Event Processing and Stream Reasoning with ETALIS

Darko Anicic

FZI Research Center for Information Technology, Germany

INQUEST 2012, Oxford, United Kingdom
Agenda

- Motivation and Introduction
- ETALIS Approach
- Implementation & Evaluation
- Applications
- Future Work

Context

  http://darko-anicic.eu/publications#thesis
MOTIVATION AND INTRODUCTION
Shifting Event Processing Toward More Intelligent Event Processing
Shifting Reasoning Toward Stream Reasoning
Scenario: Setting Speed Limits on Roads

How to capture & detect many possible situations?
- Define abstract, high-level situations, as patterns, and leave specific cases related to those situations to be inferred from the domain knowledge.

How to detect criticality of situations and their possible implications?
- A model which captures dependencies between situations and enable reasoning about implications.

Collision
- head-on (50)
- side (60)
- rear-end (50)
- rollover (40)
- pile-up (10)

Weather
- rain
- tires lose traction
- snow
- with low visibility
- ice, fog …

Road
- A1
- A1-2
- A1-2
- A10-11
- A2-3
- A20-21
Event Processing Enhanced with Domain Knowledge

**Situation**

```
+-------------------+-----------
| Temporal Knowledge| Explicit & Implicit (Static) Knowledge |
```

Reasoning about sub-class relations as reasoning about more specific or more general situations

Reasoning about transitive or indirect situations to detect relevant situations

```
Rd_1 ⊆ Ar_1
```

25.09.2012
© FZI Forschungszentrum Informatik
Event Processing & Stream Reasoning –
A Logic Programming Approach

- Background knowledge provides the context or domain in which events are interpreted.

\[ \pi = ? \]
\[ \pi \leftarrow \varphi \lor \neg \psi. \]
\[ \varphi = \text{true}. \]

Figure source: Tali Yatzkar & Opher Etzion, IBM Research
Event Processing and Stream Reasoning

ETALIS APPROACH
Overview of ETALIS Approach

**ETALIS Foundations**
- ETALIS Language
- Execution Model
- EPN in ETALIS

**ETALIS Extensions**
- Retraction in EP
- Out-of-Order EP
- EP-SPARQL

**Practical Considerations**
- Implementation
- Evaluation
- ETALIS in Use
ETALIS Language for Events - Syntax

A predicate name with arity n, t(i) denotes terms

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

BIN: SEQ, AND, PAR, OR, EQUALS, MEETS, STARTS, or FINISHES

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

\[ \text{pr}(t_1, \ldots, t_n) \leftarrow p \]

where p is an event pattern containing all variables occurring in \( \text{pr}(t_1, \ldots, t_n) \)
ETALIS: Interval-based Semantics
ETALIS Language for Events - Semantics

