LEAD: A Formal Specification For Event Processing
EURANOVA

Our business model

EURANOVA (ENX)

- **ENX Cust. services**
- **ENX Product factory**
- **R&D EXPERTISE**

**CUSTOMERS' CHALLENGES**

**CUTTING EDGE SOLUTIONS**

COLLABORATORS

- Partners
- Customers
- Academics
- Investors
- Investors

EURANOVA in figures

- **4 main areas of expertise**: High performance & Distributed architecture, Graphs, Machine learning
- **10 years of research, development, and services in data & information management**
- **24 thesis & master thesis** produced in collaboration with 4 renowned universities
- **30 Publications** in scientific papers
- **5 open source projects** released
- **3 workshops** colocated with IEEE Int. conference on Big Data (2016 - 2017 - 2018)

Currently supporting **4 major data shifts in 3 distinct industries**
Introduction

What is Complex Event Processing?

Systems that are able to detect **interesting situations** by **correlating events** from different streams, **transforming** and **aggregating** them, and then **generating actions** are referred to as **CEP engines**.
Introduction
What is Complex Event Processing?

Systems that are able to detect **interesting situations** by correlating events from different streams, **transforming** and **aggregating** them, and then **generating actions** are referred to as **CEP engines**.

**IF** Temperature > 50 **within 3 minutes** followed by **Smoke**

Raise **Fire Alarm**
Introduction

Applications

Surveillance

Traffic congestion detection

CEP

Risk prediction

RFID processing

Market analysis

Network intrusion detection
CEP Challenges

Technical

- Performance
- Maintainability
- Scalability
CEP Challenges

**Technical**
- Performance
- Maintainability
- Scalability

**Logical**
- Ambiguous Semantics (Absence of formalisms and Selection & Consumption policies)
- Lack of Expressiveness and User-friendliness
- Missing operators (Negations, Sequences, Repetitions ... etc)
Motivation

Product Roll-up Tracking

A mobile gaming company wants to profile its applications. We assume the following four streams: **installations**, **accesses**, **artifacts bought** and **shares**; and the following four actions per each user and game and within the first 3 days from installation:

1. **Success (S)**
   \[ \geq 5, \geq 2, \geq 2 \]

2. **Middle-success & Leaving (L)**
   \[ \geq 3 \text{ and } \leq 5, 0, 0 \text{ and the user did not connect within 2 days after the last access} \]

3. **Middle-success (M)**
   \[ \geq 3, \text{ and not (S) nor (L)} \]

4. **Failure (F)**
   \[ \leq 2, 0, 0 \]
Motivation

Product Roll-up Tracking

A mobile gaming company wants to profile its applications. We assume the following four streams: **installations**, **accesses**, **artifacts bought** and **shares**; and the following four actions per each user and game and within the first 3 days from installation:

1. **Success (S)**
   \[ \geq 5, \geq 2, \geq 2 \]

2. **Middle-success & Leaving (L)**
   \[ \geq 3 \text{ and } \leq 5, 0, 0 \text{ and the user did not connect within 2 days after the last access} \]

3. **Middle-success (M)**
   \[ \geq 3, \text{ and not (S) nor (L)} \]

4. **Failure (F)**
   \[ \leq 2, 0, 0 \]

There is no CEP framework capable of formulating this problem with less than four queries, although the patterns are similar to each other and have inter-dependencies.
Contributions

1. A pattern algebra that extends the common set of operators in CEP, and defines them formally using TRIO [1, 2], a logic-based specification language aggrandized with temporal features.

2. A rule grammar that, using our pattern algebra, allows users to obtain different kinds of actions, depending on the characteristics of a matched pattern.

3. A novel logical execution plan created based on a combination of timed colored petri nets with aging tokens [3] and prioritized petri nets [4], that we believe will facilitate the deployment of this plan in the future.
Roadmap

