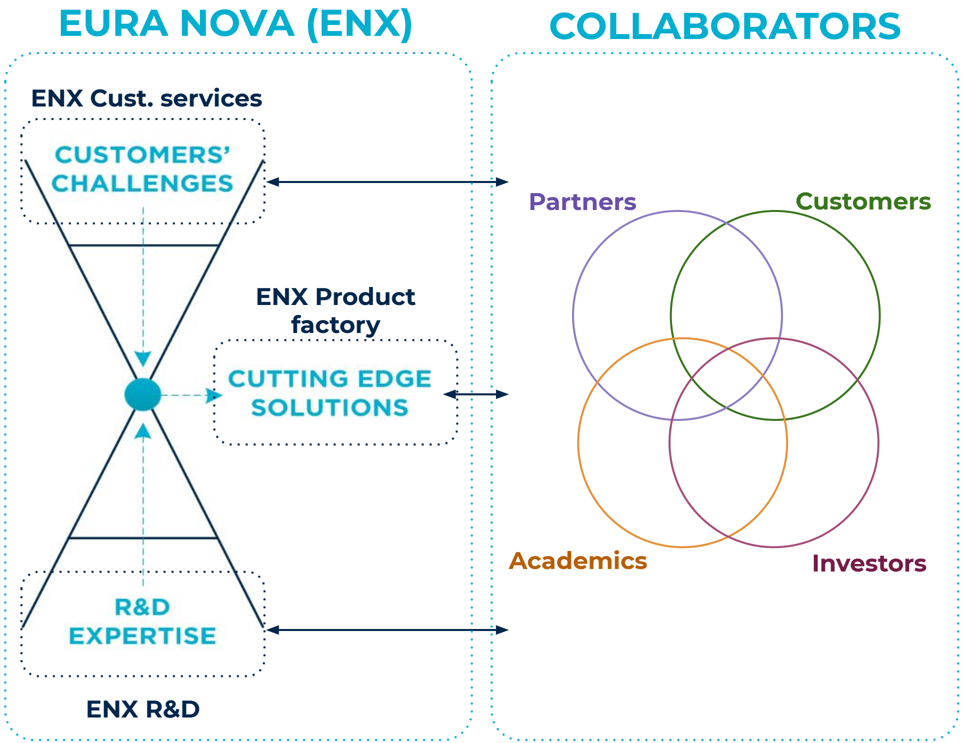
LEAD: A Formal Specification For Event Processing

Anas Al Bassit, R&D@EURANOVA

Sabri Skhiri, R&D@EURANOVA

EURA NOVA

Our business model



EURA NOVA 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

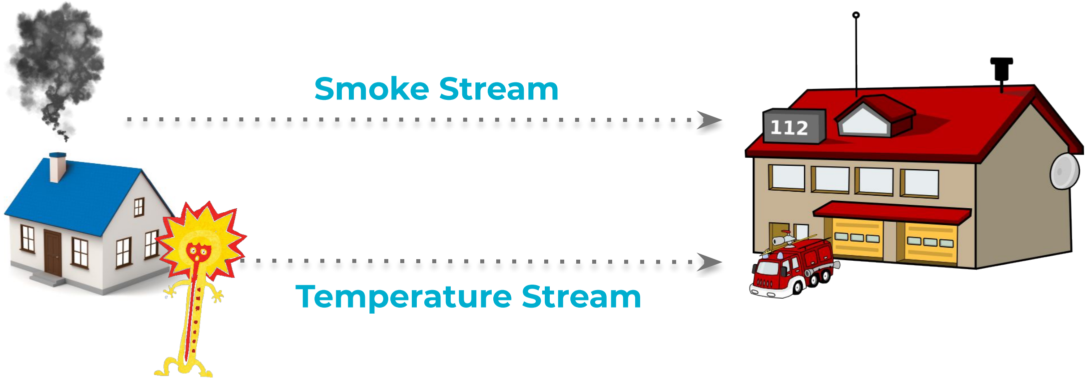
What is Complex Event Processing?

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**

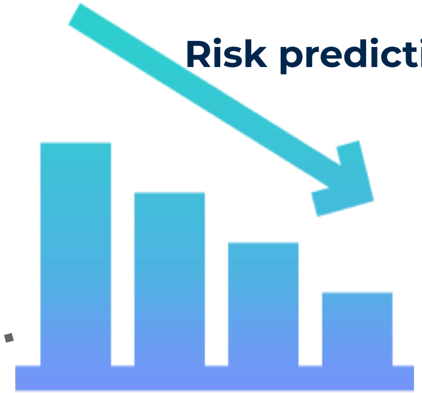
Raise **Fire Alarm**

Introduction

Applications

Surveillance

Traffic congestion detection



Risk prediction

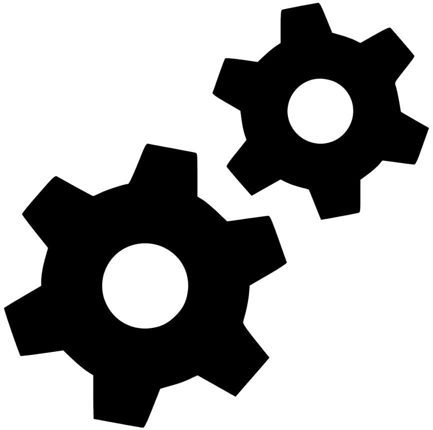
CEP

Network intrusion detection

Market analysis

RFID processing

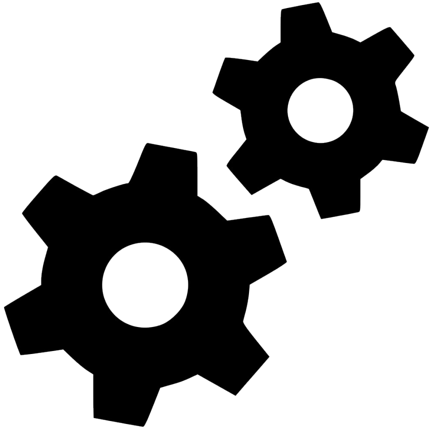
CEP Challenges

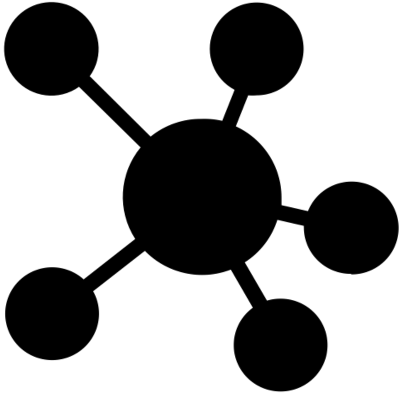


Technical

- 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**)

   ≥5, **≥2**, **≥2**

2. Middle-success & Leaving (**L**)

   ≥3 and **≤5**, **0**, **0** and the user did not connect within 2 days after the last access

3. Middle-success (**M**)

   ≥3, and not (**S**) nor (**L**)

4. Failure (**F**)

   ≤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**)

   ≥5, **≥2**, **≥2**

2. Middle-success & Leaving (**L**)

   ≥3 and **≤5**, **0**, **0** and the user did not connect within 2 days after the last access

3. Middle-success (**M**)

   ≥3, and not (**S**) nor (**L**)

4. Failure (**F**)

   ≤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.

