Formal Methods for Event Processing

Alexander Artikis\textsuperscript{1} and Opher Etzion\textsuperscript{2}

\textsuperscript{1}NCSR “Demokritos”, Greece
\textsuperscript{2}IBM Research, Israel

a.artikis1@iit.demokritos.gr, opher@il.ibm.com
An example to kick-off this talk:

THE LUGGAGE PERSPECTIVE:
Across the 24 largest airlines more than 5.6 million bags went missing in 2006, this is an average of 15.7 bags per 1,000 travelers. 15% of the bags are never found.

BBC News, April 4, 2007

Act: Passenger has been re-routed to another destination – send the luggage

Notify: Bag has been checked but did not reach the ULD within 20 minutes

Act: Bag has reached the wrong aircraft

Notify: Bag has been checked but did not reach the connecting flight

Some of these are reactive and some are proactive
What is an event-driven application?

I want to know about it immediately and react in the best possible way.

A suspicious account is detected whenever there are at least three big cash deposits in the last 15 days followed by transfers abroad.

- **Detect**: At least three big cash deposits in the last 15 days followed by transfers abroad
- **Derive**: A suspicious account
- **Decide**: Assign risk score
- **Do**: Open investigation
What is an event-driven application?

Did Something Happen?

Situation

It Happened

Awareness

Detect

Derive

Reaction

Decide

Do

I want to know about it immediately and react in the best possible way.
So, what are characteristics of event-driven scenarios?

A suspicious account is detected whenever there are at least three big cash deposits in the last 15 days followed by transfers abroad.

- Events trigger action
- Events influence logic for the results
- There may be multiple events whose combined content influences the results
- Temporal contexts (15 days) influence the results

The next section provides an introduction to event processing.
Tutorial Structure

- Introduction
- Temporal reasoning systems
- Event processing under uncertainty
- Ontology- and user-oriented modeling of event processing
- Open issues
Tutorial Resources

- Slides on slideshare.
Introduction

Temporal reasoning systems

Event processing under uncertainty

Ontology- and user-oriented modeling of event processing

Open issues
Event processing history

Apama acquisition By Progress

First start-ups Descendents of academic projects

Around 2000

TIBCO and Oracle announce products

2005 Streambase Coral8

Hitting the analysts hype...

2007 EPTS Established

2008 IBM Joins

Around 2020 M&A: TIBCO/Streambase Software AG/Apama

2012 New players: SAS, Yahoo, Twitter

2013 EP at the height of BIG DATA hype Cycle
Event processing in 2013

There is now an accelerated development of new event-based systems – many of the current trends in computing are event-driven.
Event Processing is being used for various reasons.

**EP Solution Segments – Business Value**

- **Information Dissemination**
  - Getting the right information in the right granularity to the right person at the right time.

- **Observation**
  - Quick observation into exceptional business behavior and notification to the appropriate people.

- **Active Diagnostics**
  - Diagnose problems based on symptoms and resolve them.

- **Predictive Processing**
  - Mitigate or eliminate predictive events.

- **Real-Time Operational**
  - Reactions to events are done as part of business transactions—achieving low latency decisions and quick reaction to threats and opportunities.

**BUSINESS**

**IT**

**CONSUMERS**
Healthcare scenario: Personalized Health Monitoring–Diagnosis + Information Dissemination

Patient is hooked up to multiple monitors (in hospital or at home) - the physician can set up rules based on multiple measurements and patient’s history when to send an alert and to whom:

Example: If fever is more than 103F, and blood pressure is up from yesterday – alert the nurse

Example: If blood pressure is up constantly within 2 days total of more than 10% alert the physician
Retail BAM Example: Smart Shelf

Individual shelf Alerts:
1. More than 25 percent of those who touched a product item in the last 2 hours – returned it back to the shelf.
2. More than 30 percent of the product items on the shelf has been removed in the last 15 minutes
3. A product item that does not belong on the shelf has been put there

Business Level Alerts:
1. A Certain product was not been taken from any shelf in the last hour
2. A certain item has been taken from the shelf, but did not reach the checkout within 1 hour

Detect
Derive
Decide
Do
Big data hype cycle 2012
What’s new?

In recent years – architectures, abstractions, and dedicated commercial products emerge to support functionality that was traditionally carried out within regular programming. For some applications it is an improvement in TCO; for others is breaking the cost-effectiveness barrier.

The analog: moving from files to DBMS
An event is anything that happens, or is contemplated as happening.

An event is a state change of some entity or actor.