Algebraic Operators

\[
\begin{align*}
\forall x. (P(x) \land Q(x)) & \leftrightarrow (\forall x. P(x) \land Q(x)) \\
\exists x. (P(x) \land Q(x)) & \rightarrow (\exists x. P(x)) \\
\exists x. (P(x) \lor Q(x)) & \leftrightarrow (\exists x. P(x)) \\
(\forall x. P(x)) \lor (\forall x. Q(x)) & \rightarrow (\forall x. (P(x) \lor Q(x))) \\
\exists x. \forall y. R(x, y) & \rightarrow (\forall y. \exists x. R(x, y)) \\
\neg(\exists x. P(x)) & \leftrightarrow (\forall x. \neg P(x)) \\
(\exists x. P(x)) & \leftrightarrow (\exists x. \neg P(x)) \\
\forall x. P(x) & \leftrightarrow (\forall x. P(x)) \\
\exists x. P(x) & \leftrightarrow (\exists x. P(x)) \\
(\forall x. P(x)) & \rightarrow (\forall x. (\neg P(x))) \\
(\exists x. P(x)) & \rightarrow (\exists x. (\neg P(x)))
\end{align*}
\]
Pattern Model

Event Representation & Formal Definitions

Sequence Operator

$\rightarrow (\Omega_1, \Omega_2) =_{\text{def}}$

$\forall E_{\Omega_1}, E_{\Omega_2} \subseteq E, \exists m_1 \in M_{\Omega_1}, \exists m_2 \in M_{\Omega_2}$

$\{ \text{Match}(\Omega_1 \rightarrow \Omega_2, m_1 \bowtie m_2) \leftrightarrow \}$

$[(\text{Match}(\Omega_1, m_1) \land \text{In}(\Omega_2, m_2) \land \text{Match}(\Omega_2, m_2)) \lor$

$\exists t_1 > 0((\text{Past}(\text{Match}(\Omega_1, m_1), t_1)) \land$

$\text{Past}(\text{In}(\Omega_2, m_2), t_1)) \land \text{Match}(\Omega_2, m_2)]}$

Repetition Operator

$+(\Omega, w_{acc}, w_{rt}, w_{in}) =_{\text{def}}$

$\forall w_{rt} \in W(P), \forall w_{acc}, w_{in} \in W(P_i), \forall E_{\Omega} \subseteq E, \exists M \subseteq M_{\Omega+}, \exists t$

$\{ \text{Match}(\Omega^+, \bigcup_{i \in \{1, \ldots, |\Omega^+|\}} m_i = M, w_{acc}, w_{rt}, w_{in}) \leftrightarrow$

$[\text{Past}(\text{In}(), t) \land w_{rt} \land \neg w_{in} \land \forall m_i \in M, \exists t_1 < t$

$(\text{Past}(\text{Match}(\Omega, m_i), t_1) \land \text{check}((\Omega, m_i), w_{acc})))\}$
Pattern Model

LEAD Operators

Basic Operators:
- Renaming
- Filtering

Core Operators:
- Conjunction
- Disjunction
- Negation
- Sequence
- Repetition
- Subcontext

Temporal Constraints
- Within
- Wait

Selection & Consumption Policies:
- First
- Last
- Adjacent
- Every
- All
- All ... Consume
- Repetition Max
- Repetition Min
Pattern Model

Context and Sub-context

Middle-success & Leaving (L)

- $3 \leq \text{accesses} \leq 5$
- The user did not connect within 2 days after the last access
Pattern Model

Context and Sub-context

Middle-success & Leaving (L)

- $3 \leq \text{accesses} \leq 5$
- The user did not connect within 2 days after the last access

Diagram:

- Context:
  - Installed
  - Installed + 3 days || ac::6

- Sub-context:
  - $3 \leq \text{accesses} \leq 5$
  - The user did not connect within 2 days after the last access
Pattern Model
Context and Sub-context

Middle-success & Leaving (L)

- $3 \leq \text{ accesses} \leq 5$
- The user did not connect within 2 days after the last access
Roadmap

Rule Grammar

Select <agg-func-idx>
From <stream-name>
Where <pred>
Window <window-length>: default-window-length

Select A.* B.* C.*
From StreamA A, StreamB B, StreamC C
Where A.A1 = B.A2 and A.A3 = C.A3 and B.A3 = C.A1

Algebraic Operators

c. (P(x) \land Q(x)) \iff (\forall x. P(x) \land Q(x))
\forall x. (P(x) \land Q(x)) \iff (\exists x. P(x))
\exists x. (P(x) \lor Q(x)) \iff (\exists x. P(x))
(\forall x. P(x)) \lor (\forall x. Q(x)) \iff (\forall x. (P(x) \lor Q(x)))
\exists x. (\forall y. R(x, y)) \iff (\forall y. \exists x. R(x, y))
\neg (\exists x. P(x)) \iff (\forall x. (\neg P(x))
\forall x. (\neg P(x)) \iff (\exists x. (\neg P(x))
\exists!pt. P(x) \iff (\forall xpt. (\neg P(x))
\forall xpt. (\neg P(x)) \iff (\exists xpt. (\neg P(x))
= t \rightarrow F(x)) \iff F(x)
\neg = t \rightarrow F(x)) \iff F(x)
Rule Grammar