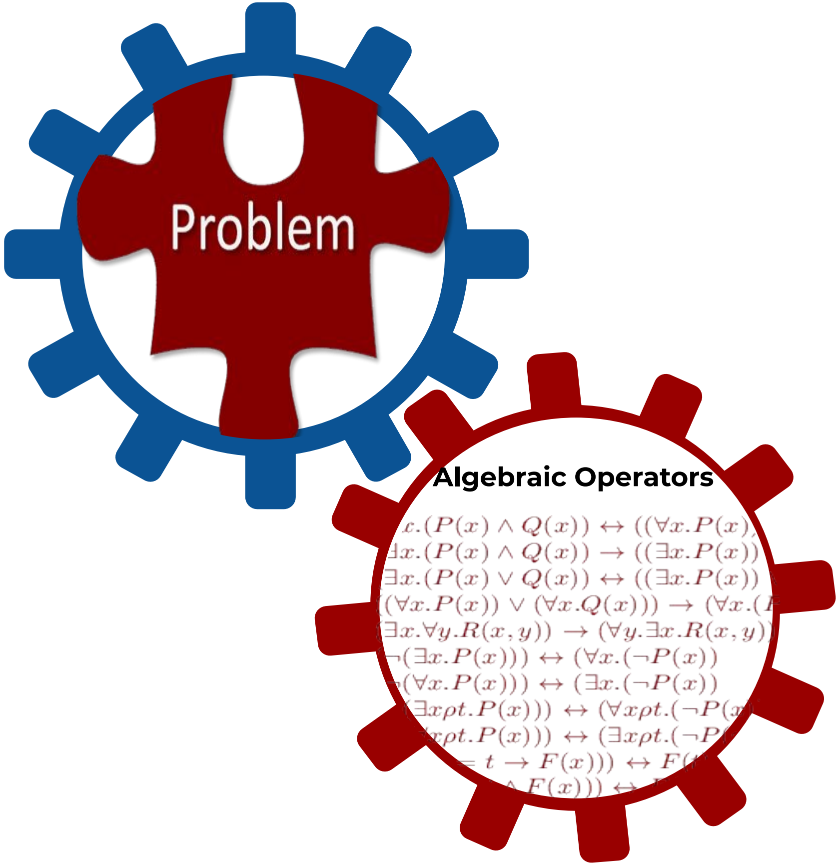
Contributions

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

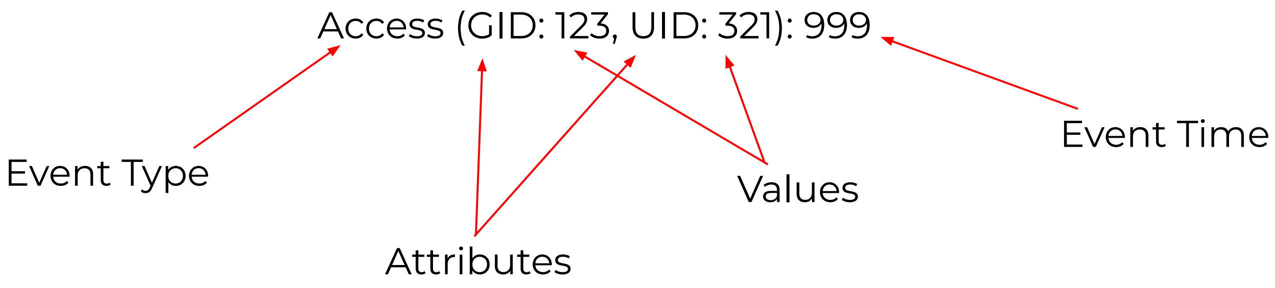
3. A novel logical execution plan created based on a combination of timed colored petri nets with aging tokens [3] and prioritized petri nets [4], that we believe will facilitate the deployment of this plan in the future.

Roadmap



Pattern Model

Event Representation & Formal Definitions



Sequence Operator

$$\rightarrow(\Omega_1, \Omega_2) =_{def}$$
$$\forall E_{\Omega_1}, E_{\Omega_2} \subseteq E, \exists m_1 \in M_{\Omega_1}, \exists m_2 \in M_{\Omega_1}$$
$$\{Match(\Omega_1 \rightarrow \Omega_2, m_1 \bowtie m_2) \leftrightarrow$$
$$[(Match(\Omega_1, m_1) \wedge In(\Omega_2, m2) \wedge Match(\Omega_2, m2)) \vee$$
$$\exists t_1 > 0((Past(Match(\Omega_1, m_1), t_1)) \wedge$$
$$Past(In(\Omega_2, m_2), t_1)) \wedge Match(\Omega_2, m_2)]\}$$

Repetition Operator

$$+(\Omega, w_{acc}, w_{rt}, w_{in}) =_{def}$$
$$\forall w_{rt} \in W(P), \forall w_{acc}, w_{in} \in W(P_i), \forall E_{\Omega} \subseteq E, \exists M \subseteq M_{\Omega+}, \exists t$$
$$\{Match(\Omega^+, \cup_{i \in \{1, \dots, |\Omega^+|\}} m_i = M, w_{acc}, w_{rt}, w_{in}) \leftrightarrow$$
$$[Past(In(), t) \wedge w_{rt} \wedge \neg w_{in} \wedge \forall m_i \in M, \exists t_1 < t$$
$$(Past(Match(\Omega, m_i), t_1) \wedge check((\Omega, m_i), w_{acc}))]\}$$

Pattern Model

LEAD Operators

Basic Operators:

- Renaming
- Filtering

Temporal Constraints

- Within
- Wait

Core Operators:

- Conjunction
- Disjunction
- Negation
- Sequence
- Repetition
- Subcontext

Selection & Consumption Policies:

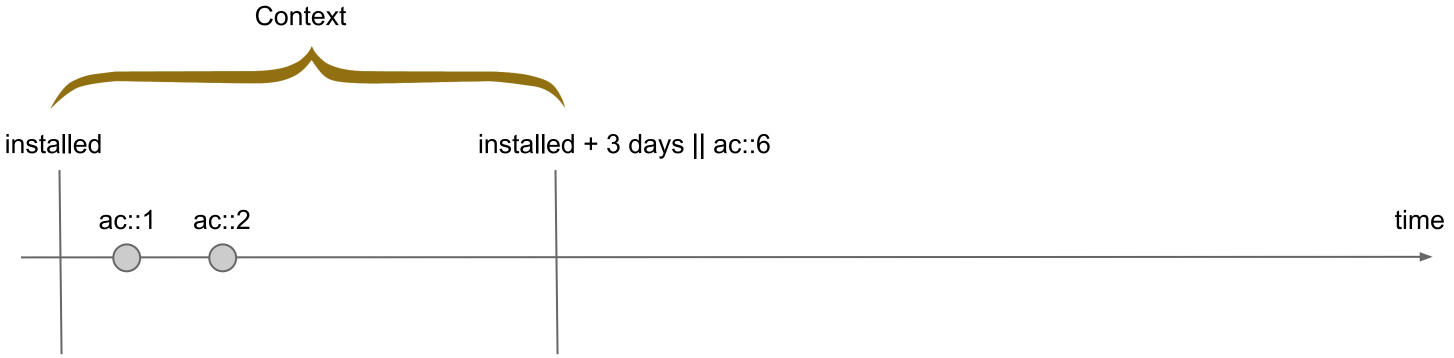
- First
- Last
- Adjacent
- Every
- All
- All … Consume
- Repetition Max
- Repetition Min

Pattern Model

Context and Sub-context

Middle-success & Leaving (**L**)

- **3≤ accesses ≤5**
- The user did not connect within 2 days after the last access

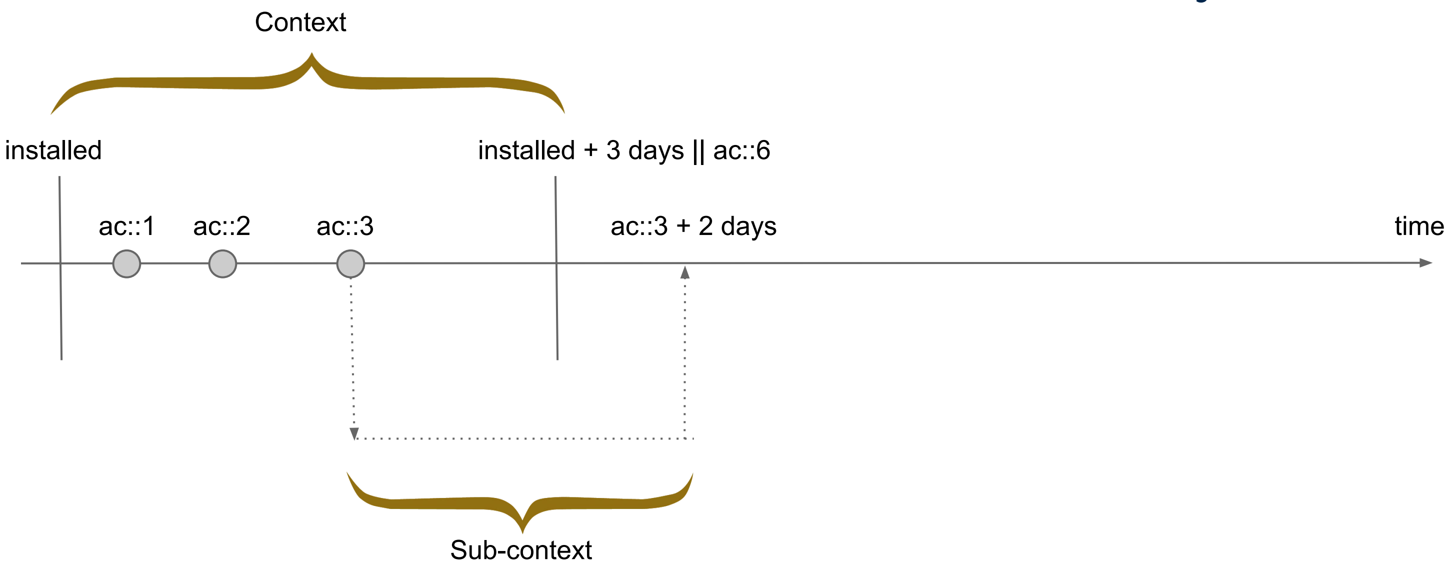


Pattern Model

Context and Sub-context

Middle-success & Leaving (L)

- **3≤ accesses ≤5**
- **The user did not connect within 2 days after the last access**

