Retractable Complex Event Processing and Stream Reasoning

RuleML 2011

Barcelona, Spain

Darko Anicic,
Sebastian Rudolph,
Paul Fodor,
Nenad Stojanovic

Agenda

➡ **Introduction, Motivation**

- ETALIS Language for Events
  - Syntax;
  - Semantics;
  - Experimental Results;
- Conclusion.

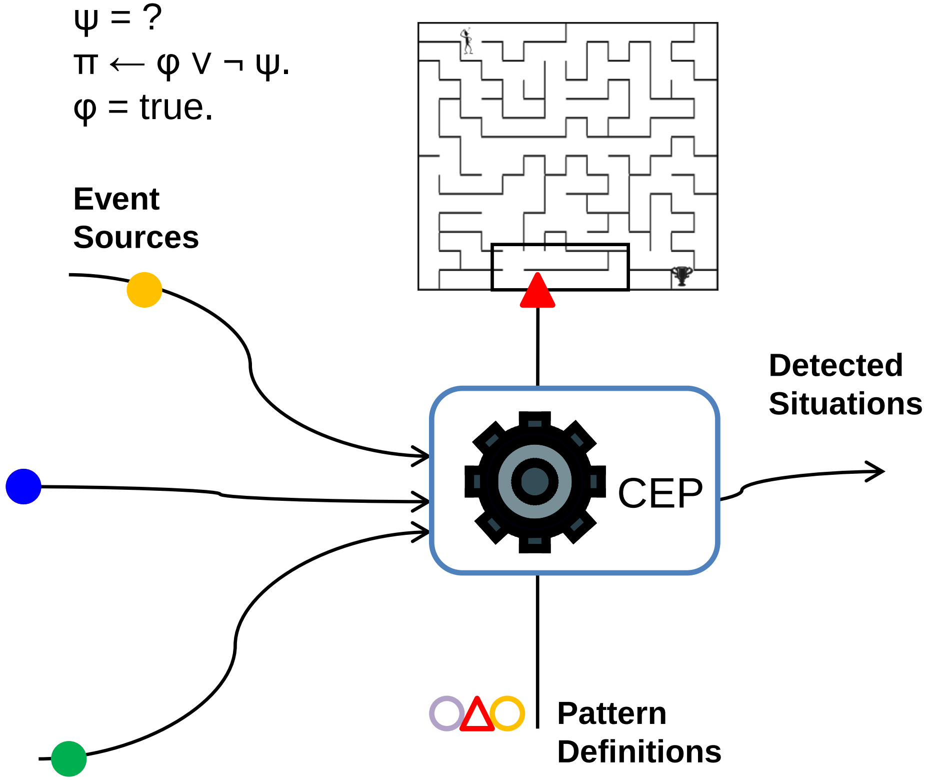
Logic-based Complex Event Processing in ETALIS

Use background (contextual) knowledge to explore **semantic relations** between events, and detect otherwise undetectable **complex situation.**

$\psi = ?$

$\pi \leftarrow \varphi \vee \neg \psi.$

$\varphi = \text{true}.$



CEP with on-the-fly knowledge evaluation and **stream reasoning**:

- Complex situation based on **explicit** data (events) and **implicit/ explicit** knowledge
- Classification and filtering
- Context evaluation
- Intelligent recommendation
- Predictive analysis

Knowledge-based CEP

Motivation

FZI

Knowledge-based CEP & Stream Reasoning

- Today's CEP systems are focused mostly on **throughput** and **timeliness**;
- Time critical actions/decisions are supposed to be taken upon detection of complex events;
- These actions additionaly require evaluation of background knowledge;
- Knowledge captures the domain of interest or context related to actions/decisions;
- The task of reasoning over streaming data (events) and the background knowledge constitutes a challenge known as **Stream Reasoning**.

Current CEP systems provide **on the-fly analysis** of data streams, but mainly fall short when it comes to combining streams with evolving **knowledge** and performing **reasoning** tasks.