<table>
<thead>
<tr>
<th>pattern</th>
<th>$\mathcal{I}_\mu$(pattern)</th>
</tr>
</thead>
<tbody>
<tr>
<td>pr$(t_1,\ldots,t_n)$</td>
<td>$\mathcal{I}(\text{pr}(\mu^<em>(t_1),\ldots,\mu^</em>(t_n)))$</td>
</tr>
</tbody>
</table>
| p WHERE t | $\mathcal{I}_\mu(p)$ if $\mu^*(t) = \text{true}$  
$\emptyset$ 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$_2$ | $\{\langle q_1, q_4 \rangle \mid \langle q_1, q_2 \rangle \in \mathcal{I}_\mu(p_1) \text{ and } \langle q_3, q_4 \rangle \in \mathcal{I}_\mu(p_2) \text{ for some } q_2, q_3 \in \mathbb{Q}^+ \text{ with } q_2 < q_3\}$ |
| p$_1$ AND p$_2$ | $\{\langle \min(q_1, q_3), \max(q_2, q_4) \rangle \mid \langle q_1, q_2 \rangle \in \mathcal{I}_\mu(p_1) \text{ and } \langle q_3, q_4 \rangle \in \mathcal{I}_\mu(p_2) \text{ for some } q_2, q_3 \in \mathbb{Q}^+\}$ |
| p$_1$ PAR p$_2$ | $\{\langle \min(q_1, q_3), \max(q_2, q_4) \rangle \mid \langle q_1, q_2 \rangle \in \mathcal{I}_\mu(p_1) \text{ and } \langle q_3, q_4 \rangle \in \mathcal{I}_\mu(p_2) \text{ for some } q_2, q_3 \in \mathbb{Q}^+ \text{ with } \max(q_1, q_3) < \min(q_2, q_4)\}$ |
| 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$_2$ | $\{\langle q_1, q_3 \rangle \mid \langle q_1, q_2 \rangle \in \mathcal{I}_\mu(p_1) \text{ and } \langle q_2, q_3 \rangle \in \mathcal{I}_\mu(p_2) \text{ for some } q_2 \in \mathbb{Q}^+\}$ |
| p$_1$ DURING p$_2$ | $\{\langle q_3, q_4 \rangle \mid \langle q_1, q_2 \rangle \in \mathcal{I}_\mu(p_1) \text{ and } \langle q_3, q_4 \rangle \in \mathcal{I}_\mu(p_2) \text{ for some } q_2, q_3 \in \mathbb{Q}^+ \text{ with } q_3 < q_1 < q_2 < q_4\}$ |
| p$_1$ STARTS p$_2$ | $\{\langle q_1, q_3 \rangle \mid \langle q_1, q_2 \rangle \in \mathcal{I}_\mu(p_1) \text{ and } \langle q_1, q_3 \rangle \in \mathcal{I}_\mu(p_2) \text{ for some } q_2 \in \mathbb{Q}^+ \text{ with } q_2 < q_3\}$ |
| p$_1$ FINISHES p$_2$ | $\{\langle q_1, q_3 \rangle \mid \langle q_2, q_3 \rangle \in \mathcal{I}_\mu(p_1) \text{ and } \langle q_1, q_3 \rangle \in \mathcal{I}_\mu(p_2) \text{ for some } q_2 \in \mathbb{Q}^+ \text{ with } q_1 < q_2\}$ |
| $\text{NOT}(p_1)$, $[p_2, p_3]$ | $\mathcal{I}_\mu(p_2 \text{ SEQ } p_3) \setminus \mathcal{I}_\mu(p_2 \text{ SEQ } p_1 \text{ SEQ } p_3)$ |

Definition of extensional interpretation of event patterns. We use $p(x)$ for patterns, $q(x)$ for rational numbers, $t(x)$ for terms and pr for event predicates.

25.09.2012 © FZI Forschungszentrum Informatik
Example: Event Filtering

heatIndexEffect(\text{Note, Area}) \leftarrow \\
\text{heatIndex}(<Loc, \text{Index}>)
\text{WHERE}\{\text{shadeValuesRule}(<\text{Index, Note}>), \text{areaRule}(<\text{Loc, Area}>)\}.

\text{shadeValuesRule}(\text{Index,'Caution'}) : \ - 80 \leq \text{Index, Index} < 90, !. \\
\text{shadeValuesRule}(\text{Index,'ExtremeCaution'}) : \ - 90 \leq \text{Index, Index} < 105, !. \\
\text{shadeValuesRule}(\text{Index,'Danger'}) : \ - 105 \leq \text{Index, Index} < 130, !. \\
\text{shadeValuesRule}(\text{Index,'ExtremeDanger'}) : \ - 130 \leq \text{Index, !}. \\

\text{areaRule}(\text{loc}(<\text{N', X,'W', Y}>, \text{Area}_1)) : \ - 4042 < X, X < 4049, 7358 < Y, Y < 7370, !. \\
\text{...} \\
\text{areaRule}(\text{loc}(<\text{N', X,'W', Y}>, \text{Area}_n)) : \ - 4034 < X, X < 4040, 7368 < Y, Y < 7399, !.
Example: Sub-Class Relations

```
enhancedFire(Loc) ←
  (activeFire(Loc) AND weatherObservation(Loc, Observ)) \cdot 3 hours
WHERE (rdfs:subClassOf(Observ, 'wt:WindObservation').
```

```
observ_1
  rdf:type         wt:diablo ;
  wt:speed         "60"^^xsd:int ;
  wt:temperature   "30"^^xsd:int ;
  wt:region        "California"^^xsd:string .

wt:windObservation rdfs:subClassOf wt:weatherObservation .
wt:diablo          rdfs:subClassOf wt:windObservation .
wt:sundowner       rdfs:subClassOf wt:windObservation .

rdf : type(a, Y) : − rdfs : subClassOf(X, Y), rdf : type(a, X).
```
Example: Iterative and Aggregative Patterns