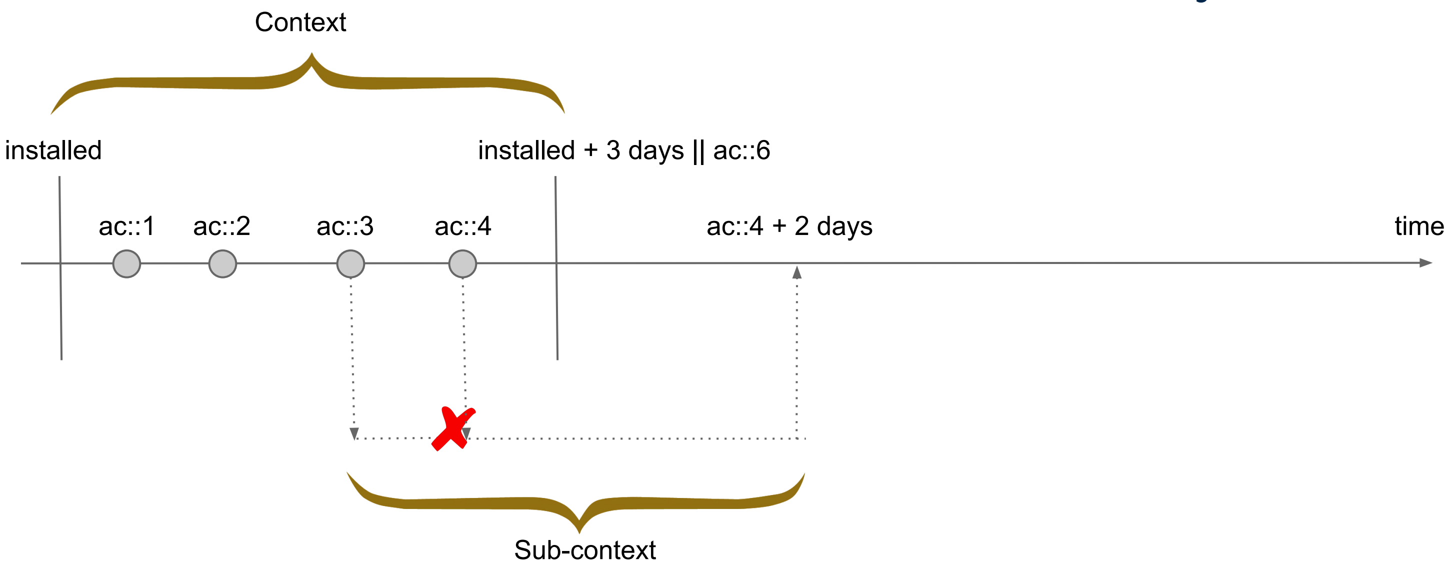


Pattern Model

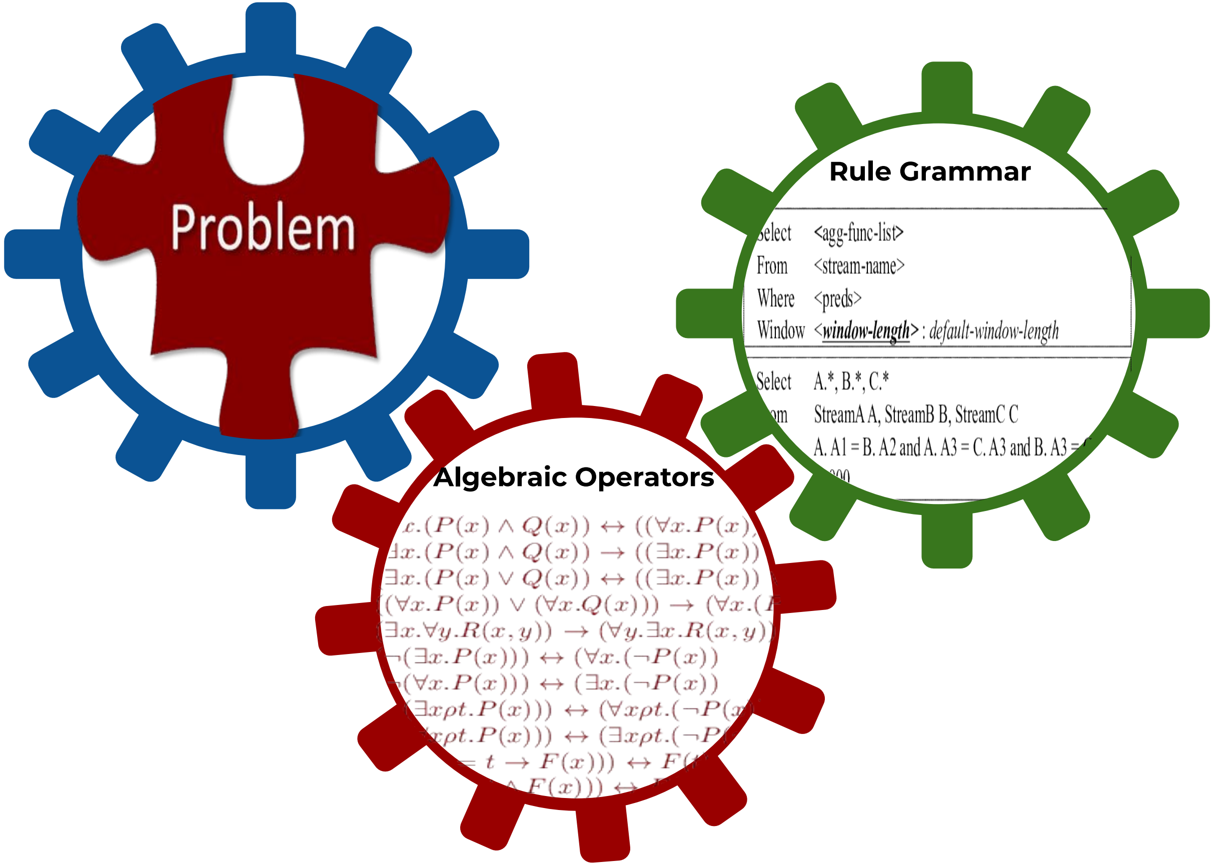
Context and Sub-context

Middle-success & Leaving (L)

- **3≤ accesses ≤5**
- **The user did not connect within 2 days after the last access**



Roadmap



Problem
Rule Grammar
Select <agg-func-list>
From <stream-name>
Where <preds>
Window <window-length> : default-window-length
Select A.*, B.*, C.*
StreamA A, StreamB B, StreamC C
Algebraic Operators

Rule Grammar

Grammar

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

Rule Grammar

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
```

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
```