An event is a detectable condition that can trigger a notification.
In daily life we often react to events..
Many times we react to combination of events within a context

The house sensor detects that the child did not arrive home within 2 hours from the scheduled end of classes for the day

I want to be notified when my own investment portfolio is down 5% since the start of the trading Day; have an agent call me when I am available, send SMS when I am in a meeting, and Email when I am out of office.
Pattern detection is one of the notable functions of event processing.
What we actually want to react to are situations.

Sometimes the situation is determined by detecting that some pattern occurred in the flowing events.

Sometimes the events can approximate or indicate with some certainty that the situation has occurred.

Toll violation

Frustrated customer
Increase in business velocity happens constantly and makes tremendous impact on business and life.
Growth in sensing sources

- RFID
- GPS
- Instrumentation
- Observation
- Need to fuse all these...
- Program emitting Events
- The Web
- Business Process State Changes
EPTS use cases survey - industries

- Banking/Financial: 25%
- Energy/Utilities: 18%
- Aerospace/Defense: 11%
- Education: 7%
- Telecom: 7%
- Transportation/Logistics: 7%
- Other: 25%

Other industries: Computer Software, Healthcare, Manufacturing, Retail and distribution, etc.
Introduction

Temporal reasoning systems

Event processing under uncertainty

Ontology- and user-oriented modeling of event processing

Open issues
Running Example

City Transport Management

IRM Demonstrator

SENSOR & GEO DATA PROCESSING

EVENT RECOGNITION

INFORMATION EXTRACTION

DATA COMMUNICATION

High-Level Events

Low-Level Events

Data

Communication

SENSOR NETWORK

RESOURCE DATA & DIGITAL MAPS

Actual Operation

Operator

Operator

Driver

Driver

Training/Debriefing

Control Centre
Event Processing for City Transport Management

- **Input**: Low-Level Events (LLE) coming from GPS, accelerometers, internal thermometers, microphones, internal cameras.
- **Output**: High-Level Events (HLE) concerning punctuality, passenger and driver safety, passenger and driver comfort, passenger satisfaction, traffic congestions, etc.
### Event Processing for City Transport Management

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>scheduled stop enter</td>
</tr>
<tr>
<td>215</td>
<td>scheduled stop leave</td>
</tr>
<tr>
<td>[215, 400]</td>
<td>abrupt acceleration</td>
</tr>
<tr>
<td>[500, 600]</td>
<td>very sharp turn</td>
</tr>
<tr>
<td>700</td>
<td>late stop enter</td>
</tr>
<tr>
<td>705</td>
<td>passenger density change to high</td>
</tr>
<tr>
<td>820</td>
<td>scheduled stop enter</td>
</tr>
<tr>
<td>815</td>
<td>passenger density change to low</td>
</tr>
</tbody>
</table>

...
### Event Processing for City Transport Management

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>200 scheduled stop enter</td>
<td></td>
</tr>
<tr>
<td>215 scheduled stop leave</td>
<td>215 punctual</td>
</tr>
<tr>
<td>[215, 400] abrupt acceleration</td>
<td>[215, 400] uncomfortable driving</td>
</tr>
<tr>
<td>[500, 600] very sharp turn</td>
<td>[500, 600] unsafe driving</td>
</tr>
<tr>
<td>700 late stop enter</td>
<td></td>
</tr>
<tr>
<td>705 passenger density</td>
<td></td>
</tr>
<tr>
<td></td>
<td>change to high</td>
</tr>
<tr>
<td>715 scheduled stop leave</td>
<td></td>
</tr>
<tr>
<td>820 scheduled stop enter</td>
<td></td>
</tr>
<tr>
<td>815 passenger density</td>
<td></td>
</tr>
<tr>
<td></td>
<td>change to low</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>
# Event Processing for City Transport Management

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>200 scheduled stop enter</td>
<td></td>
</tr>
<tr>
<td>215 scheduled stop leave</td>
<td>215 punctual</td>
</tr>
<tr>
<td>[215, 400] abrupt acceleration</td>
<td>[215, 400] uncomfortable driving</td>
</tr>
<tr>
<td>[500, 600] very sharp turn</td>
<td>[500, 600] unsafe driving</td>
</tr>
<tr>
<td>700 late stop enter</td>
<td>700 non-punctual</td>
</tr>
<tr>
<td>705 passenger density</td>
<td>since(705) reducing passenger</td>
</tr>
<tr>
<td>change to high</td>
<td></td>
</tr>
<tr>
<td>715 scheduled stop leave</td>
<td></td>
</tr>
<tr>
<td>820 scheduled stop enter</td>
<td></td>
</tr>
<tr>
<td>815 passenger density</td>
<td>[705, 815] reducing passenger</td>
</tr>
<tr>
<td>change to low</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>
Event Definitions

