Plan-Based Semantic Enrichment of Event Streams

Kia Teymourian, Adrian Paschke

Corporate Semantic Web Research Group
Freie Universität Berlin

Extended Semantic Web Conference 2014
Heraklion, Crete, Greece
25- 29 May, 2014

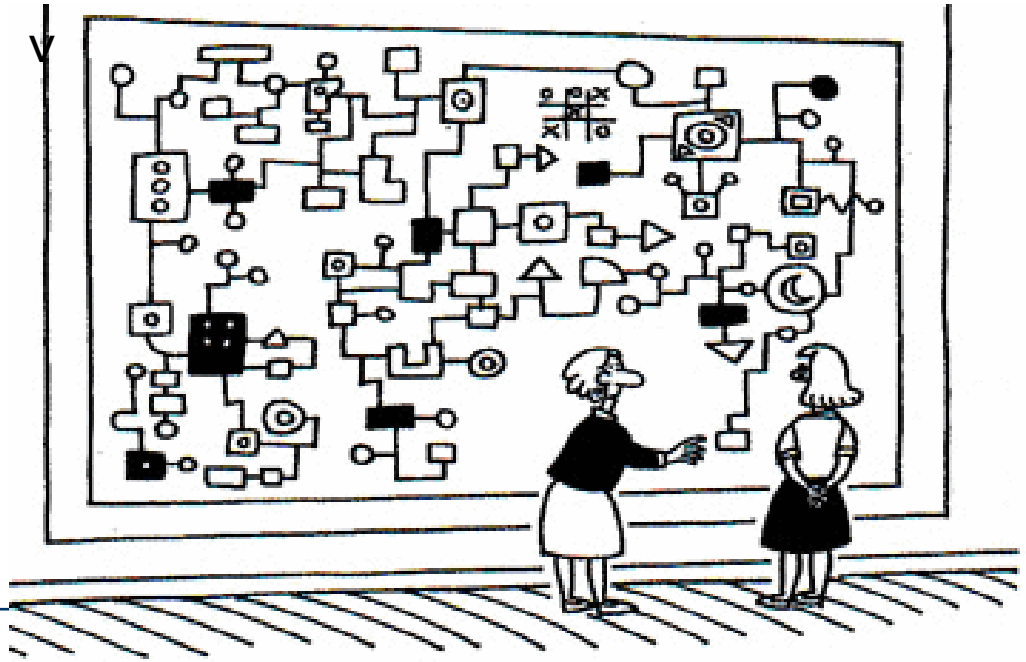
Freie Universität Berlin

Outline

- Knowledge-Based Complex Event Processing
- Event Stream Enrichment
- Planning of Event Enrichment

Motivation

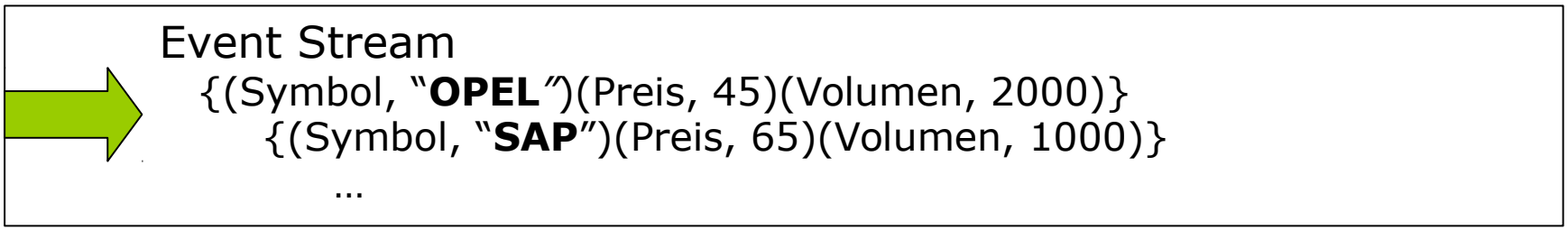
- Huge amounts of **streaming data** and **Background knowledge** about the domain
- **Higher-level queries** based on background knowledge to **detect complex events**
- *Improving the **expressiveness**, **agility** and **flexibility***



Event Stream

- **Event Stream:**
  - Stream of Event Instances
  - e.g. Simple Attribute/Values Tuples

- **Example:**
  - Stock market event stream

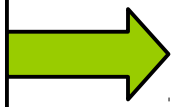


- Not an RDF Event Stream

Example: Semantic Event Processing

Query:

Select stocks of companies,
who have in **Europe production facilities** *and*
produce products **from Metal** *and*
more than **10,000 employees** *and*
are at the moment in **reconstruction phase** *and*
their price **increased** in the past 5 minutes.



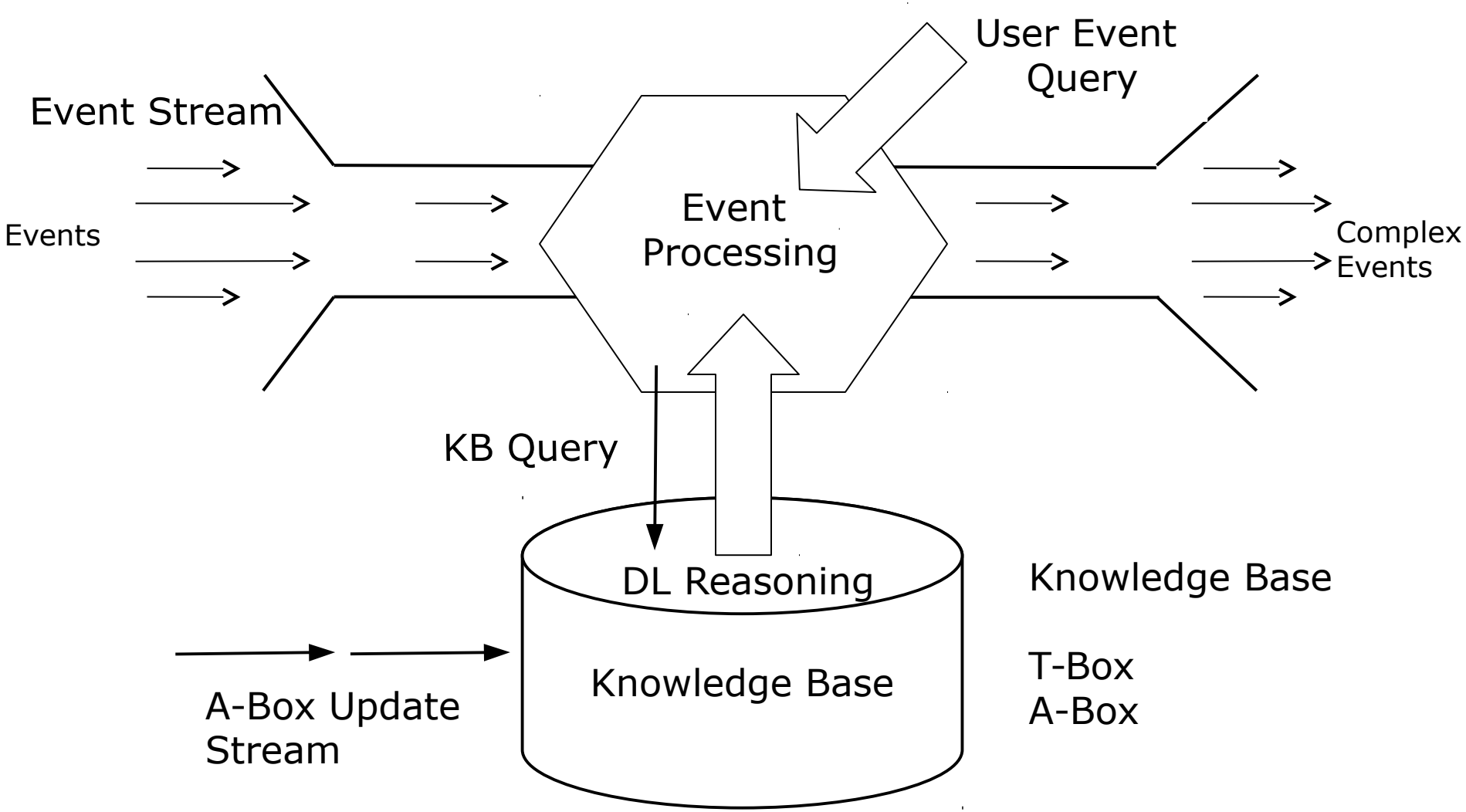
Event Stream

{(Symbol, "**OPEL**")(Preis, 45)(Volumen, 2000)}
{(Symbol, "**SAP**")(Preis, 65)(Volumen, 1000)}