FZI

Non-blocking Event Revision – Transactional Events

- Events in today's CEP systems are assumed to be **immutable** and therefore **always correct**;
- In some situations however revisions are required:
  - an event was reported by mistake, but did not happen in reality;
  - an event was triggered and later revoked due to a transaction failure.
- As recognised in [Ryvkina et al. ICDE'06], event stream sources may issue **revision tuples** that amend previously issued events.

Current CEP systems provide **on** the-fly analysis of data streams, but typically don't take these **revision** tuples into account and produce **correct** revision outputs.

Related Work

DSMS Approaches for retractions in CEP:

- D. Carney et al. Monitoring streams: a new class of data management applications. In VLDB’02
- A. S. Maskey et al. Replay-based approaches to revision processing in stream query engines. In SSPS’02.
  - based on archives of recent data and replying
  - whole recent history is kept archived
- R. S. Barga et al. Consistent streaming through time: A vision for event stream processing. In CIDR’07.
  - based on blocking, buffering and synchronisation point

Stream Reasoning approaches:

- D. F. Barbieri et al. An execution environment for C-SPARQL queries. In EDBT’10.
- A. Bolles et al. Streaming SPARQL – Extending SPARQL to Process Data Streams. In ESWC’10.

Agenda

FZI

- Introduction, Motivation
- **ETALIS: Retractable CEP and Stream Reasoning**
  - **Syntax;**
  - **Semantics;**
  - **Experimental Results;**
- Conclusion.

ETALIS: Language Syntax

ETALIS Language for Events is formally defined by:

$$P ::= \mathtt{pr}(t_1, \ldots, t_n) \quad | \; P \text{ WHERE } t \;|\; q \;|\; (P).q \\ \qquad\qquad\qquad\qquad | \; P \text{ BIN } P \;|\; \text{NOT}(P).[P, P]$$

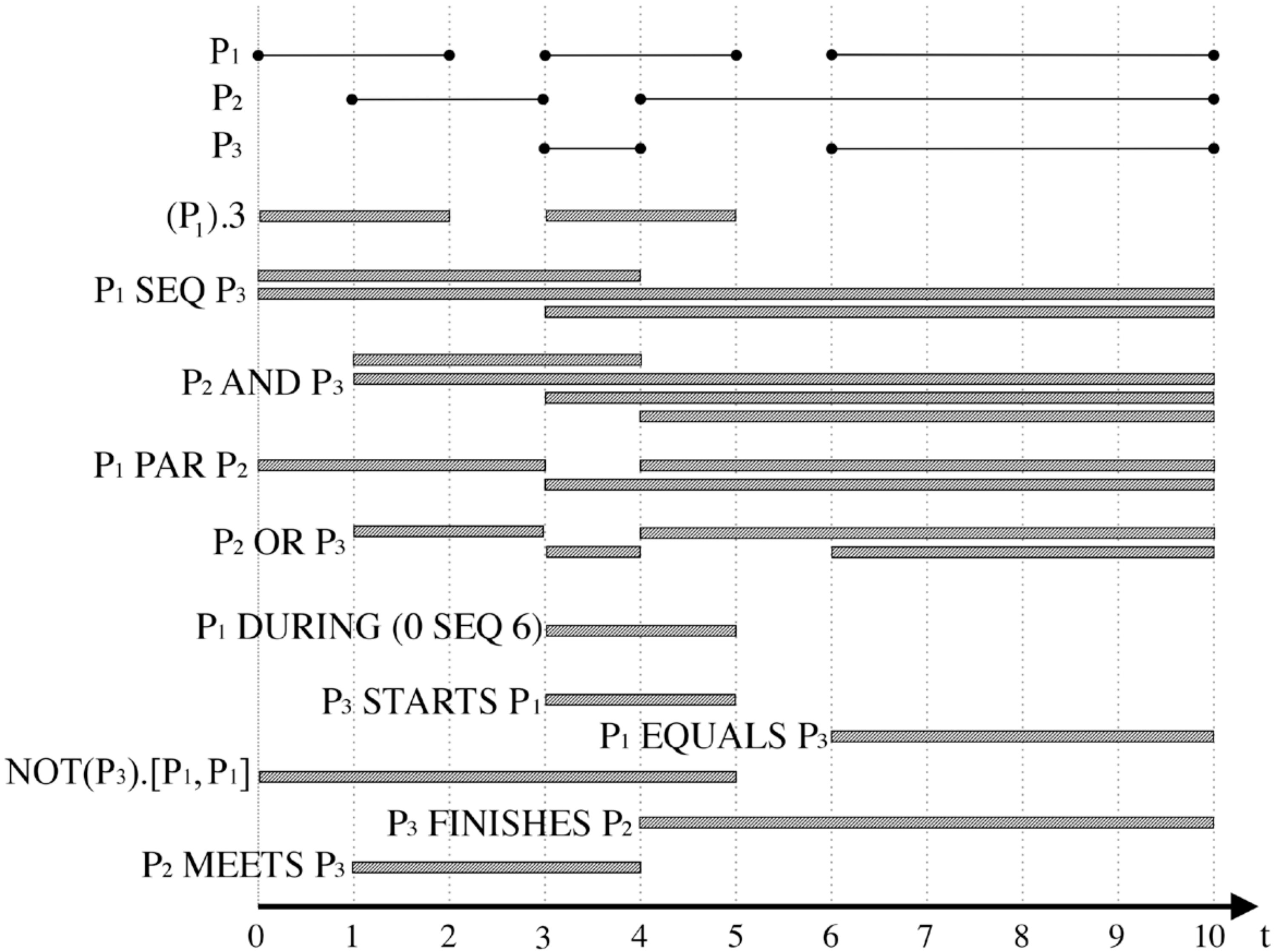
- pr - a predicate name with arity n;
- $t_{(i)}$ - denote terms;
- t - is a term of type boolean;
- q - is a nonnegative rational number;
- BIN - is one of the binary operators: SEQ, AND, PAR, OR, EQUALS, MEETS, STARTS, or FINISHES.

