Stream Reasoning For Linked Data

M. Balduini, J-P Calbimonte, O. Corcho, D. Dell'Aglio, E. Della Valle, and J.Z. Pan

http://streamreasoning.org/sr4ld2013

ISWC 2013
Sydney, Australia

Other Stream Reasoning approaches

Oscar Corcho, Emanuele Della Valle
ocorcho@fi.upm.es, emanuele.dellavalle@polimi.it

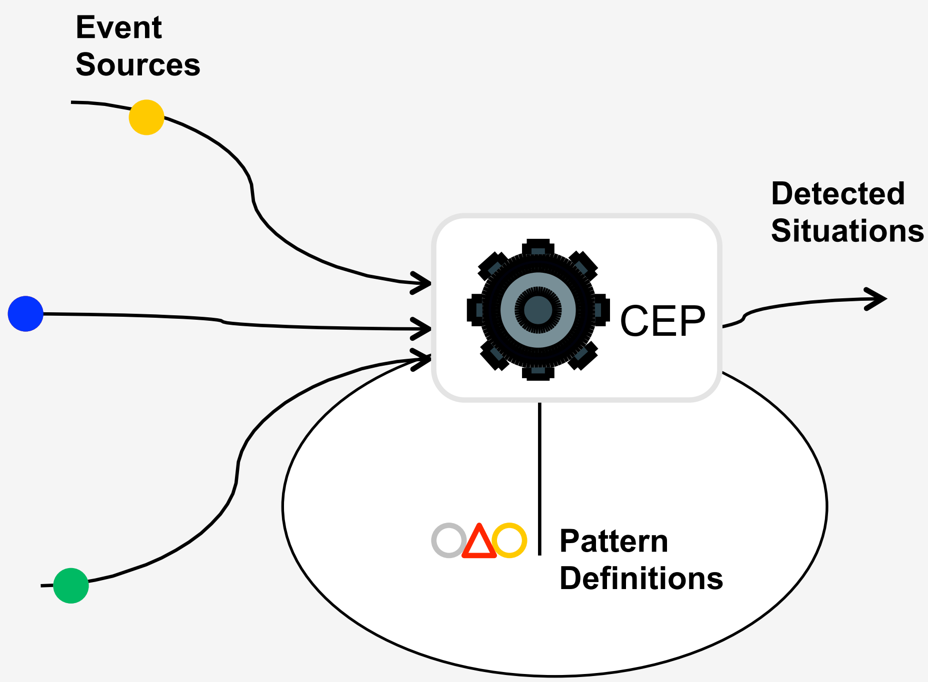
Share, Remix, Reuse — Legally

Agenda

- ETALIS and EP-SPARQL
  - A Declarative Framework for Matching Iterative and Aggregative Patterns against Event Streams
    - Darko Anicic, Sebastian Rudolph, Paul Fodor, Nenad Stojanovic

Iterative and Aggregative Patterns in ETALIS

Recursive CEP in ETALIS



ETALIS Features:

- Logic-based CEP
  - Stream (deductive) reasoning
- Iterative and aggregative patterns
- Implementation
  - http://code.google.com/p/etalis

ETALIS - A Logic Rule-based CEP

- **Iterative patterns**
  - An output (complex) event is treated as an input event of the same CEP processing agent;
- A **rule-based approach**
  - Rules can express complex relationships between events by matching certain **temporal**, **relational** or **causal** conditions
  - It can specify and evaluate **contextual knowledge**

ETALIS: Language Syntax

ETALIS Language for Events is formally defined by:

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

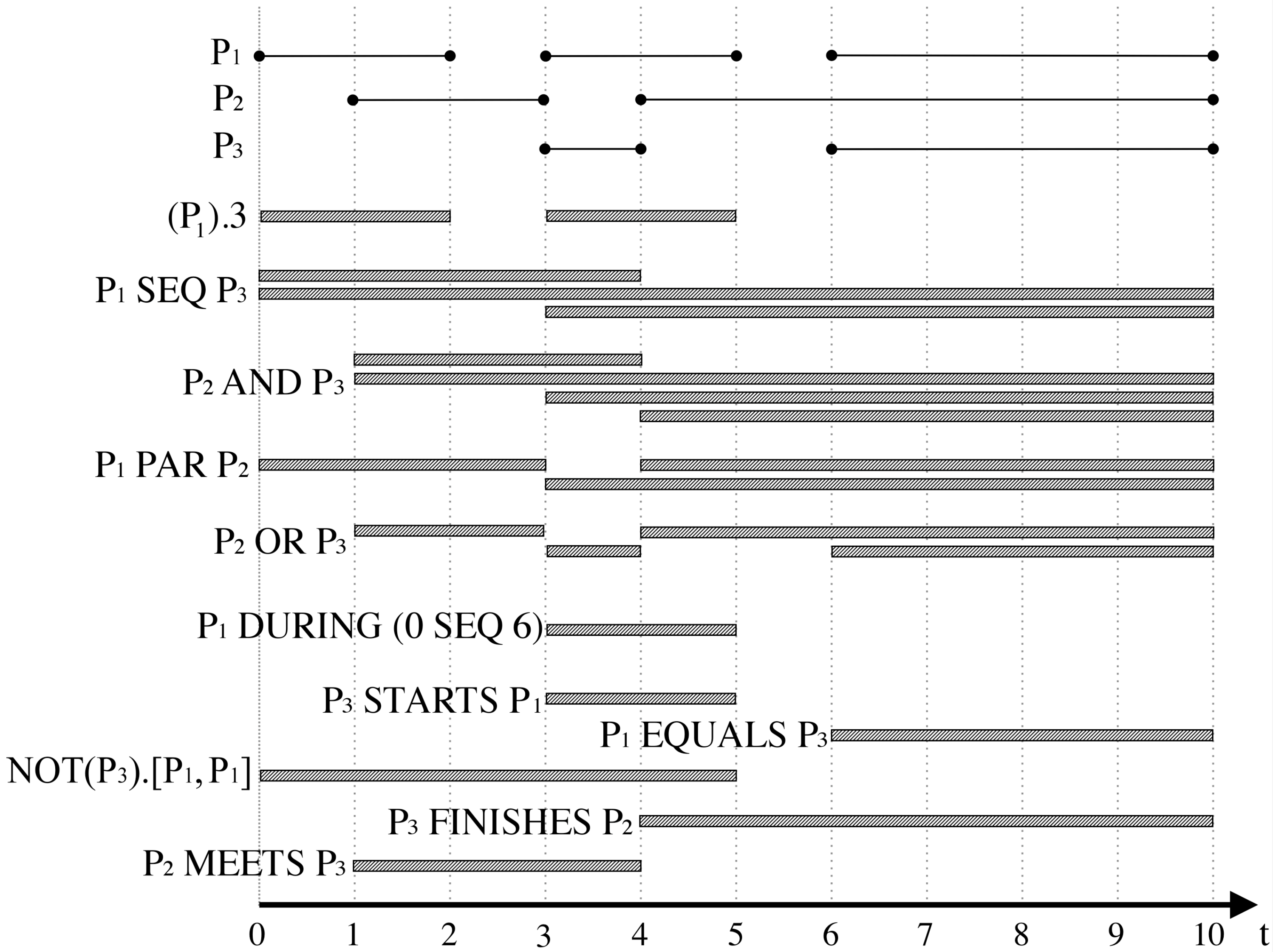
- pr - a predicate name with arity n;
- $t_{(i)}$ - denote terms;
- t - is a term of type boolean;
- q - is a nonnegative rational number;
- BIN - is one of the binary operators: SEQ, AND, PAR, OR, EQUALS, MEETS, STARTS, or FINISHES.

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: Declarative Semantics

pattern	$\mathcal{I}_\mu$(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.

EP-SPARQL (I)

- Basics
  - SPARQL extension (as with other previously seen languages)
  - Interval-based: 2 timestamps

RDF stream – a set of *triple occurrences* $\langle\langle s, p, o\rangle, t_\alpha, t_\omega\rangle$ where $\langle s, p, o\rangle$ is an RDF triple and t_α, t_ω are the start and end of the interval.

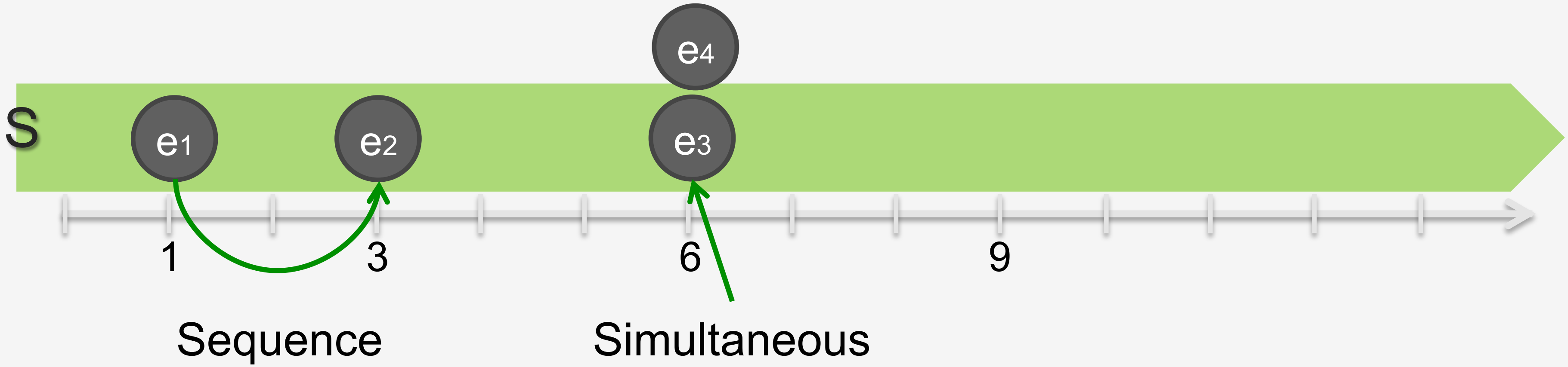
- Operators
  - FILTER, AND, UNION, OPTIONAL, SEQ, EQUALS, OPTIONALSEQ, and EQUALSOPTIONAL
    - Be careful with the management of timestamps (see next)
    - E.g.,