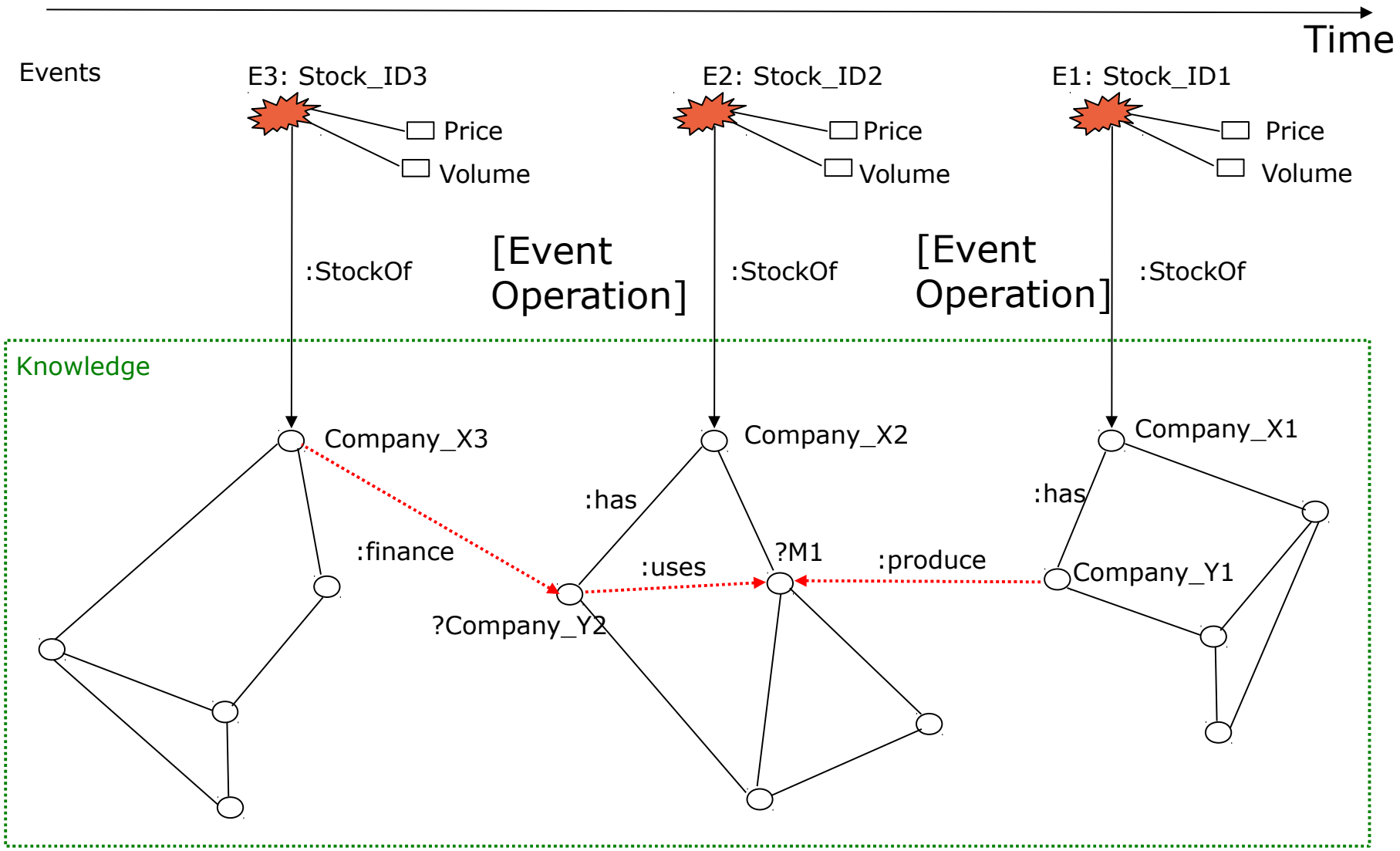
Knowledge Base

{(***OPEL***, is_a, **automobil_company**),
(**automobil_company**, build, **Cars**),
(**Cars**, are_build_from, **Metal**),
(***OPEL***, hat_production_facilities_in, **Germany**),
(**Germany,** is_in, **Europe**)
(***OPEL***, is_a, **Major_corporation**),
(**Major_corporation**, have, over**_10,000_employees**),
(***OPEL***, is_in, **reconstruction_phase**)}

Knowledge-based Event Processing



Example of Complex Event Pattern

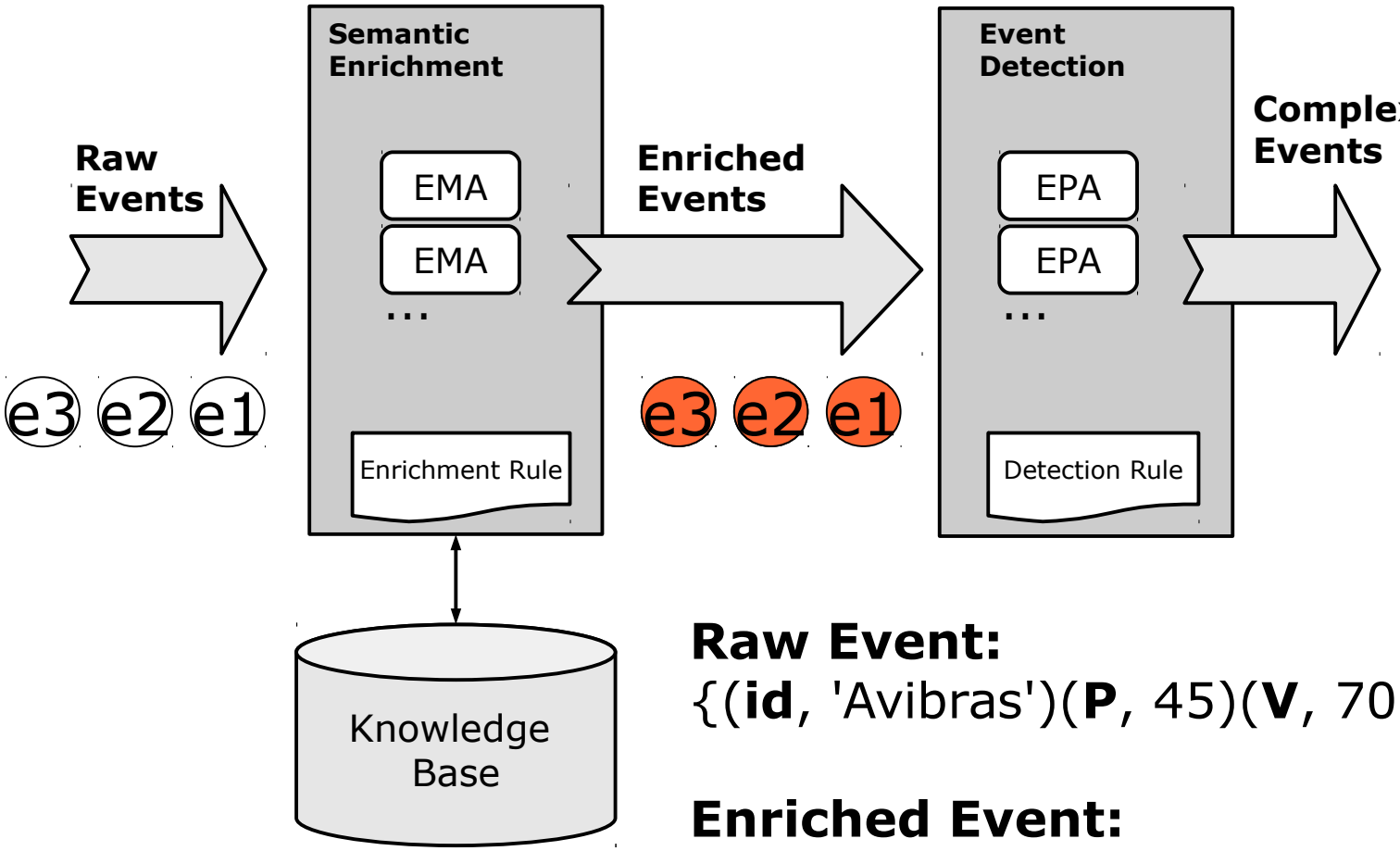


Example: Event Pattern

```
[ { ?e1            p:stockOf      ?Company_X1  .
  ?Company_X1  p:affiliates    ?Company_Y1 .
  ?Company_Y1  p:supplierOf    :Metal }

{ ?e1  SEQ  ?e2  }          %(Event Operation)

{ ?e2              p:stockOf   ?Company_X2   .
  ?Company_X2      p:affiliates ?Company_Y2  .
  ?Company_Y2  p:hasDemand  :Metal  .  }

{ ?e2  SEQ  ?e3 }           %(Event Operation)

{  ?e3               p:stockOf   ?Company_X3   .
   ?Company_X3       p:finance   ?Company_Y2 .
 } ]  [ within 2 minutes ]
```