Event rule is defined as a formula of the following shape:

$$\mathtt{pr}(t_1, \ldots, t_n) \leftarrow p$$

where p is an event pattern containing all variables occurring in $\mathtt{pr}(t_1, \ldots, t_n)$

ETALIS: Interval-based Semantics



ETALIS: Formal Semantics

pattern	$\mathcal{I}_\mu(\text{pattern})$
$\mathtt{pr}(t_1,\ldots,t_n)$	$\mathcal{I}(\mathtt{pr}(\mu^*(t_1),\ldots,\mu^*(t_n)))$
p WHERE t	$\mathcal{I}_\mu(p)$ if $\mu^*(t) = \mathit{true}$ $\emptyset$ otherwise.
q	$\{\langle q,q\rangle\}$ for all $q \in \mathbb{Q}^+$
$(p).q$	$\mathcal{I}_\mu(p) \cap \{\langle q_1,q_2\rangle \mid q_2 - q_1 = q\}$
p_1 SEQ p_2	$\{\langle q_1,q_4\rangle \mid \langle q_1,q_2\rangle \in \mathcal{I}_\mu(p_1)$ and $\langle q_3,q_4\rangle \in \mathcal{I}_\mu(p_2)$ for some $q_2,q_3 \in \mathbb{Q}^+$ with $q_2 < q_3\}$
p_1 AND p_2	$\{\langle \min(q_1,q_3),\max(q_2,q_4)\rangle \mid \langle q_1,q_2\rangle \in \mathcal{I}_\mu(p_1)$ and $\langle q_3,q_4\rangle \in \mathcal{I}_\mu(p_2)$ for some $q_2,q_3 \in \mathbb{Q}^+\}$
p_1 PAR p_2	$\{\langle \min(q_1,q_3),\max(q_2,q_4)\rangle \mid \langle q_1,q_2\rangle \in \mathcal{I}_\mu(p_1)$ and $\langle q_3,q_4\rangle \in \mathcal{I}_\mu(p_2)$ for some $q_2,q_3 \in \mathbb{Q}^+$ with $\max(q_1,q_3) < \min(q_2,q_4)\}$
p_1 OR p_2	$\mathcal{I}_\mu(p_1) \cup \mathcal{I}_\mu(p_2)$
p_1 EQUALS p_2	$\mathcal{I}_\mu(p_1) \cap \mathcal{I}_\mu(p_2)$
p_1 MEETS p_2	$\{\langle q_1,q_3\rangle \mid \langle q_1,q_2\rangle \in \mathcal{I}_\mu(p_1)$ and $\langle q_2,q_3\rangle \in \mathcal{I}_\mu(p_2)$ for some $q_2 \in \mathbb{Q}^+\}$
p_1 DURING p_2	$\{\langle q_3,q_4\rangle \mid \langle q_1,q_2\rangle \in \mathcal{I}_\mu(p_1)$ and $\langle q_3,q_4\rangle \in \mathcal{I}_\mu(p_2)$ for some $q_2,q_3 \in \mathbb{Q}^+$ with $q_3 < q_1 < q_2 < q_4\}$
p_1 STARTS p_2	$\{\langle q_1,q_3\rangle \mid \langle q_1,q_2\rangle \in \mathcal{I}_\mu(p_1)$ and $\langle q_1,q_3\rangle \in \mathcal{I}_\mu(p_2)$ for some $q_2 \in \mathbb{Q}^+$ with $q_2 < q_3\}$
p_1 FINISHES p_2	$\{\langle q_1,q_3\rangle \mid \langle q_2,q_3\rangle \in \mathcal{I}_\mu(p_1)$ and $\langle q_1,q_3\rangle \in \mathcal{I}_\mu(p_2)$ for some $q_2 \in \mathbb{Q}^+$ with $q_1 < q_2\}$
NOT$(p_1).[p_2,p_3]$	$\mathcal{I}_\mu(p_2$ SEQ $p_3) \setminus \mathcal{I}_\mu(p_2$ SEQ p_1 SEQ $p_3)$

