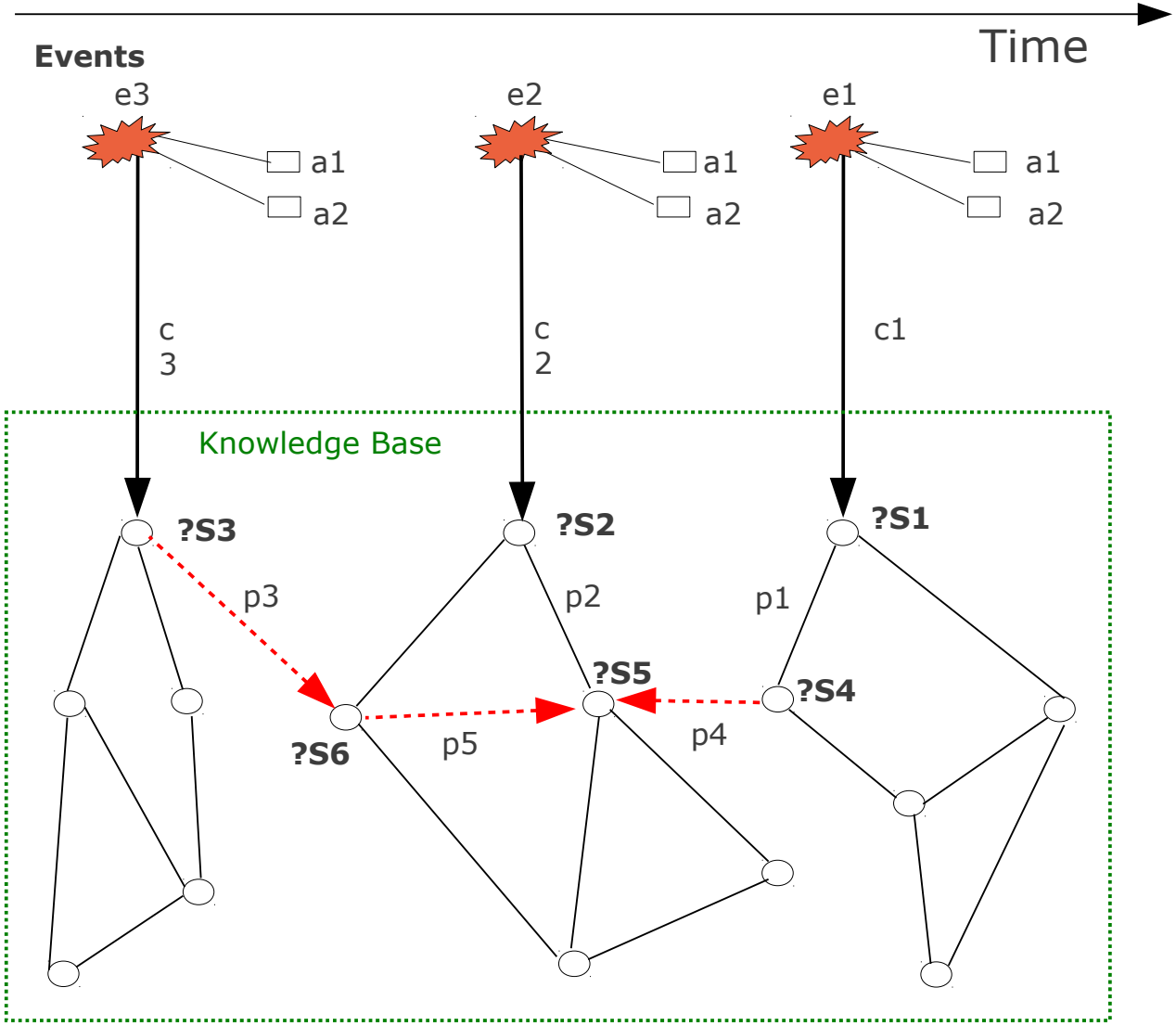


Event Detection Pattern

1. **Event Algebra Operation**
   - Sequence, Disjunction, Conjunction, Simultaneous, Negation, etc.
2. **SPARQL Query**
   - Operations to detect events based on their semantics (related meaning in background knowledge)
3. **Stream Windowing Operation**
   - Operations to slide event stream

{ [SPARQL Query] ,
[Event Algebra Operation],
[SPARQL Query],
[Event Algebra Operation],
[SPARQL Query], ...
} **[Sliding Window Operation]**

Example of Complex Event Pattern



Query

```
{
{ (?e1, c1, ?s1) .
  (?s1, p*, ?s) . }

[?e1 SEQ ?e2]
[Within 2 min.]

{ (?e2, c2, ?s2) .
  (?s2, p*, ?s) . } }

[?e2 SEQ e3]
[Within 5 min.]

{ (?e3, c3, ?s3) .
  (?s3, p*, ?s) . }
}
```

Event Query Rule Categories

- Categorization are based on the following factors:
  1) **Number of SPARQL queries** on KB in each event processing step
  2) Whether the SPARQL query **depends on incoming event data** and is generated based on their attributes
  3) **Number of event attributes** used for generating SPARQL queries
  4) **Number of events** used to generate SAPRQL queries (Events in a Window of Stream or Single Event)

Event Query Rule Categories

- Classification of most relevant and interesting event query rules based on embedding form of SPARQL predicates inside event query rule.
  - **Category – A:** Single SPARQL query inside the rule
  - **Category – B:** Several SPARQL queries are embedded in an event query rule and combined with event algebra operations.
- Not a complete categorization of all possible rules.