FROM <streams>
[DEFINE <event types | event instances>]
[ENRICH <event types>]
MATCH <pattern expression>
[PARTITION BY <attributes | window>]
EMIT <actions | complex emit>
Rule Grammar

FROM <streams>
[DEFINE <event types | event instances>]
[ENRICH <event types>]
MATCH <pattern expression>
[PARTITION BY <attributes | window>]
EMIT <actions | complex emit>

<complex emit> FIRST <check clause> |
ANY <check clause> |
<check clause> <complex emit> |
(<where clause> actions)+
<where clause> WHERE conditions
Rule Grammar

Product Roll-up Tracking Rule

FROM Installations AS _in, Accesses AS _ac, ArtifactsBought AS _ab, Shares AS _sh
DEFINE TimeEvent tc(_in.event_time, _in.event_time + 3 days)
    EventType leaving(BOOLEAN leaving(FALSE))
MATCH _in Followed By (collect(_ac) terminate (!tc or count()==6) AS acs
    and collect(_ab) terminate (!tc or count()==2) AS abs
    and collect(_sh) terminate (!tc or count()==2) AS shs)
Subcontext (ac ==> acs.RANGE(3, 5) (MATCH (not _ac Within 2 days) Emit Event leaving(TRUE)))
    terminate(abs.count()>0 or shs.count()>0) AS ls
PARTITION BY _in.uid, _in.gid
CHECK FIRST
    WHERE (count(acs)>=5 and count(abs)==2 and count(shs)==2) Emit Event Success(gid)
    WHERE (count(acs)>=3)
        CHECK FIRST
            WHERE (AT LEAST 1 (ls.event_time > _in.event_time + 3 days) and count(abs)==0 and count(shs)==0)
                Emit Event Middle_Success_leaving(gid)
            WHERE (TRUE) Emit Event Middle_Success(gid) END
    WHERE (count(acs) <= 2 and count(abs)==0 and count(shs)==0) Emit Event Failure(gid) END
FROM Installations AS _in, Accesses AS _ac, ArtifactsBought AS _ab, Shares AS _sh
DEFINE TimeEvent tc(_in.event_time, _in.event_time + 3 days)
    EventType leaving(BOOLEAN leaving(FALSE))
MATCH _in Followed By (collect(_ac) terminate (!tc or count()==6) AS acs
    and collect(_ab) terminate (!tc or count()==2) AS abs
    and collect(_sh) terminate (!tc or count()==2) AS shs)
Subcontext (ac => acs.RANGE(3, 5) (MATCH (not _ac Within 2 days) Emit Event leaving(TRUE))
    terminate(abs.count()>0 or shs.count()>0) AS ls
PARTITION BY _in.uid, _in.gid
CHECK FIRST
    WHERE (count(acs)>=5 and count(abs)==2 and count(shs)==2) Emit Event Success(gid)
    WHERE (count(acs)>=3)
        CHECK FIRST
            WHERE (AT LEAST 1 (ls.event_time > _in.event_time + 3 days) and count(abs)==0 and count(shs)==0)
                Emit Event Middle_Success_Leaving(gid)
            WHERE (TRUE) Emit Event Middle_Success(gid) END
    WHERE (count(acs) <= 2 and count(abs)==0 and count(shs)==0) Emit Event Failure(gid) END
Rule Grammar