Definition of extensional interpretation of event patterns. We use $p_{(x)}$ for patterns, $q_{(x)}$ for rational numbers, $t_{(x)}$ for terms and $\mathtt{pr}$ for event predicates.

ETALIS: Operational Semantics (SEQ)

a SEQ b SEQ c ⟶ ce1

((a SEQ b) SEQ c) ⟶ ce1

a SEQ b ⟶ ie
ie SEQ c ⟶ ce1

1. Complex pattern (not event-driven rule)

2. Decoupling

3. Binarization

4. Event-driven backward chaining rules

Algorithm 1 Sequence.

Input: event binary goal $\texttt{ie} \leftarrow \texttt{a SEQ b}$.

Output: event-driven backward chaining rules for SEQ operator.

Each event binary goal $\texttt{ie} \leftarrow \texttt{a SEQ b}$ is converted into: {

$$\texttt{a}(T_1, T_2) :- \texttt{for\_each}(\texttt{a}, 1, [T_1, T_2]).$$
$$\texttt{a}(1, T_1, T_2) :- \texttt{assert}(\texttt{goal}(\texttt{b}(\_,\_), \texttt{a}(T_1, T_2), \texttt{ie}(\_,\_))).$$
$$\texttt{b}(T_3, T_4) :- \texttt{for\_each}(\texttt{b}, 1, [T_3, T_4]).$$
$$\texttt{b}(1, T_3, T_4) :- \texttt{goal}(\texttt{b}(T_3, T_4), \texttt{a}(T_1, T_2), \texttt{ie}), T_2 < T_3, \texttt{retract}(\texttt{goal}(\texttt{b}(T_3, T_4), \texttt{a}(T_1, T_2), \texttt{ie}(\_,\_))), \texttt{ie}(T_1, T_4).$$

}

ETALIS: Operational Semantics (rSEQ)

FZI

Algorithm 5 Sequence with retraction.

Input: event binary goal $\mathtt{ie}_1 \leftarrow \mathtt{a}\ \mathtt{SEQ}\ \mathtt{b}$.

Output: event-driven backward chaining rules for SEQ operator including retraction.