- The k-fold sequential execution of an event a:

\[
\text{iteration}(a, 1) \leftarrow a. \\
\text{iteration}(a, k + 1) \leftarrow a \text{ SEQ } \text{iteration}(a, k).
\]

- A length-based window of size n:

\[
\text{iteration}(a, 1) \leftarrow a. \\
\text{iteration}(a, k + 1) \leftarrow \text{NOT}(a).[a, \text{iteration}(a, k)]. \\
\text{e} \leftarrow \text{iteration}(a, n).
\]

- A sum over an unbound event stream until a threshold value is met:

\[
\text{income}(\text{Price}) \leftarrow \text{sell}(\text{Item, Price}). \\
\text{income}(P1 + P2) \leftarrow \text{income}(P1) \text{ SEQ } \text{sell}(\text{Item, P2}). \\
\text{bigincome} \leftarrow \text{income}(\text{Price}) \text{ WHERE } \text{Price} > 100000.
\]
ETALIS: Operational Semantics (SEQ)

1. Complex event pattern

2. Decoupling

3. Binarization

4. Event-driven backward chaining (EDBC) rules

Algorithm 1 Sequence.

**Input:** event binary goal \( \text{ie} \leftarrow \text{a\ SEQ\ b} \).

**Output:** event-driven backward chaining rules for \( \text{SEQ} \) operator.

Each event binary goal \( \text{ie} \leftarrow \text{a\ SEQ\ b} \) is converted into:

\[
\begin{align*}
\text{a}(T_1, T_2) & \colon - \text{for.each}(a, 1, [T_1, T_2]). \\
\text{a}(1, T_1, T_2) & \colon - \text{assert(goal(b(\_, \_), a(T_1, T_2), \text{ie}(\_, \_))).} \\
\text{b}(T_3, T_4) & \colon - \text{for.each}(b, 1, [T_3, T_4]). \\
\text{b}(1, T_3, T_4) & \colon - \text{goal(b(T_3, T_4), a(T_1, T_2), ie), T_2 < T_3,} \\
& \quad \text{retract(goal(b(T_3, T_4), a(T_1, T_2), ie(\_, \_)), ie(T_1, T_4)).}
\end{align*}
\]
Order of Rule Execution: SEQ & AND

\[\text{c SEQ b } \rightarrow \text{ a} \]
\[\text{b SEQ a } \rightarrow \text{ ie}\]

\[\text{c SEQ b } \rightarrow \text{ a} \]
\[\text{b AND a } \rightarrow \text{ ie}\]

**Algorithm 2 Conjunction.**

**Input:** event binary goal \(\text{ie } \leftarrow \text{ a AND b}\).

**Output:** event-driven backward chaining rules for \(\text{AND}\) operator.

Each event binary goal \(\text{ie } \leftarrow \text{ a AND b}\) is converted into:

\[
\begin{align*}
\text{a}(T_1, T_2) & : \neg \text{ for each (a, 1, [T_1, T_2]).} \\
\text{a}(1, T_3, T_4) & : \neg \text{ goal(a(\_, \_), b(T_1, T_2), ie(\_, \_)),} \\
\text{retract(goal(a(\_, \_), b(T_1, T_2), ie(\_, \_))))}, \\
T_5 & = \text{min}\{T_1, T_3\}, T_6 = \text{max}\{T_2, T_4\}, \text{ie(T_5, T_6)}. \\
\text{a}(2, T_3, T_4) & : \neg \text{ goal(a(\_, \_), b(T_1, T_2), ie(\_, \_))),} \\
\text{assert(goal(b(\_, \_), a(T_3, T_4), ie(\_, \_))))}. \\
\end{align*}
\]

\[
\begin{align*}
\text{b}(T_1, T_2) & : \neg \text{ for each (b, 1, [T_1, T_2]).} \\
\text{b}(1, T_3, T_4) & : \neg \text{ goal(b(\_, \_), a(T_1, T_2), ie(\_, \_)),} \\
\text{retract(goal(b(\_, \_), a(T_1, T_2), ie(\_, \_))))}, \\
T_5 & = \text{min}\{T_1, T_3\}, T_6 = \text{max}\{T_2, T_4\}, \text{ie(T_5, T_6)}. \\
\text{b}(2, T_3, T_4) & : \neg \text{ goal(b(\_, \_), a(T_1, T_2), ie(\_, \_))),} \\
\text{assert(goal(a(\_, \_), b(T_3, T_4), ie(\_, \_))))}. \\
\end{align*}
\]
Additional Features in ETALIS