punctual
(Id, VehicleType)
stop enter
(Id, VehicleType, StopCode, late)
V
V
stop leave
(Id, VehicleType, StopCode, scheduled)
stop enter
(Id, VehicleType, StopCode, early)

non-punctual
(Id, VehicleType)
stop enter
(Id, VehicleType, StopCode, scheduled)
stop leave
(Id, VehicleType, StopCode, early)
stop leave
(Id, VehicleType, StopCode, late)
Event Definitions

uncomfortable driving
(Id, VehicleType)

sharp turn
(Id, VehicleType, sharp)

sharp turn
(Id, VehicleType, very_sharp)

unsafe driving
(Id, VehicleType)

abrupt acceleration
(Id, VehicleType, abrupt)

abrupt acceleration
(Id, VehicleType, very_abrupt)

abrupt deceleration
(Id, VehicleType, abrupt)

abrupt deceleration
(Id, VehicleType, very_abrupt)
Event Definitions

- compromising passenger safety \((Id, \text{VehicleType})\)
- violence \((Id, \text{VehicleType})\)
- emergency stop \((Id, \text{VehicleType})\)
- unsafe driving \((Id, \text{VehicleType})\)
- vehicle accident \((Id, \text{VehicleType})\)
A HLE can be defined as a set of events interlinked by time constraints and whose occurrence may depend on the context. This is the definition of a chronicle. Chronicle recognition systems have been used in many applications:

- Cardiac monitoring system.
- Intrusion detection in computer networks.
- Distributed diagnosis of web services.
<table>
<thead>
<tr>
<th>Predicate</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>event(E, T)</td>
<td>Event E takes place at time T</td>
</tr>
<tr>
<td>event(F: (?V1, ?V2), T)</td>
<td>An event takes place at time T changing the value of property F from ?V1 to ?V2</td>
</tr>
<tr>
<td>noevent(E, (T1,T2))</td>
<td>Event E does not take place between [T1,T2)</td>
</tr>
<tr>
<td>noevent(F: (?V1, ?V2), (T1,T2))</td>
<td>No event takes place between [T1,T2) that changes the value of property F from ?V1 to ?V2</td>
</tr>
<tr>
<td>hold(F: ?V, (T1,T2))</td>
<td>The value of property F is ?V between [T1,T2)</td>
</tr>
<tr>
<td>occurs(N,M,E,(T1,T2))</td>
<td>Event E takes place at least N times and at most M times between [T1,T2)</td>
</tr>
</tbody>
</table>
Chronicle Representation Language

chronicle punctual[?id, ?vehicle](T1) {
    event( stop_enter[?id, ?vehicle, ?stopCode, scheduled], T0 )
    event( stop_leave[?id, ?vehicle, ?stopCode, scheduled], T1 )
    T1 > T0
    end - start in [1, 2000]
}

chronicle non_punctual[?id, ?vehicle]() {
    event( stop_enter[?Id, ?vehicle, *, late], T0 )
}

chronicle punctuality_change[?id, ?vehicle, non_punctual](T1) {
    event( punctual[?id, ?vehicle], T0 )
    event( non_punctual[?id, ?vehicle], T1 )
    T1 > T0
    noevent( punctual[?id, ?vehicle], ( T0+1, T1 ) )
    noevent( non_punctual[?id, ?vehicle], ( T0+1, T1 ) )
    end - start in [1, 20000]
}
Mathematical operators in the atemporal constraints of the language are not allowed:

- cannot express that passenger safety is compromised more when a vehicle accident takes place far from a hospital or a police station.
Chronicle Representation Language

- Universal quantification is not allowed:
  - cannot express that a route is punctual if all vehicles of the route are punctual.

CRS is a purely temporal reasoning system.

It is also a very efficient and scalable system.
Each HLE definition is represented as a Temporal Constraint Network. Eg:

stop_enter [?id, ?vehicle, ?stopcode, scheduled] → [1,2000] → stop_leave [?id, ?vehicle, ?stopcode, scheduled]
Chronicle Recognition System: Consistency Checking

Compilation stage:
- Constraint propagation in the Temporal Constraint Network.
- Consistency checking.
Chronicle Recognition System: Forecasting

Recognition stage:

- Partial HLE instance evolution.
- Forward (predictive) recognition.
Chronicle Recognition System: Partial instances

HLE definition: Reduce tram endurance

A: enter tram intersection
B: abrupt deceleration
C: abrupt acceleration

A → B → C

A: [1,3]
B: [0,3]