Context

Sub-Context

Rule Grammar

Product Roll-up Tracking Rule

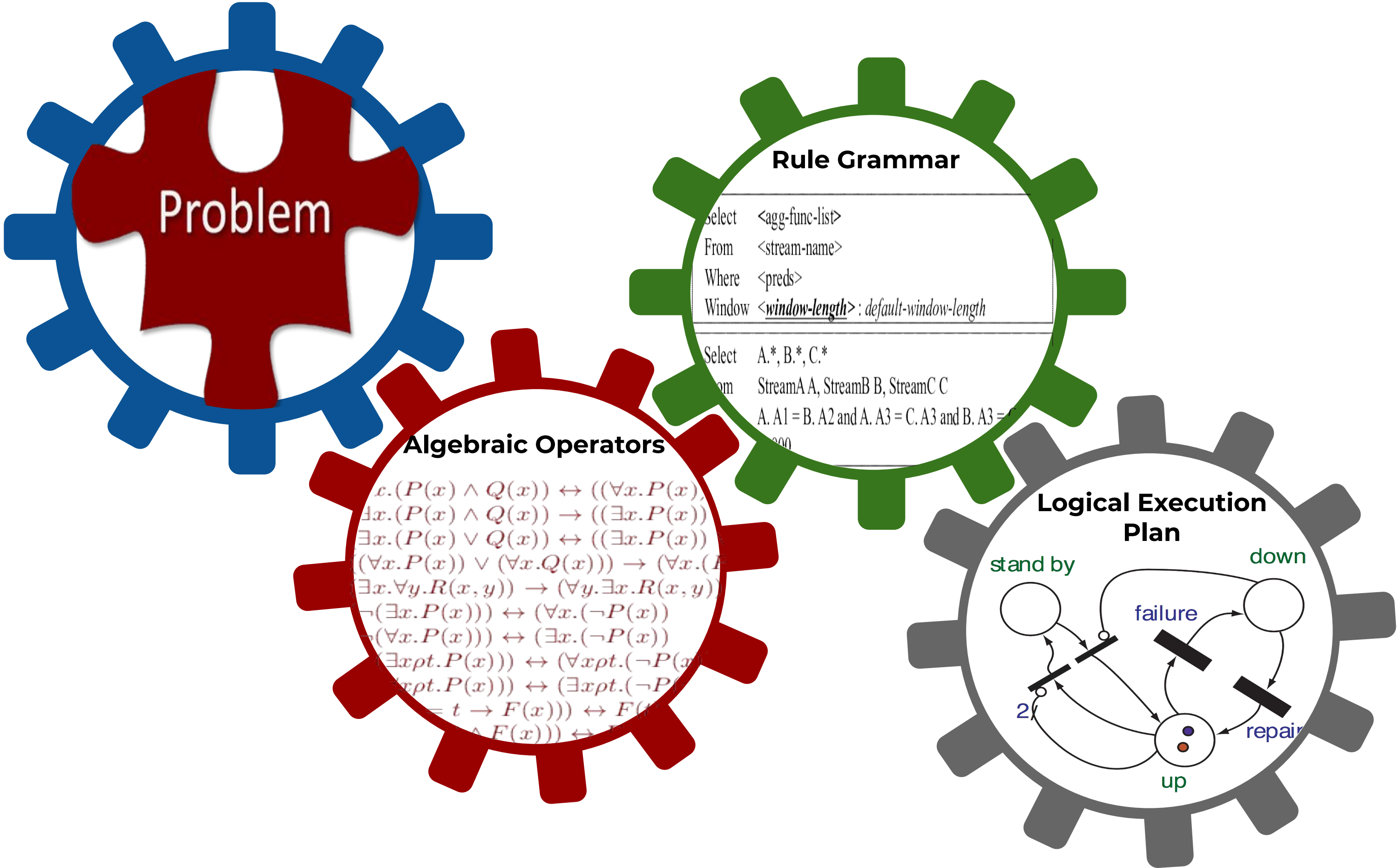
```
FROM Installations AS _in, Accesses AS _ac, ArtifactsBought AS _ab, Shares AS _sh
DEFINE TimeEvent tc(_in.event_time,  _in.event_time + 3 days)
       EventType leaving(BOOLEAN leaving(FALSE))
MATCH _in Followed By (collect(_ac) terminate (!tc or count()==6) AS acs
                      and collect(_ab) 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

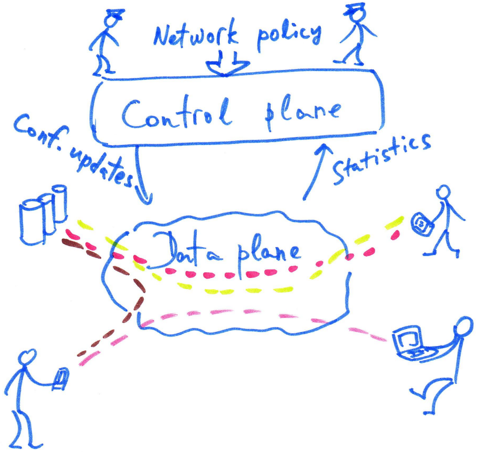
```
FROM Installations AS _in, Accesses AS _ac, ArtifactsBought AS _ab, Shares AS _sh
DEFINE TimeEvent tc(_in.event_time,  _in.event_time + 3 days)
       EventType leaving(BOOLEAN leaving(FALSE))
MATCH _in Followed By (collect(_ac) terminate (!tc or count()==6) AS acs
                      and collect(_ab) 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


Problem
Rule Grammar
From <stream-name>
Where <preds>
Window <window-length> : default-window-length
Select A.*, B.*, C.*
StreamA A, StreamB B, StreamC C
Algebraic Operators
Logical Execution Plan
stand by
down
failure
2
up

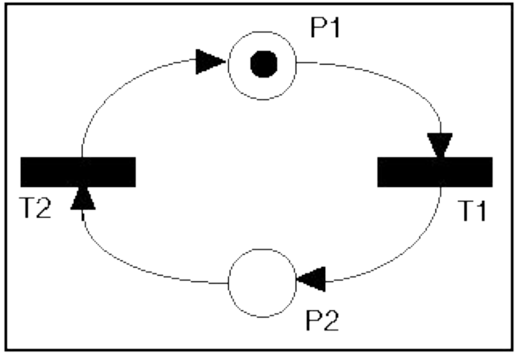
LOGICAL EXECUTION PLAN

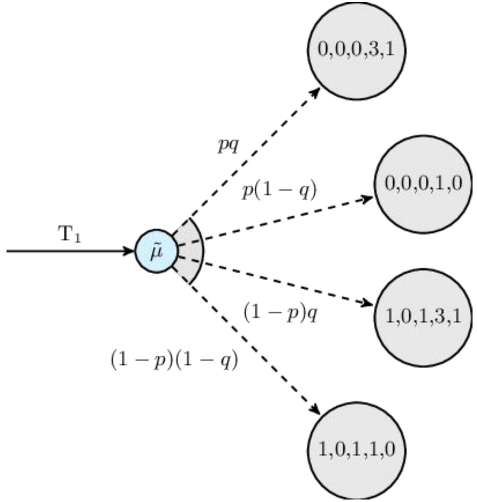
Why Petri Nets?



Concurrency & Synchronization

Places, Transitions, Edges and Tokens





Probabilistic CEP

LOGICAL EXECUTION PLAN

APCPN Definition

N = (Σ, P, I, IC, OC, TT, π, IT, G, r_0)

Σ: is a finite set of types (colours),**Σ** ⊆ $E^{[n]}$, n ∈ 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 ⊆ (**P** x **I**): is a finite non-empty set of input arcs;

OC ⊆ (**I** x **P**): is a finite non-empty set of output arcs;

TT: **P** ⇒ **Σ:** is a color function, where each place has a single type that belongs to **Σ**, and all the tokens on this place must be of the same type;

π: **IC** ⇒ **N^O** is a priority function;

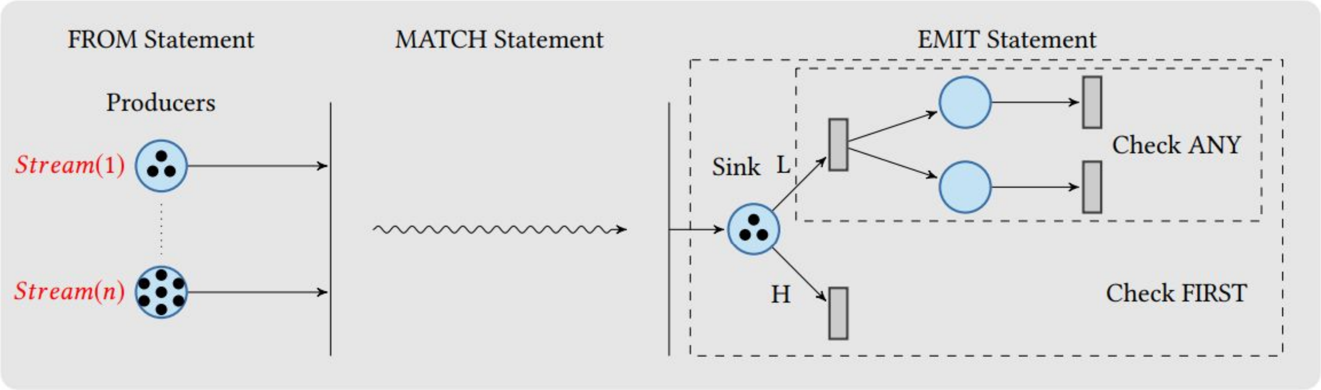
IT: **I** ⇒ **R** is a time expression function;

G: **I** ⇒ boolean is a guard function that maps each transition **i** ∈ **I** to a boolean expression over all the incoming arcs **IC(i)** ⊆ **IC**;

r_0 ∈ **R** is an initial marking from the set of all markings **R**.