Product Roll-up Tracking Rule

```
FROM Installations AS _in, Accesses AS _ac, ArtifactsBought AS _ab, Shares AS _sh
DEFINE TimeEvent tc(_in.event_time, _in.event_time + 3 days)
    EventType leaving(BOOLEAN leaving(FALSE))
MATCH _in Followed By (collect(_ac) terminate (!tc or count()==6) AS acs
    and collect(_ab) terminate (!tc or count()==2) AS abs
    and collect(_sh) terminate (!tc or count()==2) AS shs)
Subcontext (ac ==> acs.RANGE(3, 5) (MATCH (not _ac Within 2 days) Emit Event leaving(TRUE)))
    terminate(abs.count()>0 or shs.count()>0) AS ls
PARTITION BY _in.uid, _in.gid
CHECK FIRST
WHERE (count(acs)>=5 and count(abs)==2 and count(shs)==2) Emit Event Success(gid)
WHERE (count(acs)>=3)
    CHECK FIRST
    WHERE (AT LEAST 1 (ls.event_time > _in.event_time + 3 days) and count(abs)==0 and count(shs)==0)
        Emit Event Middle_Success_Leaving(gid)
    WHERE (TRUE) Emit Event Middle_Success(gid) END
WHERE (count(acs) <= 2 and count(abs)==0 and count(shs)==0) Emit Event Failure(gid) END
```
Rule Grammar

Product Roll-up Tracking Rule

FROM Installations AS _in, Accesses AS _ac, ArtifactsBought AS _ab, Shares AS _sh
DEFINE TimeEvent tc(_in.event_time, _in.event_time + 3 days)
  EventType leaving(BOOLEAN leaving(FALSE))
MATCH _in Followed By (collect(_ac) terminate (!tc or count()==6) AS acs
    and collect(_ab) terminate (!tc or count()==2) AS abs
    and collect(_sh) terminate (!tc or count()==2) AS shs)
Subcontext (ac == acs.RANGE(3, 5) (MATCH (not _ac Within 2 days) Emit Event leaving(TRUE)))
  terminate(abs.count()>0 or shs.count()>0) AS ls
PARTITION BY _in.uid, _in.gid
CHECK FIRST
WHERE (count(acs)>=5 and count(abs)==2 and count(shs)==2) Emit Event Success(gid)
WHERE (count(acs)>=3)
  CHECK FIRST
    WHERE (AT LEAST 1 (ls.event_time > _in.event_time + 3 days) and count(abs)==0 and count(shs)==0)
      Emit Event Middle_Success_Leaving(gid)
    WHERE (TRUE) Emit Event Middle_Success(gid) END
WHERE (count(acs) <= 2 and count(abs)==0 and count(shs)==0) Emit Event Failure(gid) END
Roadmap

Rule Grammar

Select <agg-func: list>
From <stream-name>
Where <pred>
Window <window-length>: default-window-length

Select A, B, C
From StreamA, StreamB, StreamC
Where A1 = B, A2 and A, A3 = C, A3 and B, A3

Logical Execution Plan

stand by

down

failure

2k

repair

up
LOGICAL EXECUTION PLAN

Why Petri Nets?

Concurrency & Synchronization

Places, Transitions, Edges and Tokens

Probabilistic CEP
LOGICAL EXECUTION PLAN

APCPN Definition

\[ \mathbf{N} = (\Sigma, P, I, IC, OC, TT, \pi, IT, G, r_0) \]

\( \Sigma \): is a finite set of types (colours), \( \Sigma \subseteq E^n \), \( n \in \mathbb{N} \);

\( P = [p_1, p_2, ..., p_{|P|}] \): is a finite set of places, which can be either stateless, i.e. they pass tokens between transitions, or stateful, i.e. they preserve tokens in ordered structures;

\( I \): is a finite set of transitions. Transitions are either temporal guards, consumers or intermediate transitions;

\( IC \subseteq (P \times I) \): is a finite non-empty set of input arcs;

\( OC \subseteq (I \times P) \): is a finite non-empty set of output arcs;

\( TT: P \Rightarrow \Sigma \): is a color function, where each place has a single type that belongs to \( \Sigma \), and all the tokens on this place must be of the same type;

\( \pi: IC \Rightarrow N^o \): is a priority function;

\( IT: I \Rightarrow R \): is a time expression function;

\( G: I \Rightarrow boolean \): is a guard function that maps each transition \( i \in I \) to a boolean expression over all the incoming arcs \( IC(i) \subseteq IC \);

\( r_0 \subseteq R \) is an initial marking from the set of all markings \( R \).
LOGICAL EXECUTION PLAN

LEAD Rules in APCPN

LEAD Rule in APCPN
LOGICAL EXECUTION PLAN

LEAD Rules in APCPN

Source Pattern

Compact version
LOGICAL EXECUTION PLAN

LEAD Rules in APCPN

Sequence Operator:

Within Operator:

A followed by B

A within 10s from B

Two forms of sequencing events
LOGICAL EXECUTION PLAN

Product Roll-up Tracking APCPN
Status & Future Work

Current Status

- $\alpha$ DSL and compiler for LEAD rules
- $\alpha$ library built to help mapping APCPNs to the physical plan in Apache Flink

Future Work

- Discussing and implement query optimizations on both logical and physical levels
- Demonstrating the power of our approach by benchmarking the performance of LEAD CEP
- Probabilistic CEP
Summary

- Both technical and logical challenges were the reasons behind LEAD;
- 18 operators were introduced and formalized using TRIO trying to eliminate ambiguous behaviours;
- The decent set of operators and extending the capabilities of the query language were meant to increase the expressive power in CEP;
- Aging tokens prioritized colored petri nets, as a logical execution plan, is where logical optimizations take place, and our intentions for a highly performant scalable engine are shown;
- Benchmarking LEAD and probabilistic CEP are the next topics to tackle as soon as LEAD is ready and well integrated with Apache Flink.
Anas Al Bassit
Research & Development
anas.albassit@euranova.eu
4th Workshop on Real-time & Stream Analytics in Big Data & Stream Data Management

COLOCATED WITH THE 2019 IEEE INTERNATIONAL CONFERENCE ON BIG DATA

Los Angeles, CA, USA
December 9-12, 2019
History of CEP

Starting from Event Stream to data mining

2003
- Borealis - Aurora (MIT - Brown)
- STREAM (Stanford)
- Telegraph (Berkeley)
- NiagaraCQ (Univ. of Wisconsin)

2005
- Event Stream Processing
- Data Stream Processing

2006
- Twitter Storm
- Twitter Heron
- Apache S4
- IBM infoSphere (System S)
- Flink Streaming
- SAMZA

2012
- Event Stream Processing
- Data Stream Processing

2014
- Mining on Data Stream

Pattern & state matching

Event Tree
- CQ & CQL
- Tree Storage
- Centralized
- Leaf Placement

CEP
- CQL
- Pattern Match.
- Hist. Storage
- Adaptive Sched.
- Op. Placement
- Rich Operators

Oracle CEP (BEA)
- ESper
- Tibco BE
- IBM-Coral8
- Rule Core

CEP 2.0
- Event Streams
- CQ & CQL
- Pattern Match.
- Hist. Storage
- Centralized
- Op. Placement
- Rich Operators

Orange CRS AUSTRAL
- Event Tree
- Chronicle QL
- Pattern Match.
- Tree Storage
- Centralized
- Leaf Placement

Clox (Univ. of Brussels)
- Orange CRS
- AUSTRAL

Event Stream Processing & CEP
- Event Streams
- CQL
- Pattern Match.
- Hist. Storage
- Adaptive Sched.
- Op. Placement
- Rich Operators

(1) Stream processing only
(2) Pattern matching only as Cont. queries

(3) Mining on Data Stream
- ML language
- Loop aware sc.
- Cache aware sc.
- Iteration Mng
- In-Memory
History of CEP

Starting from Event Stream to data mining

2003
- Borealis - Aurora (MIT - Brown)
- STREAM (Stanford)
- Telegraph (Berkeley)
- NiagaraCQ (Univ. of Wisconsin)

Pattern Matching & Stream Processing

Design for network monitoring

2005
- Data Stream Processing
- Event Stream Processing

Pure Centralized CEP using internal Stream & DB

2006
- Event StStreams
- Generic EP
- Global Sch.

Efficient CEP Impl

2012
- Data Stream Processing
- Mining on Data Stream

Distributed CEP using automation concepts

2014
- Event Stream Processing
- Focus on event stream processing
- Focus on batch data processing using streams

(1) Stream processing only
(2) Pattern matching only as Cont. queries

2003
- Event Stream Processing & CEP
- Event Streams
- CQL
- Pattern Match.
- Hist. STorage
- Adaptive Sched.
- In-Memory

2005
- Data Str. from Storage
- Dist. Storage
- Rich Operators
- Op Placement
- Co-Loc Sched.

2006
- Data Stream Processing
- Event StStreams
- Generic EP
- Global Sch.

2012
- Data Stream Processing
- Mining on Data Stream
- Distributed CEP using automation concepts

2014
- Data Stream Processing
- Event Stream Processing
- Focus on event stream processing
- Focus on batch data processing using streams

(1) Stream processing only
(2) Pattern matching only as Cont. queries