A@1 → B@5
B@5 → C[5,8]
A@1 → B@1
Chronicle Recognition System: Partial instances

HLE definition: Reduce tram endurance

A: enter tram intersection
B: abrupt deceleration
C: abrupt acceleration

A \rightarrow [1,3] \rightarrow B \rightarrow [0,3] \rightarrow C

A\@1 \hspace{1cm} \text{time}
Chronicle Recognition System: Partial instances

HLE definition: Reduce tram endurance

A: enter tram intersection
B: abrupt deceleration
C: abrupt acceleration

A@1 \rightarrow B[2,4] \rightarrow C[5,8]

A@1 \rightarrow B[2,4]

A: enter tram intersection
B: abrupt deceleration
C: abrupt acceleration
Chronicle Recognition System: Partial instances

HLE definition: Reduce tram endurance

A: enter tram intersection
B: abrupt deceleration
C: abrupt acceleration

A@1 A@3 time

A@1 B[2,4] A@3
Chronicle Recognition System: Partial instances

HLE definition: Reduce tram endurance

A: enter tram intersection
B: abrupt deceleration
C: abrupt acceleration

A @ 1

A @ 3

A @ 1

B @ 5

duplicated

B[2,4]

A @ 1

B[4,4]

A @ 3

B[4,6]

A[1,3] → B[0,3] → C
Chronicle Recognition System: Partial instances

**HLE definition: Reduce tram endurance**

- **A**: enter tram intersection
- **B**: abrupt deceleration
- **C**: abrupt acceleration

Diagram:

- A: enter tram intersection
- B: abrupt deceleration
- C: abrupt acceleration

Time:

- A@1
- A@3
- B@5

Instances:

- A@1 → B[2,4]
- A@3 → B[4,4]
- A@3 → B[4,6]
Chronicle Recognition System: Partial instances

HLE definition: Reduce tram endurance

A: enter tram intersection
B: abrupt deceleration
C: abrupt acceleration

A@1 → B[2,4] → A@3 → B[4,4] → A@1 → B[4,6] → B@5

A → [1,3] → B → [0,3] → C

killed instance

duplicated
Chronicle Recognition System: Partial instances

HLE definition: Reduce tram endurance

A: enter tram intersection
B: abrupt deceleration
C: abrupt acceleration

A@1 \[\rightarrow\] B[2,4]

A@1 \[\rightarrow\] B[4,4]

killed instance

A@1 \[\rightarrow\] B[5,6]

A@1 \[\rightarrow\] B@5

A@1 \[\rightarrow\] C[5,8]
Chronicle Recognition System: Partial instances

HLE definition: Reduce tram endurance

A: enter tram intersection
B: abrupt deceleration
C: abrupt acceleration

A@1 \rightarrow B[2,4]
A@1 \rightarrow B[4,4]
A@1 \rightarrow B[5,6]
A@3 \rightarrow B[4,6]
B@5 \rightarrow C[5,8]

killed instance

duplicated
Chronicle Recognition System

Recognition stage — partial HLE instance management:

- In order to manage all the partial HLE instances, CRS stores them in trees, one for each HLE definition.
- Each event occurrence and each clock tick traverses these trees in order to kill some HLE instances (tree nodes) or to develop some HLE instances.
- The performance of CRS depends directly on the number of partial HLE instances
  - each tick or event $O(Kn^2)$ with $K$ number of instances, $n$ size of models.
Chronicle Recognition System: Optimisation

Several techniques have been recently developed for improving efficiency. Eg, ‘temporal focusing’:

- Distinguish between very rare events and frequent events based on a priori knowledge of the monitored application.

- Focus on the rare events: If, according to a HLE definition, a rare event should take place after the frequent event, store the incoming frequent events, and start recognition only upon the arrival of the rare event.

- In this way the number of partial HLE instances is significantly reduced.

- Example: Reduce tram endurance

![Diagram with nodes A, B, and C connected by edges with time intervals [1,3] for A to B, [0,3] for B to C, and annotations A: enter tram intersection, B: abrupt deceleration, C: abrupt acceleration]
Chronicle Recognition System: Summary

- One of the earliest and most successful formal event processing systems.
- Very efficient and scalable event processing.
- But:
  - It is a purely temporal reasoning system.
  - It does not handle uncertainty.
Event Calculus

- A logic programming language for representing and reasoning about events and their effects.
- Key components:
  - event (typically instantaneous).
  - fluent: a property that may have different values at different points in time.
- Built-in representation of inertia:
  - $F = V$ holds at a particular time-point if $F = V$ has been *initiated* by an event at some earlier time-point, and not *terminated* by another event in the meantime.
HLE Definitions in the Event Calculus