Each event binary goal $\mathtt{ie}_1 \leftarrow \mathtt{a}\ \mathtt{SEQ}\ \mathtt{b}$ is converted into: {

$$
\begin{aligned}
\mathtt{a}(ID, [T_1, T_2]) &:- \mathtt{for\_each}(\mathtt{a}, 1, ID, [T_1, T_2]).\\
\mathtt{a}(1, ID, [T_1, T_2]) &:- \mathtt{assert}(\mathtt{goal}(\mathtt{b}(\_, [\_, \_]), \mathtt{a}(ID, [T_1, T_2]),\\
&\qquad \mathtt{ie}_1(\_, [\_, \_]))).\\
\mathtt{rev\_a}(ID, [T_3, T_4]) &:- \mathtt{for\_each}(\mathtt{rev\_a}, 1, ID, [T_3, T_4]).\\
\mathtt{rev\_a}(1, ID, [T_3, T_4]) &:- \mathtt{goal}(\mathtt{b}(\_, [\_, \_]), \mathtt{a}(ID, [T_1, T_2]),\\
&\qquad \mathtt{ie}_1(\_, [\_, \_])), \mathtt{retract}(\mathtt{goal}(\mathtt{b}(\_, [\_, \_]),\\
&\qquad \mathtt{a}(ID, [T_1, T_2]))).\\
\mathtt{rev\_a}(2, ID, [T_3, T_4]) &:- (\mathtt{ie}_1(ID, [T_1, T_2]),\\
&\qquad \mathtt{retract}(\mathtt{ie}_1(ID, [T_1, T_2])), \mathtt{rev\_ie}_1(ID, [T_1, T_2]));\\
&\qquad \mathtt{true}.\\
\mathtt{b}(ID, [T_3, T_4]) &:- \mathtt{for\_each}(\mathtt{b}, 1, ID, [T_3, T_4]).\\
\mathtt{b}(1, ID, [T_3, T_4]) &:- \mathtt{goal}(\mathtt{b}(\_, [\_, \_]), \mathtt{a}(ID, [T_1, T_2]),\\
&\qquad \mathtt{ie}_1(\_, [\_, \_])), T_2 < T_3, \mathtt{ie}_1(ID, [T_1, T_4]).\\
\mathtt{rev\_b}(ID, [T_5, T_6]) &:- \mathtt{for\_each}(\mathtt{rev\_b}, 1, ID, [T_5, T_6]).\\
\mathtt{rev\_b}(1, ID, [T_5, T_6]) &:- (\mathtt{ie}_1(ID, [T_1, T_4]),\\
&\qquad \mathtt{retract}(\mathtt{ie}_1(ID, [T_1, T_4])), \mathtt{rev\_ie}_1(ID, [T_1, T_4]));\\
&\qquad \mathtt{true}.\\
\mathtt{ie}_1(ID, [T_1, T_4]) &:- \mathtt{for\_each}(\mathtt{ie}_1, 1, ID, [T_1, T_4]).\\
\mathtt{ie}_1(1, ID, [T_1, T_4]) &:- \mathtt{assert}(\mathtt{ie}_1(ID, [T_1, T_4])).
\end{aligned}
$$