Semantic Enrichment of Event Streams

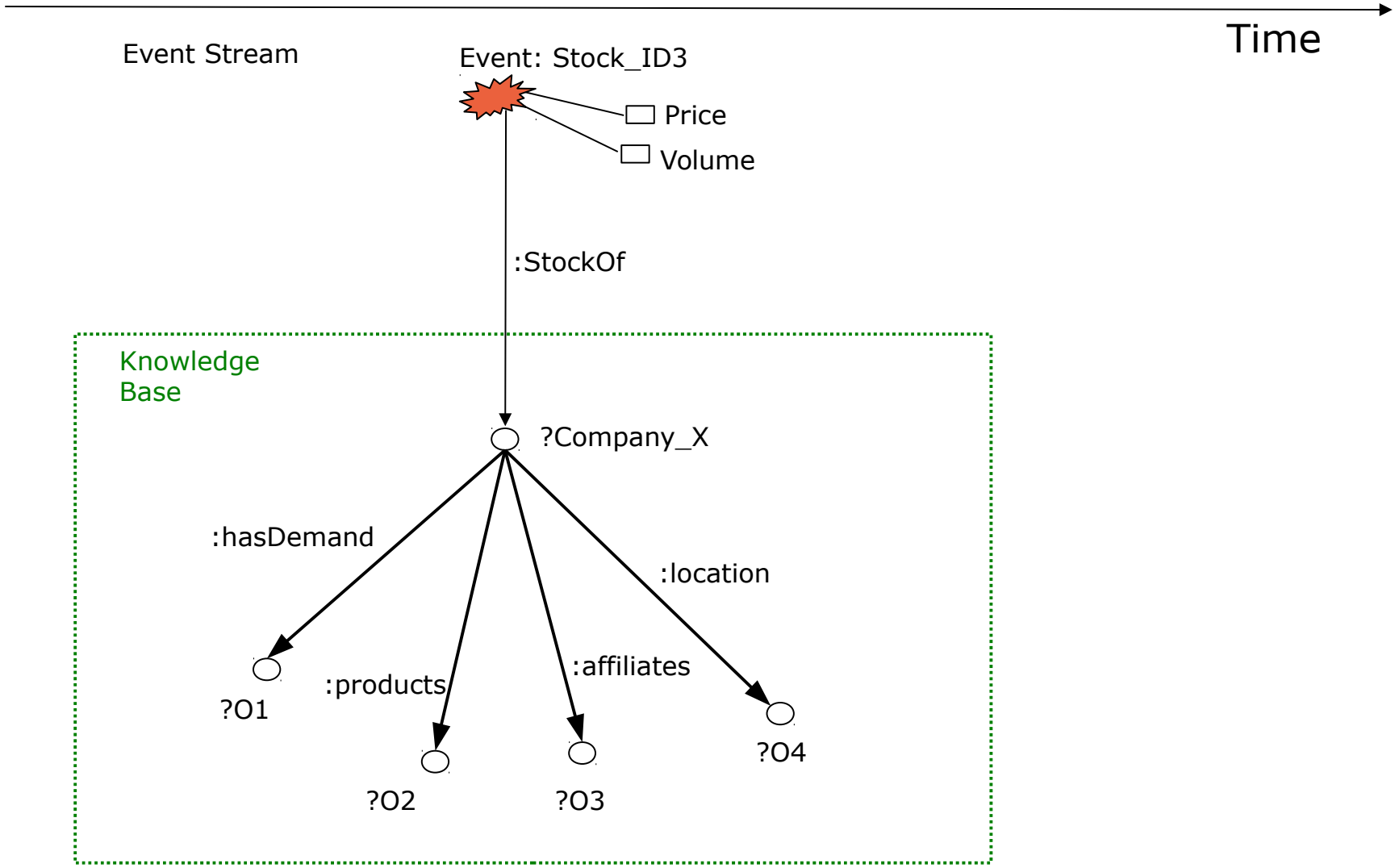
Semantic Enrichment of Event Stream



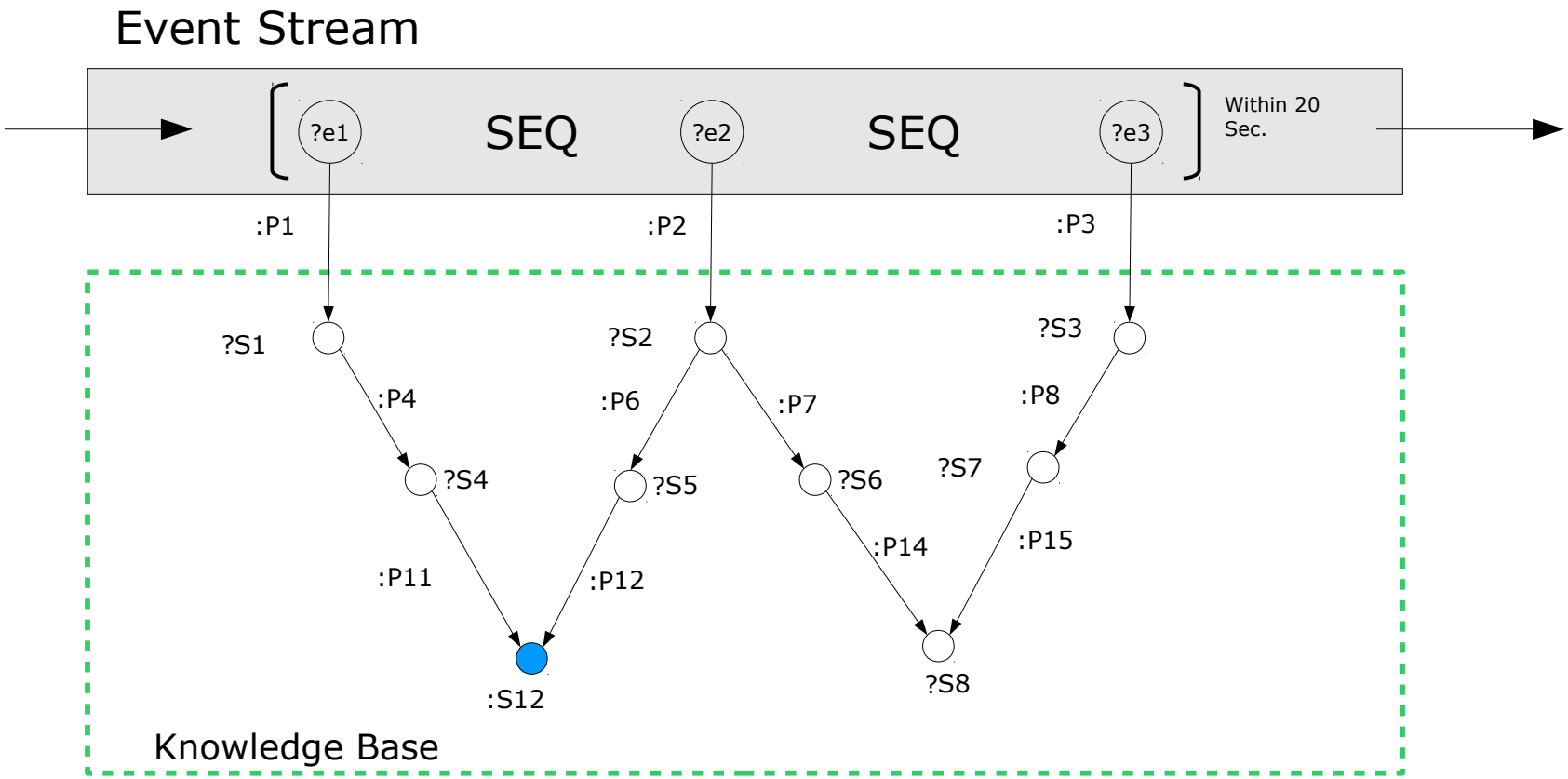
Raw Event:
{(**id**, 'Avibras')(**P**, 45)(**V**, 700)}

Enriched Event:
{(**id**, 'Avibras')(**P**, 45)(**V**, 700)
(**produce**,'Aircraft;Artillery;Electronics;
Explosive_material;')}

Example of Simple Star-Shape Pattern



Example of SEQUENCE Event Pattern



?e1	:P1	?S1
?S1	:P4	?S4
?S4	:P11	**:S12**

SEQ

?e2	:P2	?S2
?S2	:P6	?S5
?S5	:P12	**:S12**
?S2	:P7	?S6
?S6	:P14	**?S8**

SEQ

?e3	:P3	?S3
?S3	:P8	?S7
?S7	:P15	**?S8**

Processing Costs

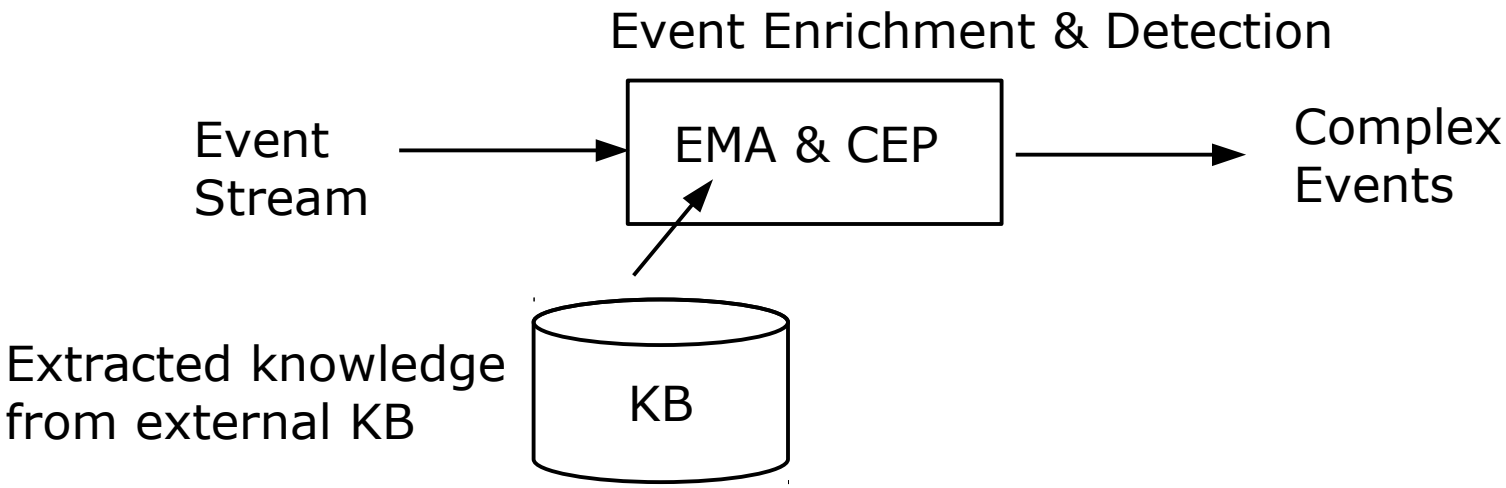
1) Cost of Event Acquisition
- Event Transmission Costs

2) Cost of Background Knowledge Acquisition
- Materialization (joins, ...)
- Reasoing Costs on external KB (highly complex)
- **Result transmission costs** (No. of returned results)

3) Computation Costs for Detection of Events
- Size of Knowledge in Memory
- Nr. of Rules