- Aggregates;
- Alarms;
- Consumption policies;
- Event-triggered actions;
- Dynamic updates;
- Multiplication of events;
- Garbage collection;
- Justification;
- Retractions;
- Out-of-order;
- Join;
- Projection;
- Selection
- Windows
- Logging;
- Event persistence
- …

- Many of these features are contributed to the use of EDBC Rules and the Logic Programming approach in ETALIS.
EP-SPARQL
EP-SPARQL: Toward Real-Time Semantic Web

Rapidly changing data represented as events

handles

Event Processing (EP)

Static or slowly evolving background knowledge

handle

Semantic Web technologies including SPARQL

EP + SPARQL = EP-SPARQL

• Temporal relatedness
• Semantic relatedness
• Stream reasoning
EP-SPARQL - Syntax

- Extends SPARQL to enable event-based processing by taking into account temporal situatedness of triple assertions.
- Syntactical and semantic downward-compatibility to plain SPARQL.

- **Operators:** `FILTER`, `AND`, `UNION`, `OPTIONAL`, `SEQ`, `EQUALS`, `OPTIONALSEQ`, `and` `EQUALSOPTIONAL`

- `getDURATION()` yields a literal of type `xsd:duration` giving the time interval associated to the graph pattern.

- `getSTARTTIME()` and `getENDTIME()` retrieve the time stamps of type `xsd:dateTime` of the start and end of the interval;
FILTER – restricts variable bindings to those $\langle \mu, t_{\alpha}, t_{\omega} \rangle$ for which the filter expression evaluates to $\text{true}$;

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

UNION – forms the disjunction of $\langle \mu, t_{\alpha}, t_{\omega} \rangle$ and $\langle \mu', t'_{\alpha}, t'_{\omega} \rangle$;

OPTIONAL – matches $\langle \mu, t_{\alpha}, t_{\omega} \rangle$ optionally with $\langle \mu', t'_{\alpha}, t'_{\omega} \rangle$ when the filter expression evaluates to $\text{true}$;

SEQ – joins $\langle \mu, t_{\alpha}, t_{\omega} \rangle$ and $\langle \mu', t'_{\alpha}, t'_{\omega} \rangle$ only if $\langle \mu', t'_{\alpha}, t'_{\omega} \rangle$ occurs strictly after $\langle \mu, t_{\alpha}, t_{\omega} \rangle$;

EQUALS – joins $\langle \mu, t_{\alpha}, t_{\omega} \rangle$ and $\langle \mu', t'_{\alpha}, t'_{\omega} \rangle$ if they occur simultaneously;

OPTIONALSEQ and EQUALSOPTIONAL are temporal-sensitive variants of OPTIONAL;

CONSTRUCT – generates the stream enriched by triples from possibly iterative CONSTRUCT rules. SELECT-queries get evaluated not against the pure input stream but against the enriched generated stream.
EP-SPARQL Example: Traffic Monitoring

```
SELECT ?road ?speed WHERE
{ ?road      tr:slowTrafficDue ?observ }
SEQ {{ ?road tr:slowTrafficDue ?observ }
{ ?observ rdfs:subClassOf      tr:SlowTraffic }
{ ?observ wt:speed             ?speed }}
FILTER (getDURATION() < "P1H"^^xsd:duration)
```

```
Observv_1
  rdf:type  tr:GhostDriver ;
  wt:speed  "50"^^xsd:int .

tr:GhostDriver  rdfs:subClassOf  tr:SlowTraffCause.
tr:BadWeather    rdfs:subClassOf  tr:SlowTraffCause.
tr:IceConditions rdfs:subClassOf  tr:BadWeather.

Observv_2
  rdf:type  tr:IceConditions ;
  wt:speed  "40"^^xsd:int .
```

IMPLEMENTATION AND EVALUATION
ETALIS: System Diagram

Parser

ETALIS EP rules
ETALIS tokenizer grammar and Parser
labeled_event_rules.P

ETALIS rules in internal format for complex event processing

EP Binarizer binarizer.P

Out-of-order EP
Event retraction
Justification

ETALIS binary rules

ETALIS Compiler
compiler.P, flags.P,

Event streams (etalis.P)