}

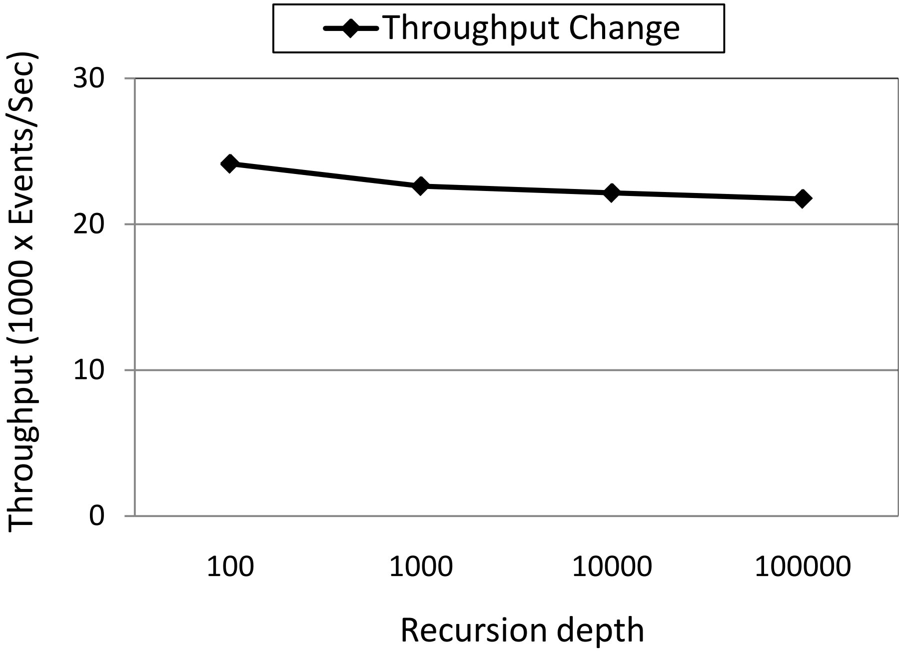
Tests I: CEP with Stream Reasoning

$$\texttt{trendIncrease}() \leftarrow (\texttt{stockIcr}(CompanyA)\ \textsc{SEQ}\ \texttt{stockIcr}(CompanyB)).10\ \textsc{AND}\ \texttt{inSupChain}(CompanyA, CompanyB).$$

$$\texttt{inSupChain}(X,Y) \leftarrow \texttt{linked}(X,Y).$$
$$\texttt{inSupChain}(X,Z) \leftarrow \texttt{linked}(X,Y)\ \textsc{AND}\ \texttt{inSupChain}(Y,Z).$$

$$\texttt{linked}(CompanyA, CompanyB)$$
...
$$\texttt{linked}(CompanyY, CompanyZ)$$

Intel Core Quad CPU Q9400 2,66GHz, 8GB of RAM, Vista x64; ETALIS on SWI Prolog 5.6.64 and YAP Prolog 5.1.3 vs. Esper 3.3.0



Tests II: Throughput Comparison

$$\mathsf{e}(ID) \leftarrow \mathsf{a}(ID)\ \textsc{bin}\ \mathsf{b}(ID).$$

$$\mathsf{e}(ID) \leftarrow \textsc{not}(\mathsf{c}(ID)).[\mathsf{a}(ID)\ \textsc{seq}\ \mathsf{b}(ID)].$$