HLE definition:

\[
punctuality(ID) = \text{non	extunderscore punctual \ initiated \ iff}
\]
\[
\text{enter	extunderscore stop}(ID, StopCode, \text{late}) \ \text{happens} \ \text{or}
\]
\[
\text{leave	extunderscore stop}(ID, StopCode, \text{early}) \ \text{happens}
\]

\[
punctuality(ID) = \text{non	extunderscore punctual \ terminatedAt \ T \ iff}
\]
\[
\text{enter	extunderscore stop}(ID, StopCode, \text{scheduled}) \ \text{happensAt} \ \ T',
\]
\[
\text{leave	extunderscore stop}(ID, StopCode, \text{scheduled}) \ \text{happensAt} \ \ T
\]

HLE recognition:

- \[ punctuality(ID) = \text{non	extunderscore punctual \ holdsFor} \ \ I \]
HLE Definitions in the Event Calculus

HLE definition:

\[
\text{driving}_\text{quality}(ID) = \text{low} \iff \\
\text{punctuality}(ID) = \text{non\_punctual} \text{ or} \\
\text{driving}_\text{style}(ID) = \text{unsafe}
\]

Compiled HLE definition:

\[
\text{driving}_\text{quality}(ID) = \text{low holdsFor } l_1 \cup l_2 \iff \\
\text{punctuality}(ID) = \text{non\_punctual holdsFor } l_1, \\
\text{driving}_\text{style}(ID) = \text{unsafe holdsFor } l_2
\]
HLE Definitions in the Event Calculus

HLE definition:

\[
\begin{align*}
\text{driving}_\text{quality}(ID) &= \text{medium} \text{ iff } \\
\text{punctuality}(ID) &= \text{punctual}, \\
\text{driving}_\text{style}(ID) &= \text{uncomfortable}
\end{align*}
\]

Compiled HLE definition:

\[
\begin{align*}
\text{driving}_\text{quality}(ID) &= \text{medium} \text{ holdsFor } I_1 \cap I_2 \text{ iff } \\
\text{punctuality}(ID) &= \text{punctual} \text{ holdsFor } I_1, \\
\text{driving}_\text{style}(ID) &= \text{uncomfortable} \text{ holdsFor } I_2
\end{align*}
\]
HLE Definitions in the Event Calculus

HLE definition:

\[ \text{driving \_quality(ID)} = \text{high} \quad \text{iff} \]
\[ \text{punctuality(ID)} = \text{punctual}, \]
\[ \text{driving \_style(ID)} \neq \text{unsafe}, \]
\[ \text{driving \_style(ID)} \neq \text{uncomfortable} \]

Compiled HLE definition:

\[ \text{driving \_quality(ID)} = \text{high} \quad \text{holdsFor} \quad I_1 \setminus (I_2 \cup I_3) \quad \text{iff} \]
\[ \text{punctuality(ID)} = \text{punctual} \quad \text{holdsFor} \quad I_1, \]
\[ \text{driving \_style(ID)} = \text{unsafe} \quad \text{holdsFor} \quad I_2, \]
\[ \text{driving \_style(ID)} = \text{uncomfortable} \quad \text{holdsFor} \quad I_3 \]
Run-Time Event Processing

Real-time decision-support in the presence of:

- Very large LLE streams.
- Non-sorted LLE streams.
- LLE revision.
- Very large HLE numbers.
Event Calculus: Run-Time Event Processing
Event Calculus: Run-Time Event Processing
Event Calculus: Summary

- Complex temporal representation.
  - Succinct representation $\rightarrow$ code maintenance.
  - Intuitive representation $\rightarrow$ facilitates interaction with domain experts unfamiliar with programming.

- Complex atemporal representation.
  - The full power of logic programming is available.

- Very efficient reasoning.
  - Even in the absence of input data filtering.
  - Even when input data arrive with a delay and are revised.

- But:
  - The Event Calculus has to deal with uncertainty.
- Introduction
- Temporal reasoning systems
- Event processing under uncertainty
- Ontology- and user-oriented modeling of event processing
- Open issues
Common Problems of Event Processing

- Limited dictionary of LLE and context variables
- Incomplete LLE stream
- Erroneous LLE detection
- Inconsistent HLE annotation
- Inconsistent LLE annotation

Therefore, an adequate treatment of uncertainty is required.
Logic-based models & Probabilistic models

- Logic-based models:
  - Very expressive with formal declarative semantics
  - Directly exploit background knowledge
  - Trouble with uncertainty

- Probabilistic graphical models:
  - Handle uncertainty
  - Lack of a formal representation language
  - Difficult to model complex events
  - Difficult to integrate background knowledge
Can these approaches combined?