Planning of Event Enrichment

Single Step Processing

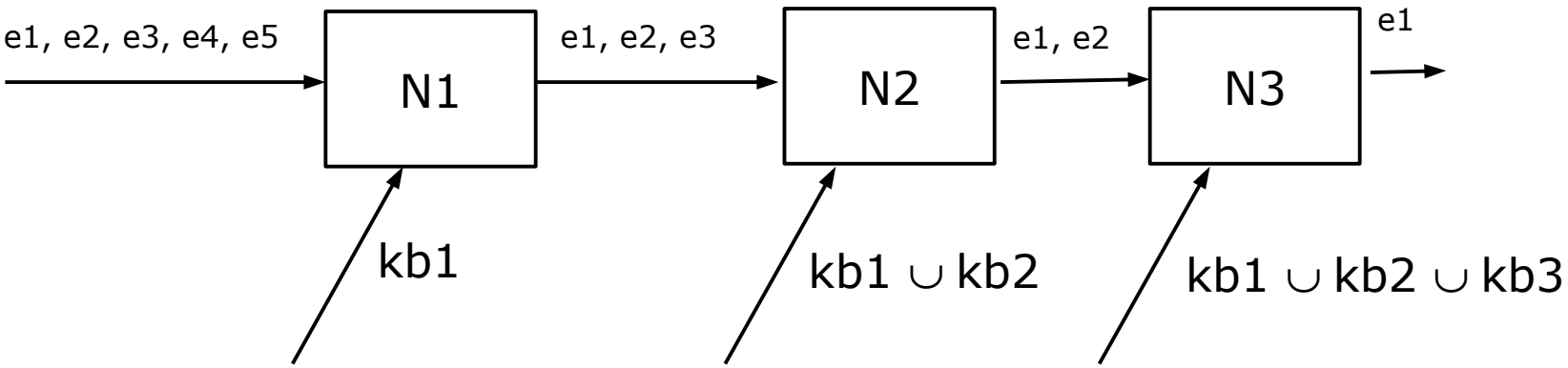


- Events are enriched in one single node
- Enrichment of all required attributes
- Complex events can be detected based on the enriched knowledge

- **Problem: large amounts of unnecessary enrichments**

Multi-Step Event Processing

Akdere et al. and Schultz-Møller et al. investigated the plan-based complex event detection across distributed sources.



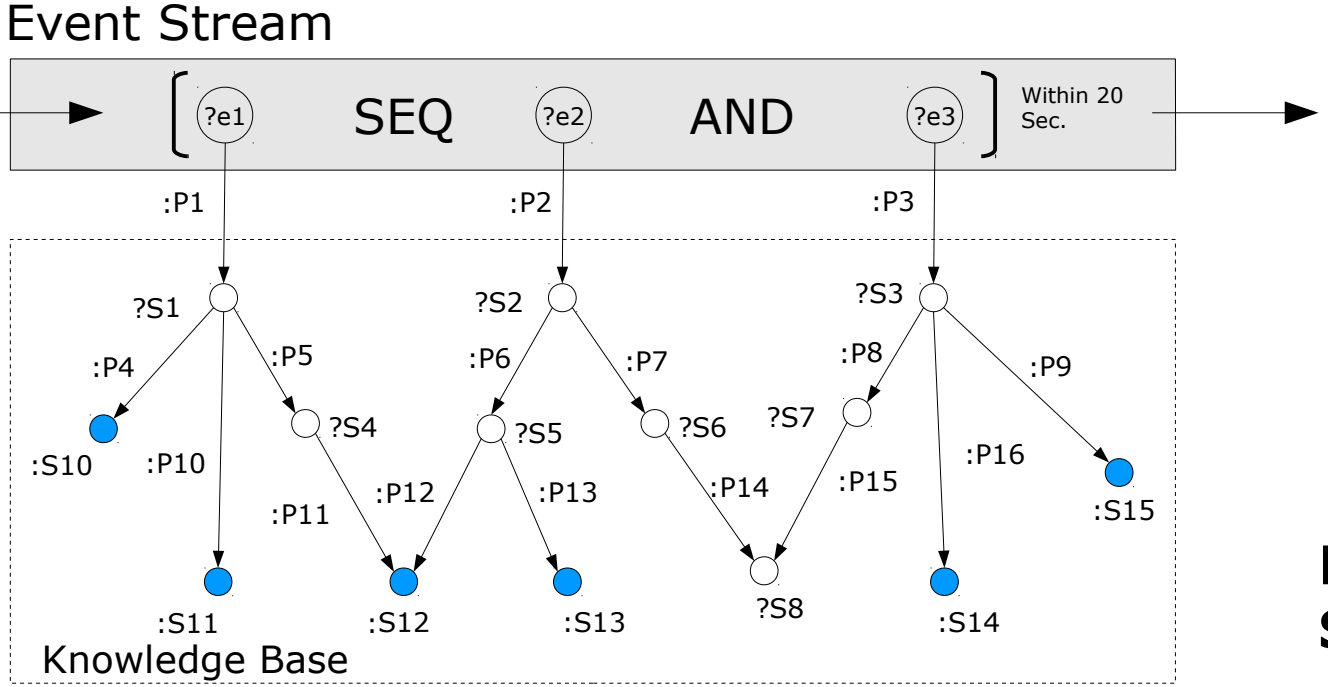
Steps:

e1() raw event

1. e1(a1) enrich with attribute a1 and check if a1 is matched
2. e1(a1,a2) enrich with attribute a2 and check if a2 is matched
3. e1(a1,a2,a3) enrich with attribute a3 and check if a3 is matched

Problem: finding the optimized plan while meeting the latency and the cost considerations

Subgraph Generation



Extracted Subgraphs

1. (**?S1** :P4 :S10)
2. (**?S1** :P10 :S11)
3. (**?S1** :P5 ?S4) (?S4 :P11 **:S12**)

4. (**?S2** :P6 ?S5) (?S5 :P12 **:S12**)
5. (**?S2** :P6 ?S5) (?S5 :P13 :S13)
6. (**?S2** :P7 ?S6) (?S6 :P14 **?S8**)

7. (**?S3** :P8 ?S7) (?S7 :P15 **?S8**)
8. (**?S3** :P16 :S14)
9. (**?S3** :P9 :S15)

- Extract subgraphs from the original query graph

For each subgraph:
- Start from the root
- Traverse the tree to the leafs

Planning of Event Enrichment in Multi-Steps

- The **search space** for the optimal plan is very large
  - Number of events in Query
  - Number of event attributes
  - Number of processing steps

- Start with an estimated plan based on:
  - **Filter Functionality Estimation of Subgraphs**
    - How good can filter the events using the ***subgraph G***?
  - **Estimated Matching Probability Factor of the Predicate**
    - How good can filter using the ***predicate p*** in subgraph G?

Evaluation

Evaluation

- **Experiments**:
  - Single Step Processing vs. Multi-Steps
  - Different Plans

- **Two hosts:**
  - Event Enrichment (Rules , OpenRDF, SPARQL, Virtuoso)
  - Event Detection (Esper Engine, Storm)

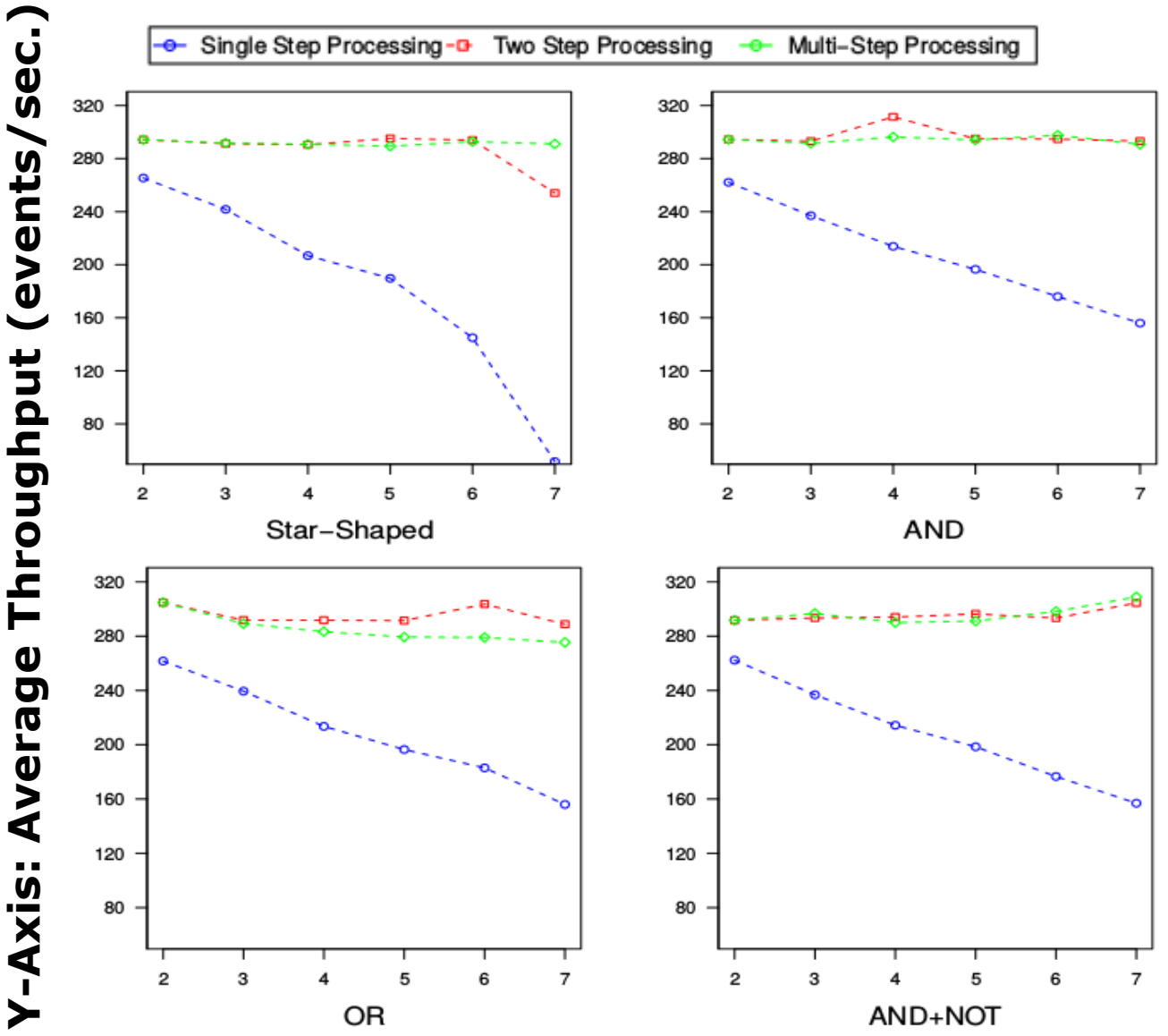
- **Event stream dataset:**
  - Stock Market Events, s&p 500, randomly generated for the stress tests.
  - Uniform probability
- **Background knowledge dataset:**
  - Mirror of DBpedia as KB