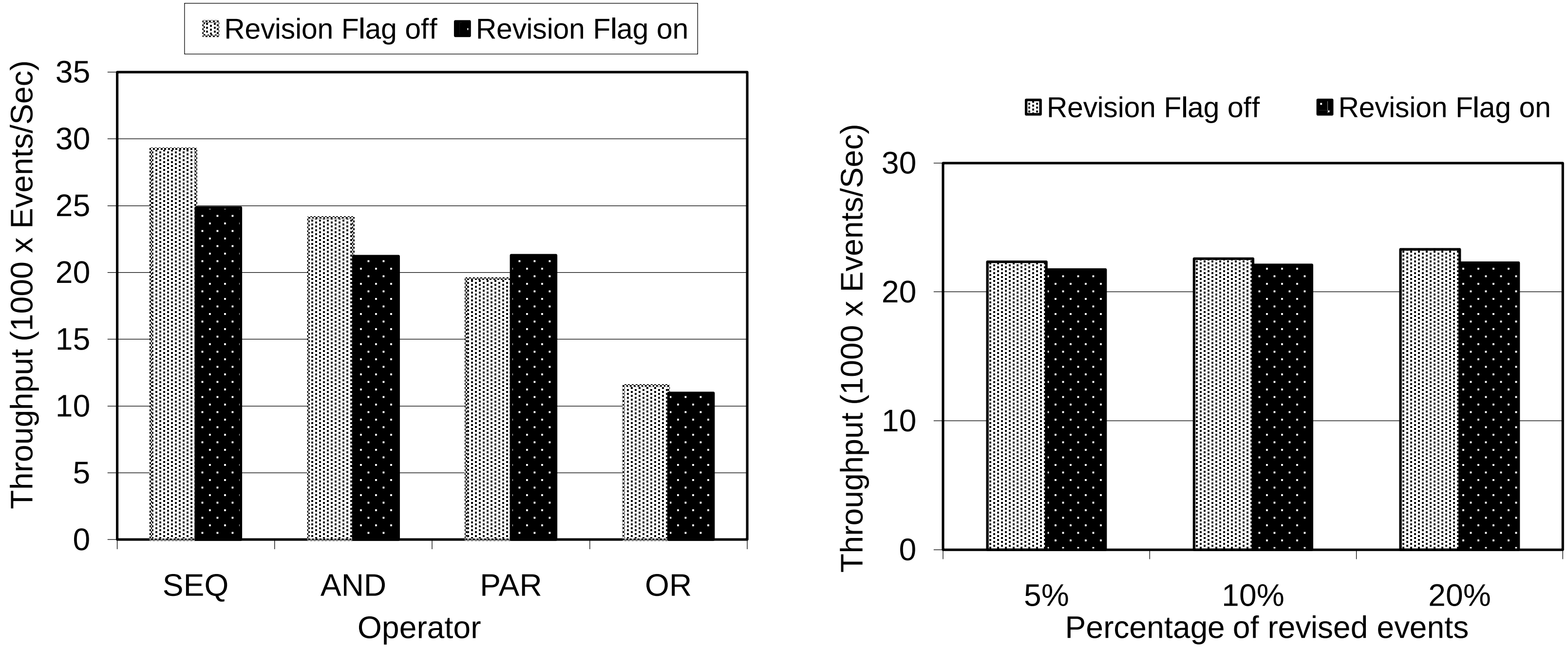
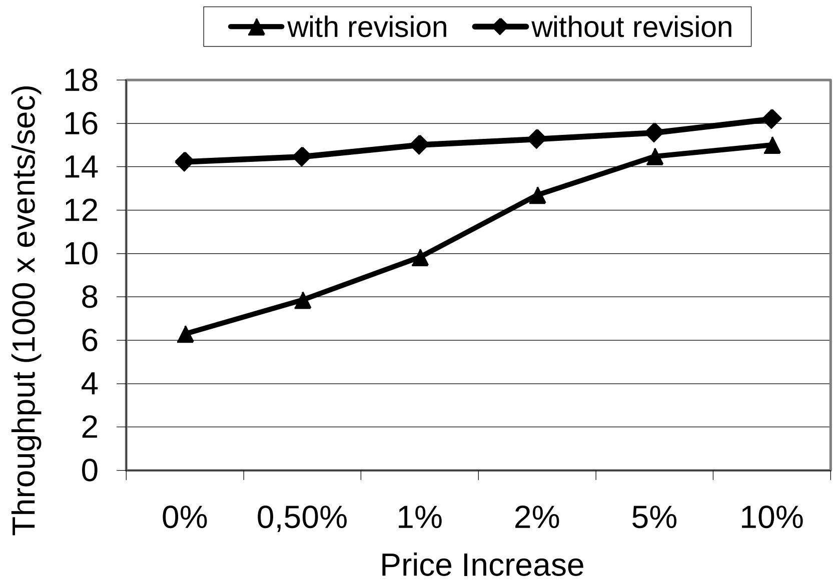


Figure: Throughput (a) Various operaors (b) Negation

Tests III: Stock price change on a real data set

$$\texttt{stockIncr}(ID, Adj_1, Adj_2) \leftarrow$$
$$\texttt{stock}(ID, Date_1, Opn_1, High_1, Low_1, Cls_1, Vol_1, Adj_1)$$
$$\text{SEQ}$$
$$\texttt{stock}(ID, Date_2, Opn_2, High_2, Low_2, Cls_2, Vol_2, Adj_2)$$
$$\text{WHERE } (Adj_1 * X < Adj_2).$$