Research communities that try combine these approaches:

- Probabilistic (Inductive) Logic Programming
- Statistical Relational Learning

How?

- Logic-based approaches incorporate statistical methods
- Probabilistic approaches learn logic-based models
A Probabilistic Logic Programming language.

Allows for independent ‘probabilistic facts’ `prob::fact`.

`Prob` indicates the probability that `fact` is part of a possible world.

Rules are written as in classic Prolog.

The probability of a query $q$ imposed on a ProbLog database (success probability) is computed by the following formula:

$$P_s(q) = P\left( \bigvee_{e \in \text{Proofs}(q)} \bigwedge_{f_i \in e} f_i \right)$$
## Event Processing using ProbLog

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>200 0.45::scheduled stop enter</td>
<td></td>
</tr>
<tr>
<td>215 0.80::scheduled stop leave   215 0.42::punctual</td>
<td></td>
</tr>
<tr>
<td>700 0.89::late stop enter</td>
<td></td>
</tr>
<tr>
<td>705 0.82::passenger density</td>
<td>change to high</td>
</tr>
<tr>
<td>715 0.67::scheduled stop leave</td>
<td></td>
</tr>
<tr>
<td>820 0.53::scheduled stop enter</td>
<td></td>
</tr>
<tr>
<td>815 0.33::passenger density</td>
<td>change to low</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>
Event Calculus in ProbLog

![Diagram of Event Calculus in ProbLog](image-url)
Markov Logic Networks (MLN)

- Syntax: weighted first-order logic formulas \((F_i, w_i)\).
- Semantics: \((F_i, w_i)\) represents a probability distribution over possible worlds.
- A world violating formulas becomes less probable, but not impossible.
Example definition of HLE ‘uncomfortable_driving’:

\[ w_1 \]

\[
\begin{align*}
abrupt\_movement(Id, Vehicle, T) & \leftarrow \\
abrupt\_acceleration(Id, Vehicle, T) \lor \\
abrupt\_deceleration(Id, Vehicle, T) \lor \\
sharp\_turn(Id, Vehicle, T)
\end{align*}
\]

\[ w_2 \]

\[
\begin{align*}
uncomfortable\_driving(Id, Vehicle, T_2) & \leftarrow \\
approach\_intersection(Id, Vehicle, T_1) \land \\
abrupt\_movement(Id, Vehicle, T_2) \land \\
before(T_1, T_2)
\end{align*}
\]
Markov Logic: Representation

- Weight: is a real-valued number
- Higher weight $\rightarrow$ Stronger constraint
- Hard constraints
  - Infinite weight values
  - Background knowledge
  - Axioms
- Soft constraints
  - Strong weight values: almost always true
  - Weak weight values: describe exceptions
Markov Logic: Network Construction

- Formulas are translated into clausal form
- Weights are divided equally among clauses

\[
\frac{1}{3}w_1 \quad \neg abrupt\_acceleration(Id, Vehicle, T) \lor abrupt\_movement(Id, Vehicle, T)
\]

\[
\frac{1}{3}w_1 \quad \neg abrupt\_deceleration(Id, Vehicle, T) \lor abrupt\_movement(Id, Vehicle, T)
\]

\[
\frac{1}{3}w_1 \quad \neg sharp\_turn(Id, Vehicle, T) \lor abrupt\_movement(Id, Vehicle, T)
\]

\[
w_2 \quad \neg approach\_intersection(Id, Vehicle, T_1) \lor \neg abrupt\_movement(Id, Vehicle, T_2) \lor \\
\neg before(T_1, T_2) \lor uncomfortable\_driving(Id, Vehicle, T_2)
\]
Markov Logic: Network Construction

Template that produces ground Markov network:

- Given a set of constants: detected LLE
- Ground all clauses
- Boolean nodes: ground predicates
- Each ground clause:
  - Forms a clique in the network
  - Associated with $w_i$ and a Boolean feature

$$P(X = x) = \frac{1}{Z} \exp \left( \sum_i w_i n_i(x) \right)$$

$$Z = \sum_{x \in \mathcal{X}} \exp(P(X = x))$$
Markov Logic: Network Construction

\[
\frac{1}{3} w_1 \ \neg \text{abrupt\_acceleration}(Id, \text{Vehicle}, T) \lor \text{abrupt\_movement}(Id, \text{Vehicle}, T)
\]

\[
\frac{1}{3} w_1 \ \neg \text{abrupt\_deceleration}(Id, \text{Vehicle}, T) \lor \text{abrupt\_movement}(Id, \text{Vehicle}, T)
\]

\[
\frac{1}{3} w_1 \ \neg \text{sharp\_turn}(Id, \text{Vehicle}, T) \lor \text{abrupt\_movement}(Id, \text{Vehicle}, T)
\]

\[
w_2 \ \neg \text{approach\_intersection}(Id, \text{Vehicle}, T_1) \lor \neg \text{abrupt\_movement}(Id, \text{Vehicle}, T_2) \lor
\neg \text{before}(T_1, T_2) \lor \text{uncomfortable\_driving}(Id, \text{Vehicle}, T_2)
\]

\[
\text{LLE:}
\]