Evaluation

- **Queries**:
  - Star-Shaped
  - With 4 Different Event Operators
    - SEQ, AND, OR, NOT
  - **Example Queries**:
  - { E_P1 **SEQ** E_P2 }
  - { E_P1 **SEQ** E_P2 **SEQ** E_P3}
  - …

No.	Predicate
1	dpedia-owl:location
2	dpedia-owl:industry
3	dpedia-owl:numberOfEmployees
4	dbpprop:products
5	dpedia-owl:subsidiary
6	rdf:type
7	dcterms:subject

Performance Comparison of Multi-Step Processing



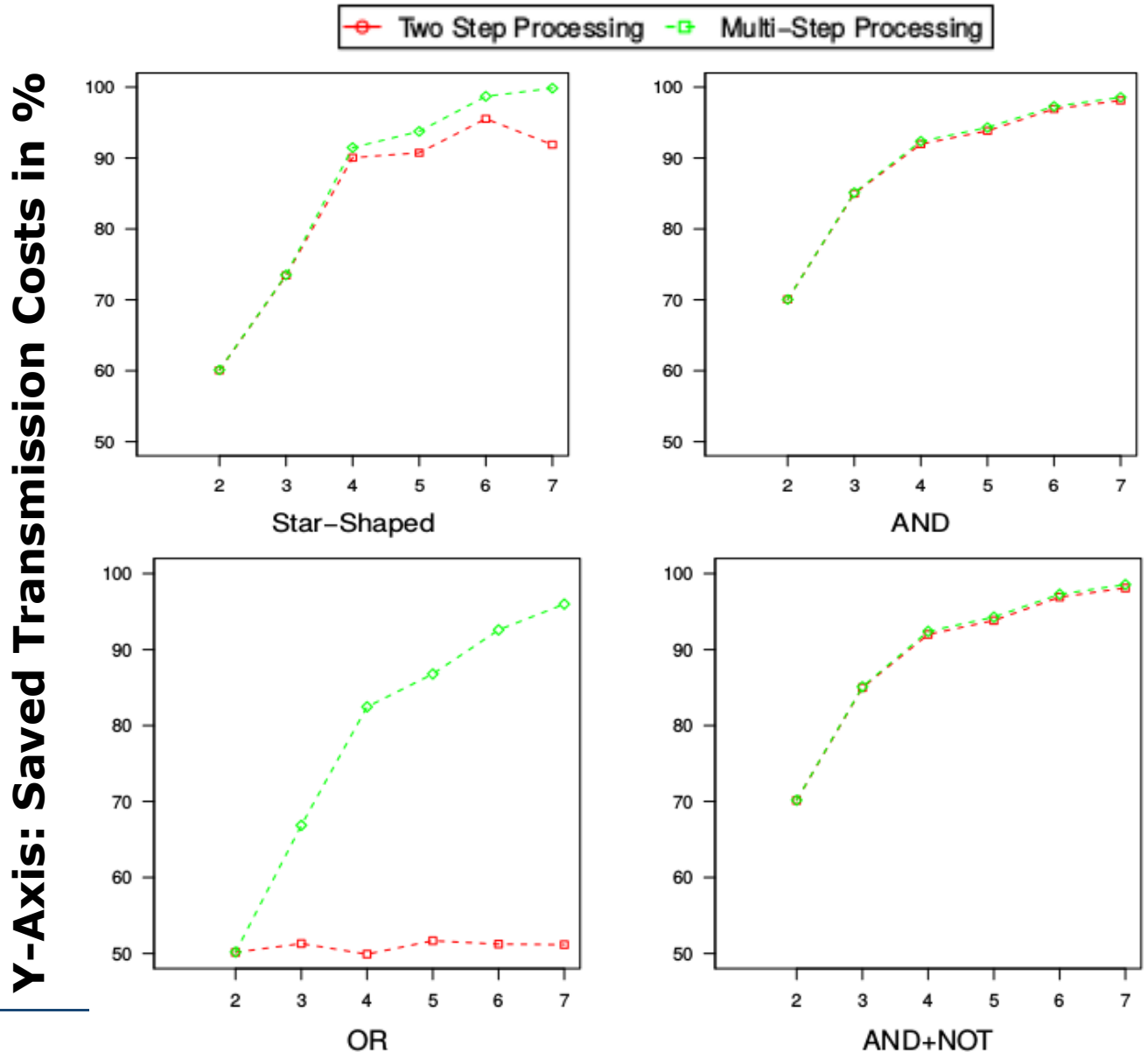
Query with 7 BGPs

In Multi-Step: 1 BGP in 1 Step

X–Axis: Number of BGPs in Event Detection Query

Number of Detection Steps

Saved Transmission Costs in Comparison to Single-Step Processing



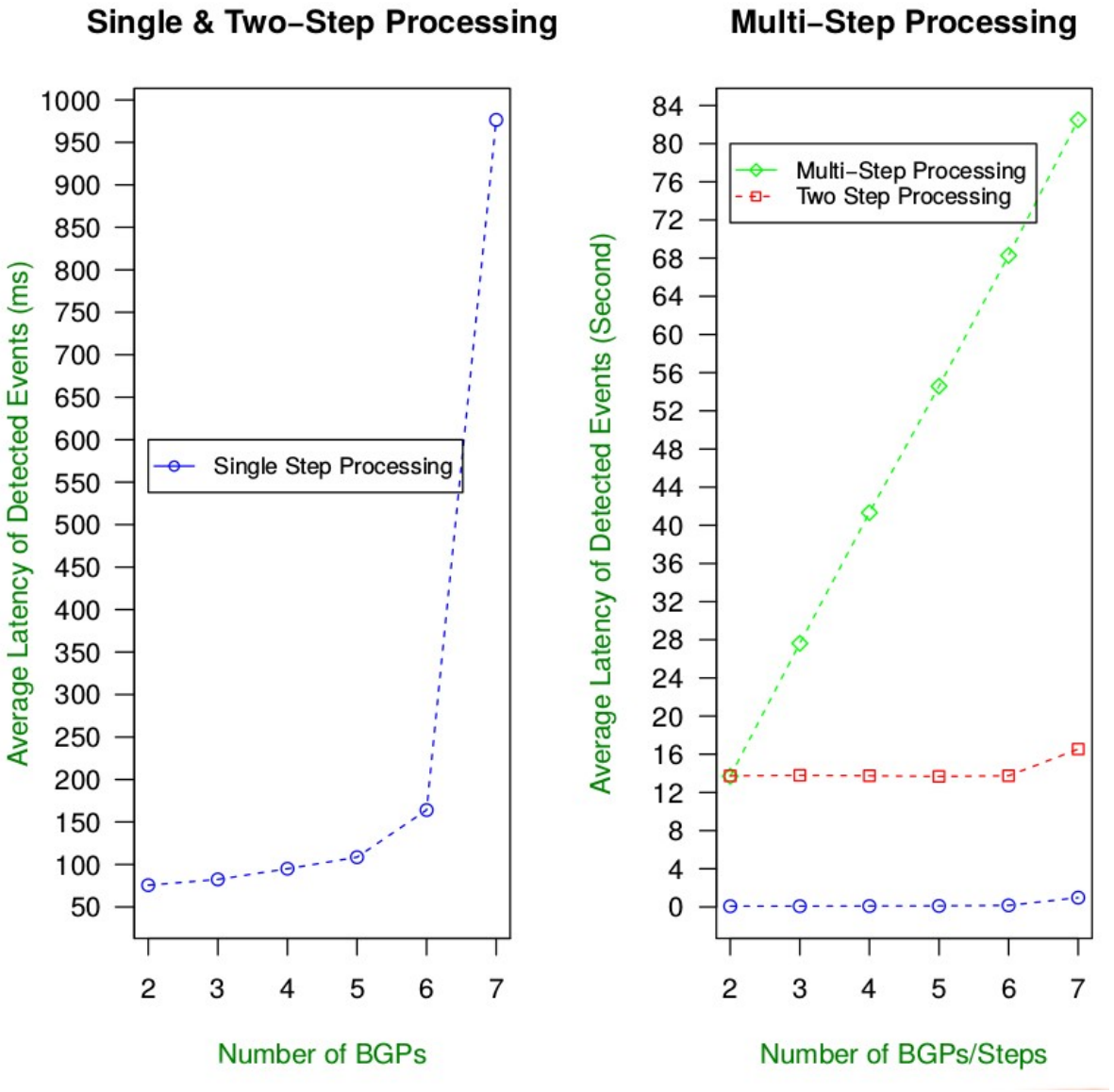
Query with 7 BGPs

In Multi-Step: 1 BGP in 1 Step

X−Axis: Number of BGPs in Event Detection Query

Number of Detection Steps

Average Latency



Average Latency of Detected Complex Events in Multi-Step Processing (Star-Shaped Query)

Planning (AND Operator)

Enrichment & Detection Plan-1

Steps	Enrichment Predicates	Matching Predicates
Step-1	5,3	(5 OR 3)
Step-2	4,2	(4 OR 2)
Step-3	1,6	(1 OR 6)
Step-4 (final)	7	((5,4,1,6) AND (3,2,6,7))