- Yahoo Finance: IBM stocks from 1962 up to now
- 5% revision tulples introduced

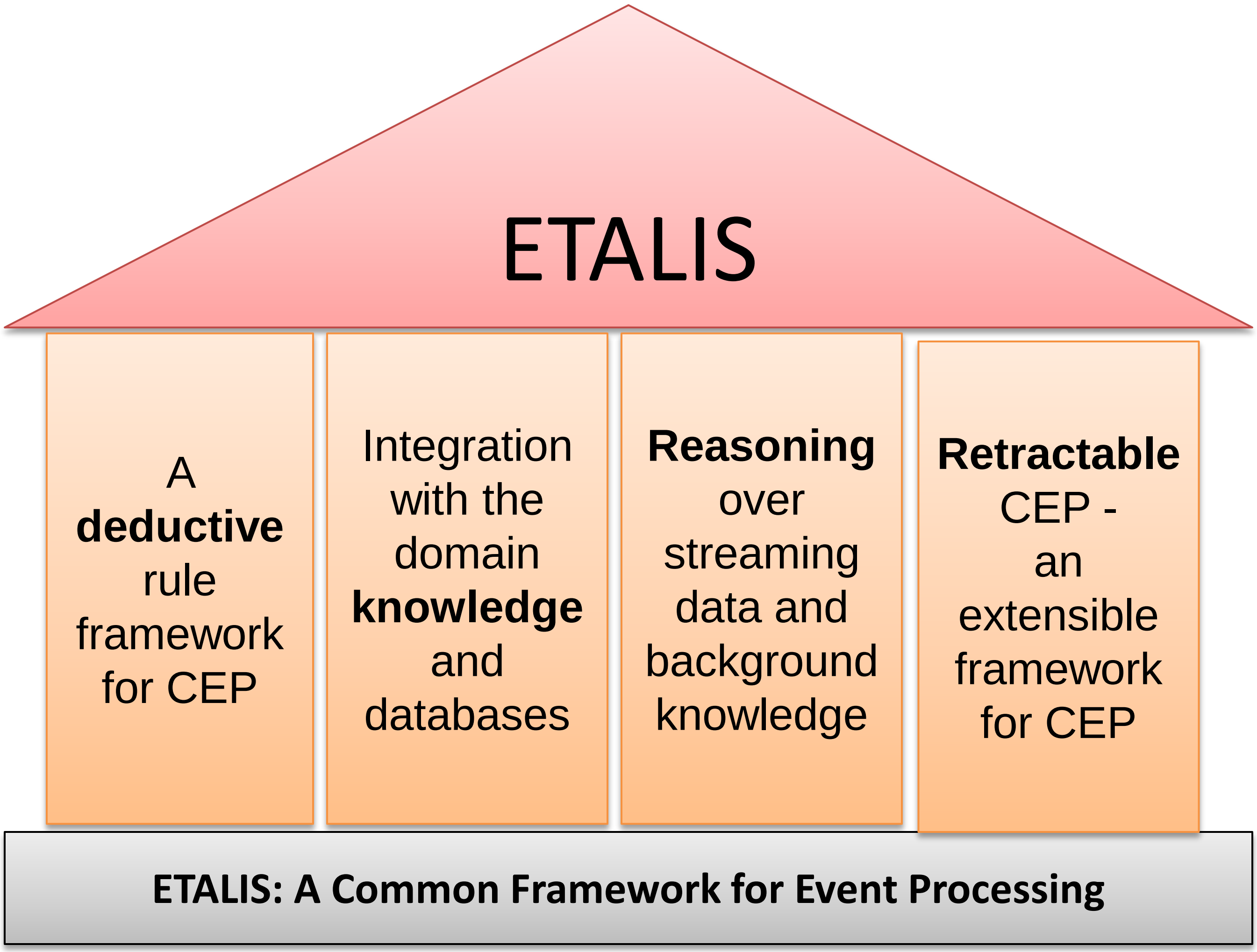


Agenda

FZI

- Introduction, Motivation
- ETALIS Language for Events
  - Syntax;
  - Semantics;
  - Experimental Results;
- **Conclusion.**

Conclusion: A Common Framework for Event Processing in ETALIS



Thank you! Questions…

FZI

ETALIS

Open source:

http://code.google.com/p/etalis

Darko.Anicic@fzi.de