Pseudocode Example of Category A1

```
stream(CEvents) :-
        SResults = sparql_select(KB_id, SQuery),
        eProcessing(SResults, EStream, EQuery).
```

```
CEvents   =  Detected Complex Events
EStream   =  Raw event stream
EQuery    =  Event pattern query
SResult   =  Result set of the SPARQL query
SQuery    =  SPARQL query part of event sQuery rule
KB_id     =  ID of the target KB
sparq_select  =  Rule predicate used for querying the external KB
```

Pseudocode Example of Category A2

```
1     stream(CEvents) : -

2     ETuple = getSingelEvent(EStream, Udef),

3     SQuery = generateSPARQL(UQuery, ETuple),

4     SResults = sparql_select(KB_id , SPQuery),

5     eProcessing(SResults, EStream , EQuery).
```

```
CEvents   =  Detected Complex Events
EStream   =  Raw event stream
EQuery    =  Event pattern query
SResult   =  Result set of the SPARQL query
SQuery    =  SPARQL query part of event sQuery rule
KB_id     =  ID of the target KB
UDef      = Event type tuples defined by users
ETuple    = Event instance tuples defined by UDef
sparq_select   =  Rule predicate used for querying the external KB
```

Pseudocode Example of Category B1

```
1       stream(CEvents) : –
2         ETuples1 = getEvents(EStream, UDef1),
3         ETuples2 = getEvents(EStream, UDef2),

4         SQuery1 = generateSPARQL(UQuery, ETuples1),
5         SQuery2 = generateSPARQL(UQuery, ETuples2),

6          SResults1 = sparql_select(KB_id, SQuery1),
7          SResults2 = sparql_select(KB_id, SQuery2),

8    eProcessing(SResults1, SResults2, EStream, EQuery).
```

```
CEvents   =  Detected Complex Events
EStream   =  Raw event stream
EQuery    =  Event pattern query
SResult   =  Result set of the SPARQL query
SQuery    =  SPARQL query part of event sQuery rule
KB_id     =  ID of the target KB
UDef      = Event type tuples defined by users
ETuple    = Event instance tuples defined by UDef
sparq_select   =  Rule predicate used for querying the external KB
```

Pseudocode Example of Category B2

```
1       stream(cEvents):-
2         ETuples1 = getEvents(EStream, UDef1),
3         SQuery1 = generateSPARQL(UQuery,ETuples1),
4         SResults1 = sparql_select(KB_id,SQuery1),

% Wait until CEvents1 is happened!
5       CEvents1= eProcessing(SResults1, EStream, EQuery),

6       ETuples2 = getEvents(EStream, UDef2),
7       SQuery2 = generateSPARQL(UQuery,  ETuples2,  8CEvents1),
8       SResults2 = sparql_select(KB_id, SQuery2),

9       eProcessing(SResults2, CEvents2, EStream,  EQuery).
```

```
Cevents   =  Detected Complex Events
EStream   =  Raw event stream
EQuery    =  Event pattern query
SResult   =  Result set of the SPARQL query
SQuery    =  SPARQL query part of event sQuery rule
KB_id     =  ID of the target KB
UDef      = Event type tuples defined by users
ETuple    = Event instance tuples defined by UDef
sparq_select  =  Rule predicate used for querying the external KB
```

Example: Implementation in Prova rule language (http://prova.ws)

Freie Universität Berlin

```
:- eval(server()).

server() :-
  sparqlrule(QueryID),
  rcvMult(XID,Protocol,Sender,event, {url->URL}) [testrule(QueryID, URL)],
  sendMsg(XID, Protocol, Sender, testrule, {url->URL}).

testrule(QueryID, URL) :-
  sparql_results(QueryID, URL, CompanyEmployees),
  CompanyEmployees > 50000.


sparqlrule(QueryID) :-
    Query = '
        PREFIX DBPPROP: <http://dbpedia.org/property/>
        PREFIX DBPEDIA: <http://dbpedia.org/resource/>

        SELECT ?company ?employees WHERE {
            ?company DBPPROP:industry DBPEDIA:Computer_software .
            ?company DBPPROP:numEmployees ?employees .
            ?company DBPPROP:industry DBPEDIA:Retail . }',

    sparql_select(Query, QueryID, [], 'http://dbpedia.org/sparql').
```

Experiments

Experimental Performance Results on Different Query Rule Categories

Installation on two machines, both Quad Core Intel(R) Xeon(R) CPU E31245 @ 3.30GHz with 16 GB RAM Dedicated network

DBPedia 3.7
Complete Mirror
288 Million RDF triples

Virtuoso Triple Store

Category of sQuery Rules	Throughput (Events/s)
A1 (Caching)	280000
A2	2200
A3	1300
B1 , B2, B3	500-4000

Conclusion & Outlook

- **Semantic Enrichment of Events**
- **Different categories** of event query rules for data fusion from external KBs.

Future Work

- Algorithms for efficient processing of events based on background knowledge
  - Enrichment of events
  - Preprocessing
  - Planning

Thank you!

http://www.corporate-semantic-web.de

AG Corporate Semantic Web
Freie Universität Berlin
http://www.inf.fu-berlin.de/groups/ag-csw/