Four Enrichment Plans

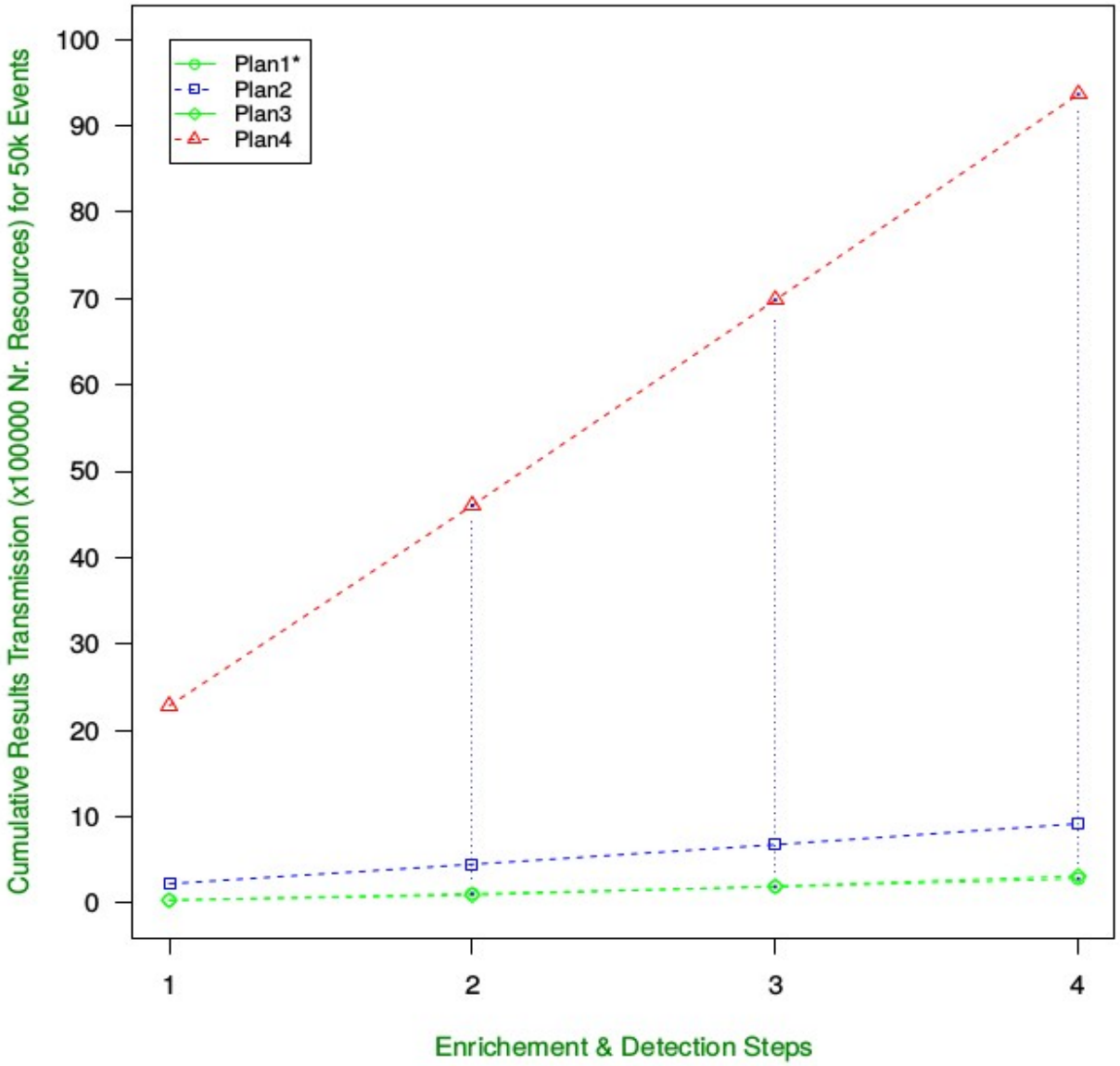
Plans	Step-1	Step-2	Step-3	Step-4(final)
Plan-1	5,4	3,2	1,6	7
Plan-2*	5,3	4,2	1,6	7
Plan-3	7,6	1,2	4,3	5
Plan-4	1,2	3,4	5,6	7

Query:
{ E_(5,4,1,6) AND E_(3,2,6,7)}

Predicate Numbers

No.	Predicate
1	dpedia-owl:location
2	dpedia-owl:industry
3	dpedia-owl:numberOfEmployees
4	dbpprop:products
5	dpedia-owl:subsidiary
6	rdf:type
7	dcterms:subject

Costs of Different Plans



Cumulative Costs for different plans (Queries with AND Operators)

50k events sent to the system and measured the transmission costs

Conclusion & Future Work

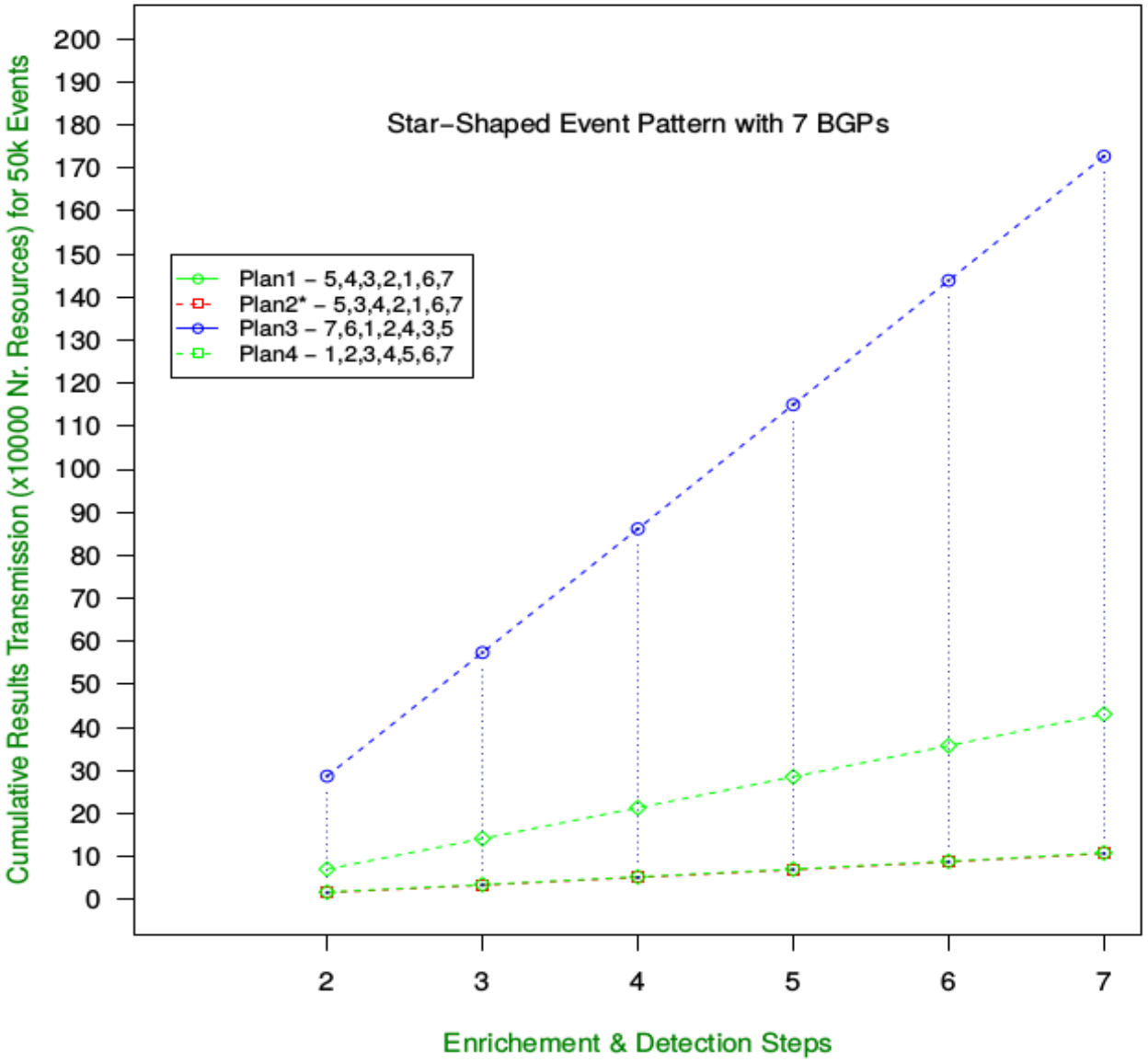
- Multi-Step Processing can **save enrichment costs** but effects the latency
- The event enrichment and detection in multi-steps can be **planned**.
- **Heuristics** about the KB can help to have an **approximated start plan** to search for the optimal plan.
- **Future Work**
  - Usage of the **intermediate joins** of event detection graphs for the planning
  - Investigate the effect of **stream entropy** on the planning

Thank You!

Questions?

Acknowledgements
This work has been partially supported by the **"InnoProfile-Transfer Corporate Smart Content"** project funded by the German Federal Ministry of Education and Research (**BMBF**) and the **BMBF Innovation Initiative** for the New German Länder "Entrepreneurial Regions".