LOGICAL EXECUTION PLAN

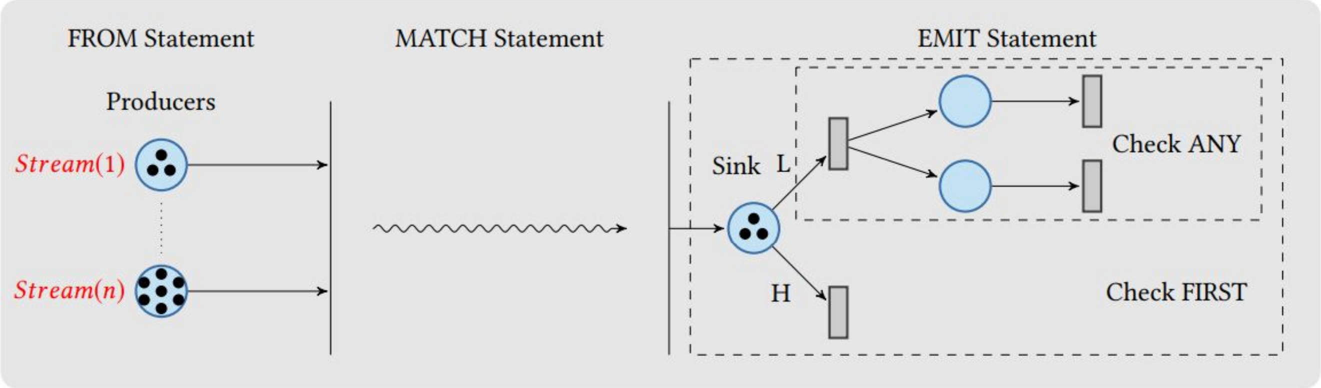
LEAD Rules in APCPN



LEAD Rule in APCPN

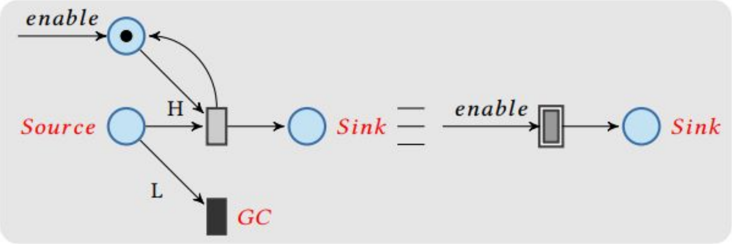
LOGICAL EXECUTION PLAN

LEAD Rules in APCPN



LEAD Rule in APCPN

Source Pattern

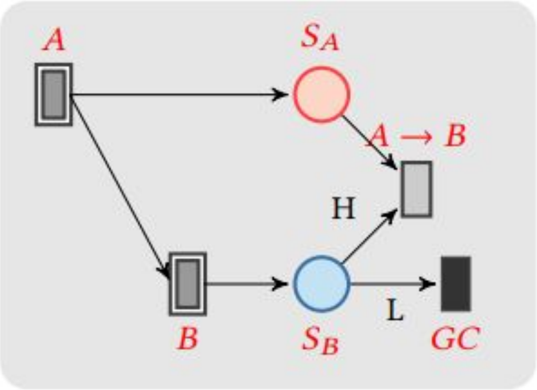


Compact version

LOGICAL EXECUTION PLAN

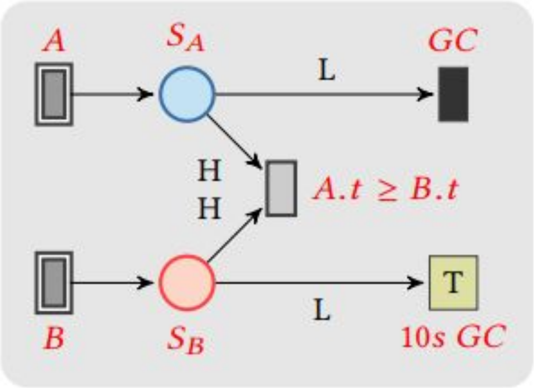
LEAD Rules in APCPN

Sequence Operator:



A followed by B

Within Operator:

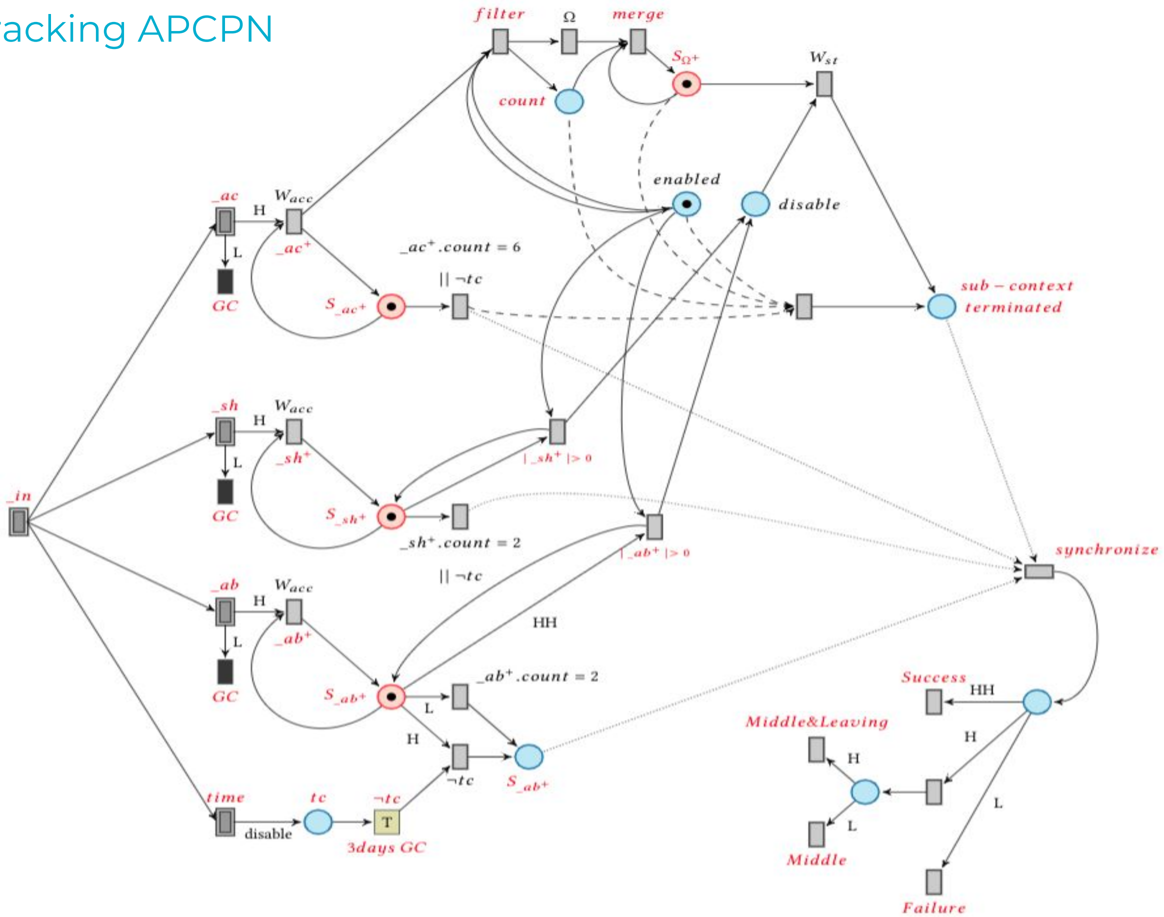


A within 10s from B

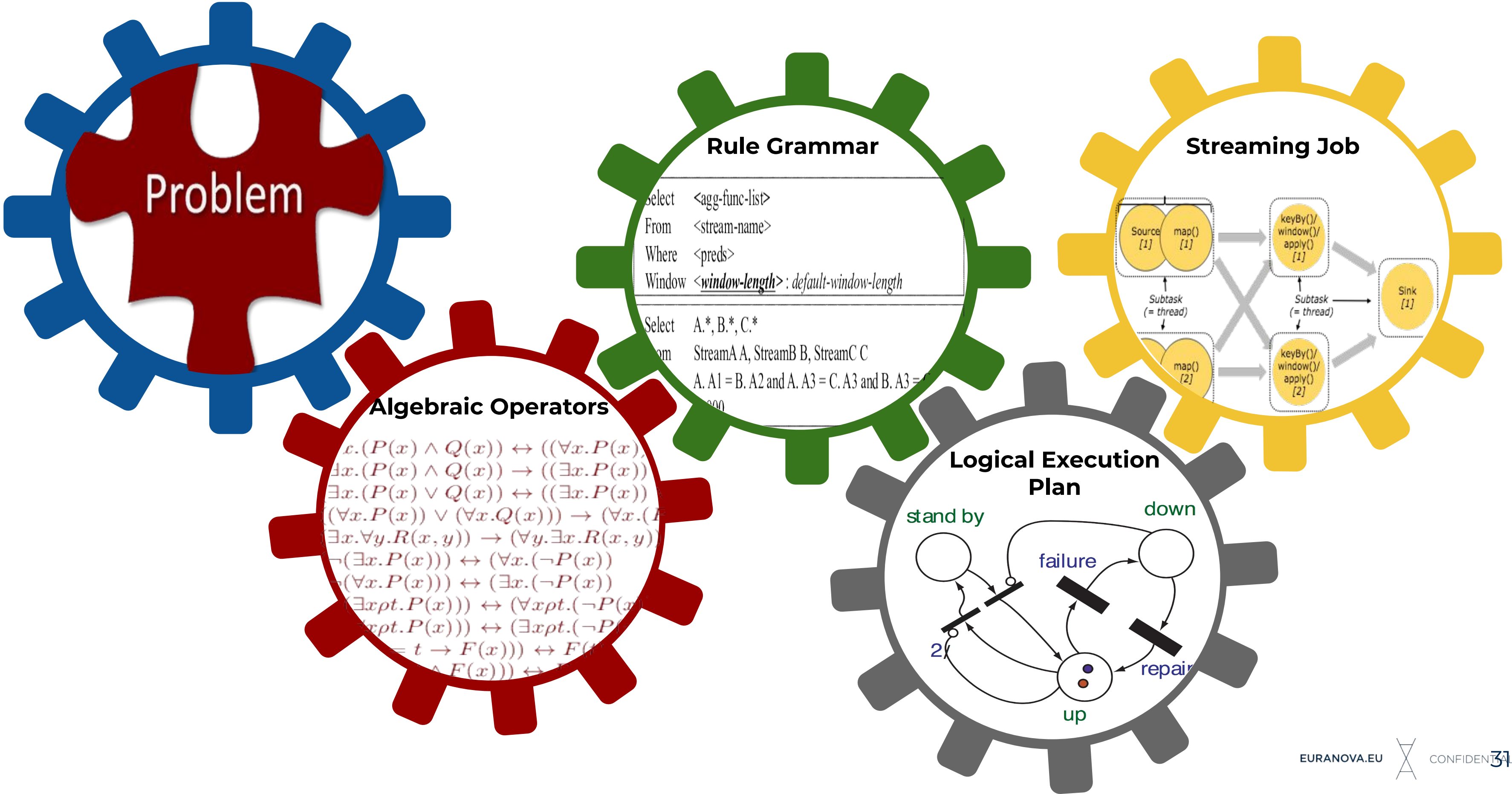
Two forms of sequencing events

LOGICAL EXECUTION PLAN

Product Roll-up Tracking APCPN



Roadmap


Problem
Rule Grammar
Select <agg-func-list>
From <stream-name>
Where <preds>
Window <window-length> : default-window-length
A.*, B.*, C.*
StreamA A, StreamB B, StreamC C
Streaming Job
Source [1]
map() [1]
keyBy()/ window()/ apply() [1]
Subtask (= thread)
Sink [1]
map() [2]
keyBy()/ window()/ apply() [2]
Algebraic Operators
Logical Execution Plan
stand by
down
failure
2
repair
up