Prolog code

Prolog System
(SWI, YAP, Sicstus, XSB,... Prolog)

Composed events

External events

Compiler

Java interface (java_interface.P),
Prolog interpreter rules
(TR interpreter (storage.P),
Event consumption policies (executor.P),
Garbage collection (garbage_collection.P),
EP justification (justify_etalis.P),
Constraints (labeled_event_rules.P),
Logging (logging.P),
Network interface (network_tcp_interface.P),
Alan’s interval algebra, string library,
Sets, lists, counters (utilis.P)....

Execution

Auxiliary components
ETALIS Interfaces
Common Operators I

Test patterns:

\[ d(\text{Id}, X, Y, Z) : -a(\text{Id}, X) \text{ SEQ } b(\text{Id}, Y) \text{ SEQ } c(\text{Id}, Z) \].

\[ d(\text{Id}, X, Y) : -a(\text{Id}, X) \text{ SEQ } (b(\text{Id}, Y) \text{ OR } c(\text{Id}, Y)) \].
Common Operators II

Test patterns:

\[ d(Id, X, Y) : \neg \text{NOT}(c(Id, Z)).[a(Id, X), b(Id, Y)]. \]

\[ tc(X, Y) : \neg a(X, Y). \]

\[ tc(X, Y) : \neg tc(X, Z) \text{ SEQ } a(Z, Y). \]
Iterative Event Patterns

Test patterns:

delivery(Start, Start) ← shipment(Start).

delivery(From, To) ← delivery(From, PrevTo)
SEQ shipment(To)
WHERE inSupChain(From, To).

inSupChain(X, Y) :- linked(X, Y).
inSupChain(X, Z) :- linked(X, Y) AND inSupChain(Y, Z).

Supply chain
Check the path from the beginning or from the last event
Evaluation: Stream Reasoning

- Sub-class relations computed on-the-fly
- Test pattern: infer over streaming triples whether the subject of a triple is an instance of the class of concern,
- or any of its 40,080 subclasses.
Scenario: Stream Reasoning Evaluation

- A Goods Delivery system in the city of Milan
- An agent delivers goods to a certain location
- While visiting a location, the system “listens” to traffic events related to the next location
- Inaccessible routes are recomputed on-the-fly

Milan ontology was generously provided by AMAT Milano and CEFRIEL team: [http://www.larkc.eu/resources/published-data-sources/](http://www.larkc.eu/resources/published-data-sources/)
APPLICATIONS
ETALIS in Use

The Fast Flower Delivery in EPIA
- Event Processing in Action, by Opher Etzion and Peter Niblett
- An EPN implemented in ETALIS

The Drug Discovery in SYNERGY
- Collaborative work on drug design
- ETALIS: extending SOA with EDA

On the Live Measurements of Environmental Phenomena
- MesoWest sensor network
- Analysing sensor data over time and geographical space
Smart Automation

Intelligent Diagnosis in Manufacturing Networks

ETALIS + Semantic Guide
Smart Home

- **FZI Living Lab: Ambient Assisted Living**
  - Demonstrates technologies and use cases
  - Includes end-users and experts into research
  - Helps to evaluate research results and service prototypes

- **Goal:** to enable real-time reactivity in an AAL scenario

- **Use sensor networks for increasing situational awareness**
  - e.g. enable real-time reaction on dangerous situations
EP & SR Outlook

FUTURE WORK
Industry 4.0: From IoT to Event-driven and-Cloud-based Services

Event-Driven Predictive Analytics

Event predictions with ETALIS

Big scale intelligent Event Processing & Stream Reasoning

Distributed ETALIS

Event-Driven Cyber-Physical Systems

Embedded ETALIS
Credits

- People that contributed greatly to this work are:
  - Dr. Sebastian Rudolph, KIT, Germany
  - Dr. Paul Fodor, Stony Brook University, USA
  - Roland Stühmer, a PhD student, FZI, Germany
  - Ahmed Khalil Hafsi, a Master student at KIT, Germany
  - Jia Ding, a Master student at KIT, Germany
  - Vesko Georgiev, a Master student at TU Munich, Germany
References


Thank you for your attention!

http://code.google.com/p/etalis
Darko.Anicic@fzi.de