Costs of Different Plans



Cumulative Costs for different plans

Star-Shaped Pattern

50k events sent to the system and measured the transmission costs

Event Enrichment

- Each event instance is enriched with additional attributes.
  - Fat events are generated

Raw Event: {(**id**, 'Avibras')(**P**, 45)(**V**, 700)}

- Enriched with **Products**

  {(**id**, 'Avibras')(**P**, 45)(**V**, 700)(**produce**,'Aircraft;Artillery;Electronics;Explosive_material;')}

- Enriched with **Industry Sector**

  {(**id**, 'Avibras')(**P**, 45)(**V**, 700)(**produce**,'Aircraft;Artillery;Electronics;Explosive_material;')(**industry**,'Electronics;Arms_industry')}

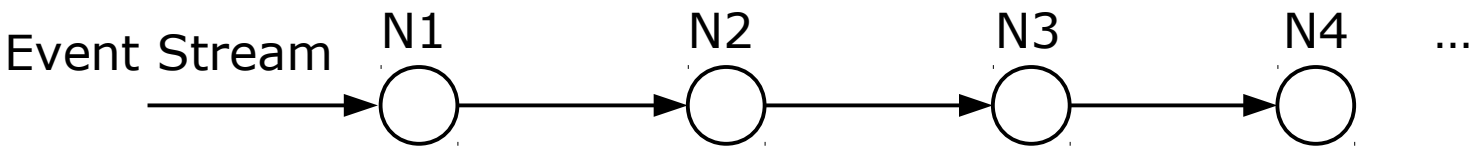
- Enriched with **Location of the company**

  {(**id**, 'Avibras')(**P**, 45)(**V**, 700)(**produce**,'Aircraft;Artillery;Electronics;Explosive_material;')(**industry**,'Electronics;Arms_industry')(**location**, 'USA;Brazil')}

- Enriched with **Location of the company**

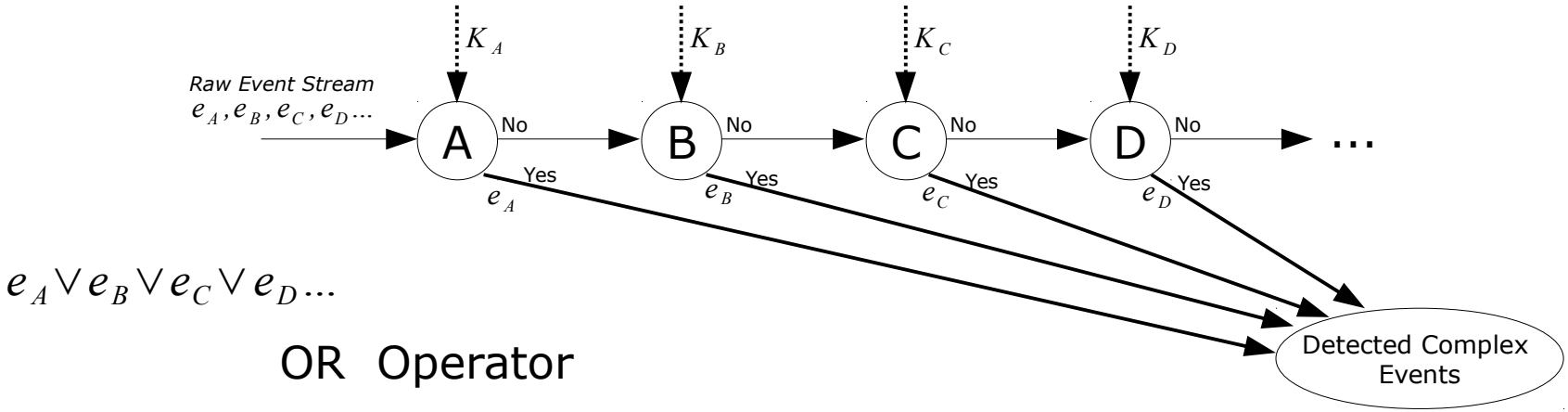
  {(**id**, 'Avibras')(**P**, 45)(**V**, 700)(**produce**,'Aircraft;Artillery;Electronics;Explosive_material;')(**industry**,'Electronics;Arms_industry')(**location**, 'USA;Brazil')(**numberOfEmployees**, 600)}

Stepwise Event Processing



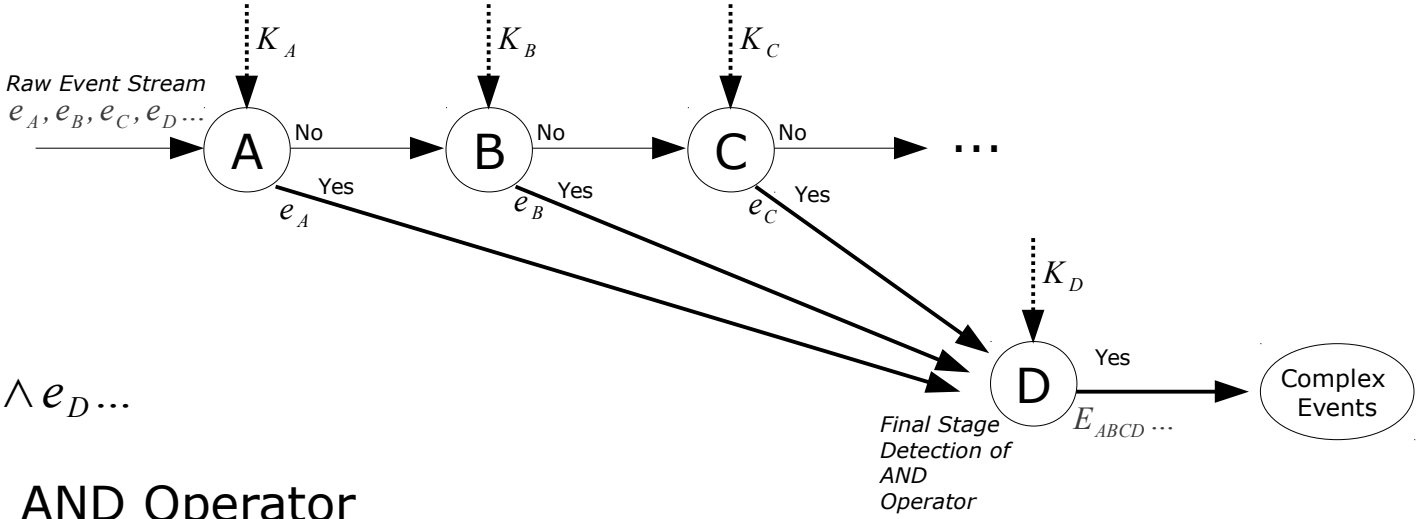
- **Each Processing Node:**
  - Enrich events
  - Detect complex event based on the enriched knowledge
- **Queries should be preprocessed**
- **Processing plan** should be prepared for each event query
- Transmission Costs:
  - Event Transmission
  - Knowledge Extraction