- \text{abrupt\_acceleration}(tr_0, \text{tram}, 101)
- \text{approach\_intersection}(tr_0, \text{tram}, 100)
- \text{before}(100, 101)

\[
\text{Constants:}
\]

- \(T = \{100, 101\}\)
- \(Id = \{tr_0\}\)
- \(\text{Vehicle} = \{\text{tram}\}\)
For example, the clause:

\[ w_2 \neg \text{approach\_intersection}(Id, \text{Vehicle}, T_1) \lor \neg \text{abrupt\_movement}(Id, \text{Vehicle}, T_2) \lor \neg \text{before}(T_1, T_2) \lor \text{uncomfortable\_driving}(Id, \text{Vehicle}, T_2) \]

produces the following groundings:

\[ w_2 \neg \text{approach\_intersection}(tr_0, \text{tram}, 100) \lor \neg \text{abrupt\_movement}(tr_0, \text{tram}, 100) \lor \neg \text{before}(100, 100) \lor \text{uncomfortable\_driving}(tr_0, \text{tram}, 100) \]

\[ w_2 \neg \text{approach\_intersection}(tr_0, \text{tram}, 100) \lor \neg \text{abrupt\_movement}(tr_0, \text{tram}, 101) \lor \neg \text{before}(100, 101) \lor \text{uncomfortable\_driving}(tr_0, \text{tram}, 101) \]

\[ w_2 \neg \text{approach\_intersection}(tr_0, \text{tram}, 101) \lor \neg \text{abrupt\_movement}(tr_0, \text{tram}, 100) \lor \neg \text{before}(101, 100) \lor \text{uncomfortable\_driving}(tr_0, \text{tram}, 100) \]

\[ w_2 \neg \text{approach\_intersection}(tr_0, \text{tram}, 101) \lor \neg \text{abrupt\_movement}(tr_0, \text{tram}, 101) \lor \neg \text{before}(101, 101) \lor \text{uncomfortable\_driving}(tr_0, \text{tram}, 101) \]
Markov Logic: Network Construction

Diagram showing relationships between before, uncomfortable driving, abrupt movement, abrupt deceleration, abrupt acceleration, and sharp turn, with specific coordinates for each node.
Markov Logic: World state discrimination

\[ P(X = x_1) = \frac{1}{Z} \exp\left(\frac{1}{3} w_1 \cdot 2 + \frac{1}{3} w_1 \cdot 2 + \frac{1}{3} w_1 \cdot 2 + w_2 \cdot 4\right) = \frac{1}{Z} e^{2w_1 + 4w_2} \]
Markov Logic: World state discrimination

\[ P(X = x_2) = \frac{1}{2} \exp\left( \frac{1}{3} w_1 \cdot 2 + \frac{1}{3} w_1 \cdot 2 + \frac{1}{3} w_1 \cdot 2 + w_2 \cdot 3 \right) = \frac{1}{2} e^{2w_1+3w_2} \]
Event recognition involves querying about HLE
- Having a ground Markov network
- Apply standard probabilistic inference methods
- Large network with complex structure
- Infeasible inference
- MLN combine logical and probabilistic inference methods
Query: The trams that are driven in an uncomfortable manner given a LLE stream

- Query variables $Q$: HLE

\[
P(Q \mid E = e) = \frac{P(Q, E = e, H)}{P(E = e, H)}
\]
Markov Logic: Conditional inference

Query: The trams that are driven in an uncomfortable manner given a LLE stream

- Query variables $Q$: HLE
- Evidence variables $E$: LLE

$$P(Q \mid E = e) = \frac{P(Q, E = e, H)}{P(E = e, H)}$$
Markov Logic: Conditional inference

Query: The trams that are driven in an uncomfortable manner given a LLE stream

- Query variables $Q$: HLE
- Evidence variables $E$: LLE
- Hidden variables $H$

$$P(Q \mid E = e) = \frac{P(Q, E = e, H)}{P(E = e, H)}$$
Markov Logic: Conditional inference