AND – joins $\langle \mu, t_\alpha, t_\omega\rangle$ and $\langle \mu', t'_\alpha, t'_\omega\rangle$. The joined tuple has timestamp $t''_\alpha = \min(t_\alpha, t'_\alpha)$, $t''_\omega = \max(t_\omega, t'_\omega)$;

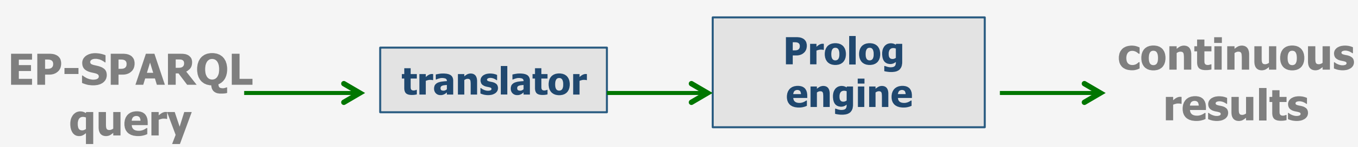
- Special functions
  - getDuration(), getStartTime(), getEndTime()

EP-SPARQL (II)

- Sequence operators and CEP world



- SEQ: joins $e_{ti,tf}$ and $e'_{ti',tf'}$ if e' occurs after e
- EQUALS: joins $e_{ti,tf}$ and $e'_{ti',tf'}$ if they occur simultaneously
- OPTIONALSEQ, OPTIONALEQUALS: Optional join variants



EP-SPARQL Example

Continuously search for companies having a larger than 20% stock price increase in less than 15 days without having acquired another company during that period.

```
SELECT ?company WHERE
       { ?company hasStockprice ?price1 }
   SEQ { { ?company hasAcquired ?othercompany }
         OPTIONALSEQ
         { ?company hasStockPrice ?price2 } }
FILTER ( ?price2 > ?price1 * 1.2 &&
         !BOUND(?othercompany) &&
         getDURATION() < "P15D"^^xsd:duration)
```

EP-SPARQL sample translation (SEQ)

```
SELECT ?company WHERE
       { ?comp hasStockPrice ?pr1 }
   SEQ { ?comp hasStockPrice ?pr2 }
   SEQ { ?comp hasStockPrice ?pr3 }
```

$\langle\langle s, p, o\rangle, t_i, t_j\rangle$ represented as $triple(s, p, o, T_i, T_j)$, and τ represents s, p, o.

$triple(\tau_i, T_1, T_4) \leftarrow triple(\tau_1, T_1, T_2)$ SEQ $triple(\tau_2, T_3, T_4)$.
$triple(\tau, T_1, T_6) \leftarrow triple(\tau_i, T_1, T_4)$ SEQ $triple(\tau_3, T_5, T_6)$.

Rule transformation – Incremental computation (Prolog syntax)

$triple(\tau_1, T_1, T_2)$:-
  $assert(goal(triple(\tau_2, \_, \_), triple(\tau_1, T_1, T_2), triple(\tau_i, \_, \_)))$.

$triple(\tau_2, T_3, T_4) : -$
  $goal(triple(\tau_2, \_, \_), triple(\tau_1, T_1, T_2), triple(\tau, \_, \_))$,
  $T_2 < T_3$,
  $retract(goal(triple(\tau_2, \_, \_), triple(\tau_1, T_1, T_2), triple(\tau_i, \_, \_)))$,
  $triple(\tau_i, T_1, T_4)$.

Stream Reasoning For Linked Data

M. Balduini, J-P Calbimonte, O. Corcho, D. Dell'Aglio, E. Della Valle, and J.Z. Pan

http://streamreasoning.org/sr4ld2013

ISWC 2013
Sydney, Australia

Other Stream Reasoning approaches

Oscar Corcho, Emanuele Della Valle
ocorcho@fi.upm.es, emanuele.dellavalle@polimi.it