Event Operators & Stepwise Event Matching



$e_A \vee e_B \vee e_C \vee e_D \ldots$

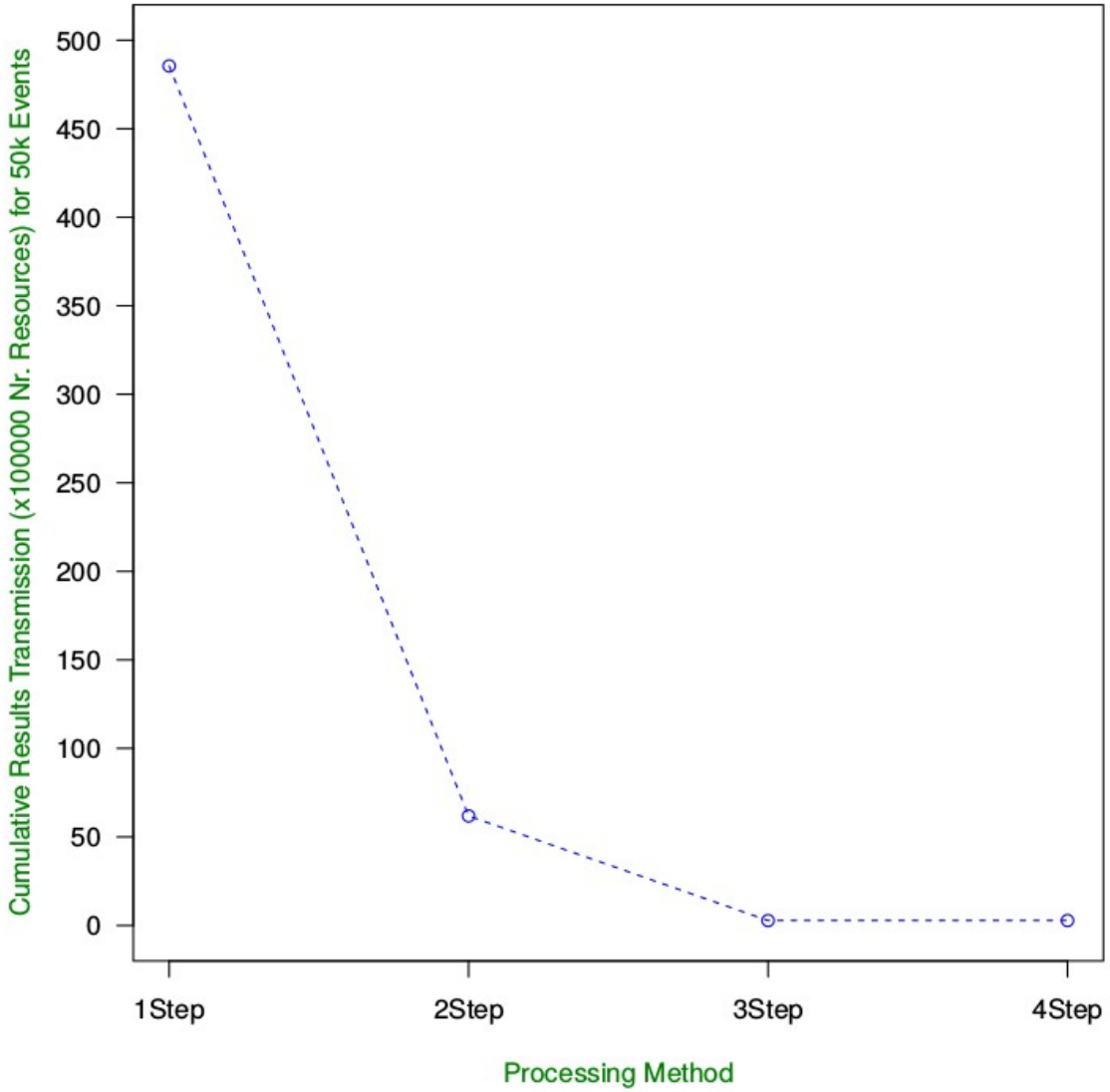
OR Operator



$e_A \wedge e_B \wedge e_C \wedge e_D \ldots$

AND Operator

Cumulative Transmission Cost



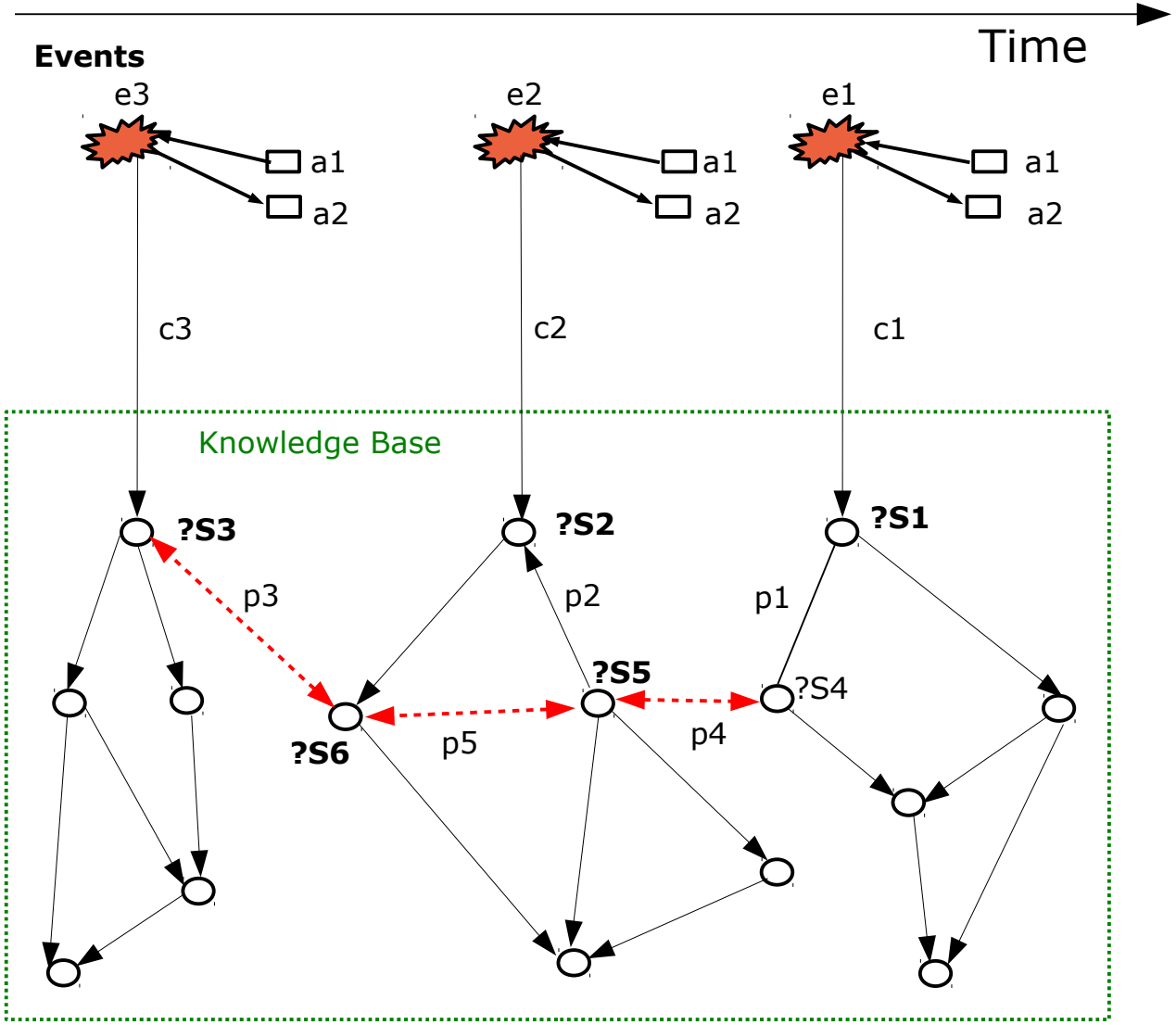
Cumulative Transmission Cost for Different Steps using **AND** Operation

Motivation for Semantic CEP

- **Today's business environment:**
  - High *complex processes, events, …*
  - Existing domain background knowledge
  - Missing systems for detection of events based on available background knowledge.

- **Many event processing use cases require:**
  - High expressiveness
  - High flexibility
  - Simplicity

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 Detection Pattern

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

{ [RDF Triple Pattern] ,
[Event Operation],
[RDF Triple Pattern],
[Event Operation],
[RDF Triple Pattern], ...
} **[Sliding Window Operation]**

Example: Prova event pattern - Caching

Freie Universität Berlin

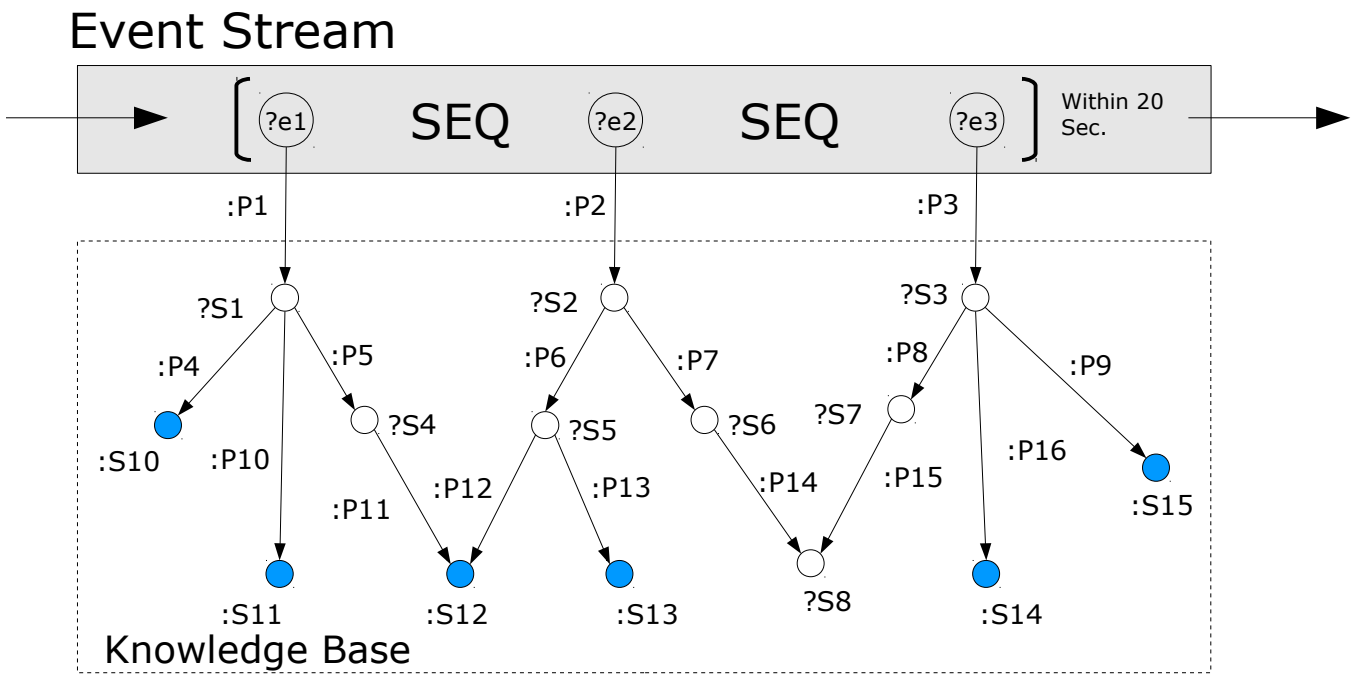
```
:- eval(server()).

server() :-
    sparqlrule(QueryID),
    rcvMult(XID, Protocol, Sender, event,
      {time->Time, url->URL, lastprice->Lastprice, volume->Volume, high->High, low->Low})
      [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').
```

Example: Prova event pattern - Polling

Freie Universität Berlin

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

server() :-
   rcvMult(XID, Protocol, Sender, event, {url->URL, lastprice->Lastprice, volume->Volume}),

  testrule(URL, Lastprice),

  sendMsg(XID, Protocol, Sender, testrule, {url->URL, lastprice->Lastprice}).

testrule(URL, Lastprice) :-
    Query = '
        PREFIX DBPPROP: <http://dbpedia.org/property/>
        PREFIX DBPEDIA: <http://dbpedia.org/resource/>
        PREFIX dbpedia-owl: <http://dbpedia.org/ontology/>

        SELECT ?employees WHERE {
            <$url> DBPPROP:industry DBPEDIA:Computer_software .
            <$url> dbpedia-owl:numberOfEmployees ?employees .
            <$url> DBPPROP:industry DBPEDIA:Retail .
        } ',
    sparql_select(Query, QueryID, [url(URL)], 'http://dbpedia.org/sparql'),
    sparql_results(QueryID, CompanyEmployees),
    CompanyEmployees > 10000,
    Lastprice > 80,
    println(["Complex event: ", URL, " ", Lastprice]).
```

Example of SEQUENCE Event Pattern



?e1	:P1	?S1
?S1	:P4	?S10
?S1	:P10	:S11
?S1	:P5	?S4
?S4	:P11	**:S12**

SEQ

?e2	:P2	?S2
?S2	:P6	?S5
?S5	:P12	**:S12**
?S5	:P13	:S13
?S2	:P7	?S6
?S6	:P14	**?S8**

SEQ

?e3	:P3	?S3
?S3	:P8	?S7
?S7	:P15	**?S8**
?S3	:P16	:S14
?S4	:P9	:S15