Status & Future Work

Current Status

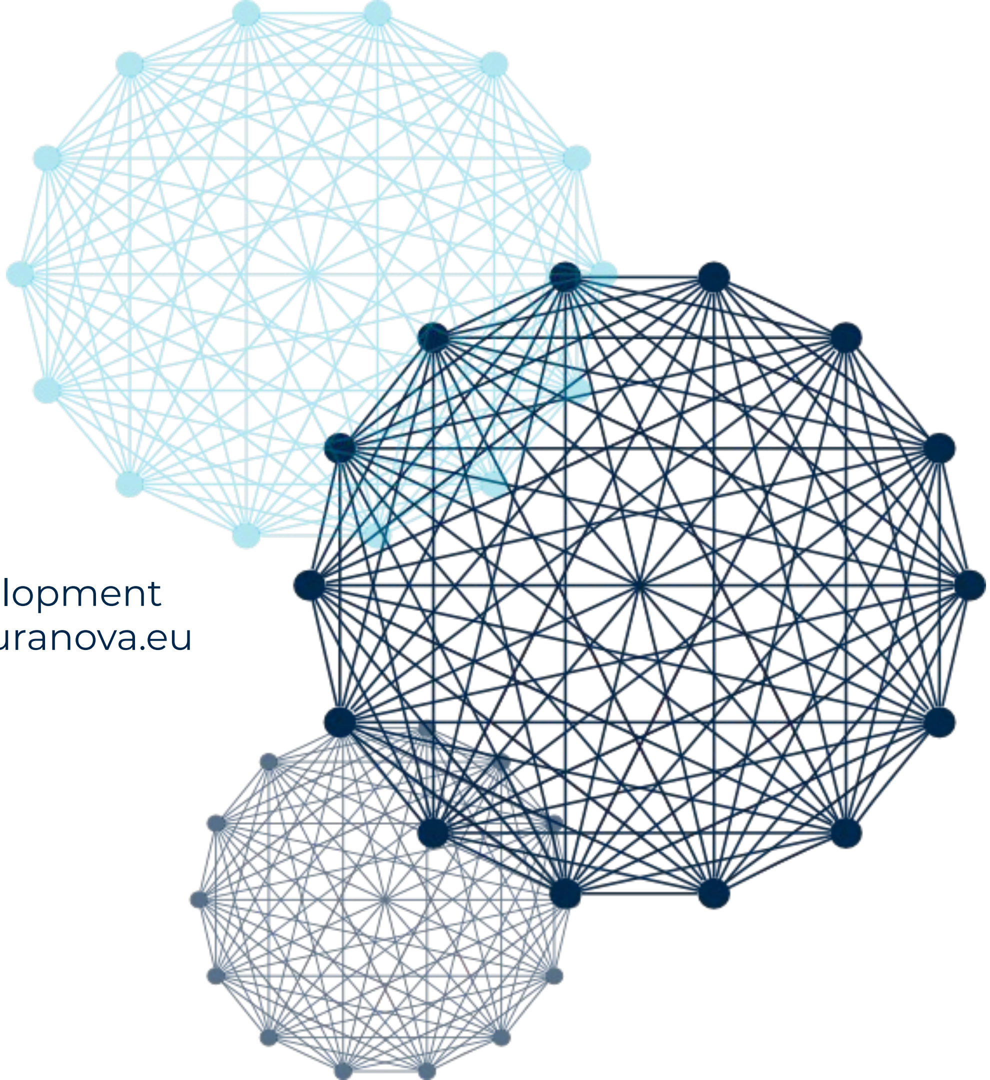
- α DSL and compiler for LEAD rules
- α library built to help mapping APCPNs to the physical plan in Apache Flink

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

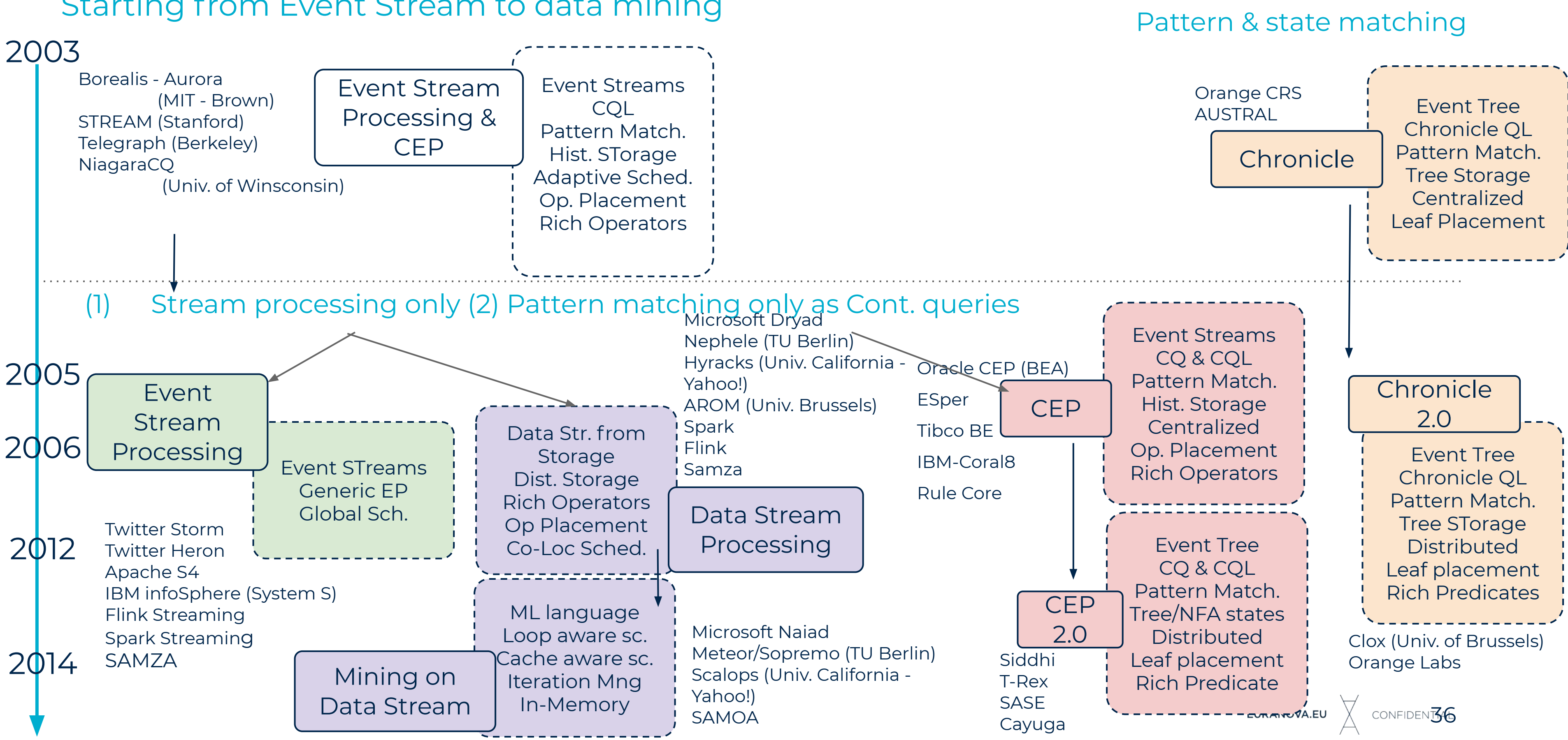
Los Angeles, CA, USA

December 9-12, 2019

ENX . Workshop

History of CEP

Starting from Event Stream to data mining



History of CEP

Starting from Event Stream to data mining