- Efficiently approximated with sampling
- Markov Chain Monte Carlo (MCMC): e.g. Gibbs sampling
- Random walks in state space
- Reject all states where $E = e$ does not hold
Markov Logic: Deterministic dependencies

- MCMC is a pure statistical method
- MLN combine logic and probabilistic models
- Hard constrained formulas:
  - deterministic dependences
  - isolated regions in state space
- Strong constrained formulas:
  - near-deterministic dependencies
  - difficult to cross regions
- Combination of satisfiability testing with MCMC
Event Calculus

![Event Calculus Diagram](image-url)
Event Calculus in MLN

Hard-constrained inertia rules:

2.3 \( HLE \) \textbf{initiatedAt} \( T \) if
\[ \text{[Conditions]} \]
\( \neg(HLE \text{ holdsAt } T) \) iff
\( \neg(HLE \text{ holdsAt } T-1), \)
\( \neg(HLE \text{ initiatedAt } T-1) \)

2.5 \( HLE \) \textbf{terminatedAt} \( T \) if
\[ \text{[Conditions]} \]
\( HLE \text{ holdsAt } T \) iff
\( HLE \text{ holdsAt } T-1, \)
\( \neg(HLE \text{ terminatedAt } T-1) \)
Event Calculus in MLN

Soft-constrained initiation inertia rules:

2.3 $HLE$ initiatedAt $T$ if
   
   [Conditions]

2.5 $HLE$ terminatedAt $T$ if
   
   [Conditions]

2.8 $\neg(HLE \text{ holdsAt } T)$ iff
   
   $\neg(HLE \text{ holdsAt } T-1),$
   
   $\neg(HLE \text{ initiatedAt } T-1)$
Event Calculus in MLN

Soft-constrained termination inertia rules:

2.3 \( HLE \ \text{initiatedAt} \ T \) if
[Conditions]

\( \neg (HLE \ \text{holdsAt} \ T) \) iff
\( \neg (HLE \ \text{hold}\text{sAt} \ T - 1) \),
\( \neg (HLE \ \text{initiatedAt} \ T - 1) \)

2.5 \( HLE \ \text{terminatedAt} \ T \) if
[Conditions]

2.8 \( HLE \ \text{hold}\text{sAt} \ T \) iff
\( HLE \ \text{hold}\text{sAt} \ T - 1 \),
\( \neg (HLE \ \text{termin}\text{inatedAt} \ T - 1) \)
Event Calculus in MLN

Soft-constrained termination inertia rules:

2.3 \textit{HLE} \textbf{initiatedAt} \textit{T} \textit{if} \\
\[ \text{[Conditions]} \]
\[ \neg (\textit{HLE} \text{ holdsAt} \textit{T}) \text{ iff} \]
\[ \neg (\textit{HLE} \text{ holdsAt} \textit{T} - 1), \]
\[ \neg (\textit{HLE} \text{ initiatedAt} \textit{T} - 1) \]

2.5 \textit{HLE} \textbf{terminatedAt} \textit{T} \textit{if} \\
\[ \text{[Conditions]} \]
0.8 \textit{HLE} \text{ holdsAt} \textit{T} \text{ iff} \\
\textit{HLE} \text{ holdsAt} \textit{T} - 1, \\
\neg (\textit{HLE} \text{ terminatedAt} \textit{T} - 1)
Event Processing Under Uncertainty: Summary

There are ways to deal with:
  ▶ Uncertainty in the input.
  ▶ Uncertainty in the HLE definitions.

But:
  ▶ We are still missing a framework for real-time event processing under uncertainty.
Introduction

Temporal reasoning systems

Event processing under uncertainty

Ontology- and user-oriented modeling of event processing

Open issues
Event processing: one of the major gaps is that all products in this area are geared towards IT developers.

Comprehensive user survey shows that 84% of the users wish that event rules will be defined by business users.

There is a gap.

Current models: implementation oriented
Business analysts oriented
Modeling

Chart 16: Event Rule Definers

- Business Specialist: 40%
- Business Analyst: 44%
- IT Developer: 16%
A delivery company uses independent drivers for its deliveries. Drivers are ranked based on their record in meeting delivery deadlines. The example is a dynamic event-driven update of the drivers' ranking.
How is driver ranking computed based on events?

(a) New driver (initiation) gets ranking of 10 points

(b) For every 20 assignments, if no deadline is missed, driver gets a ranking increase of 1 point

(c) For every 20 assignments, if more than 5 deadlines are missed, driver gets a ranking decrease of 1 point.
Current practice is to embed Driver Ranking in the Delivery Process Model (unnecessarily complex, tied to proprietary software, not understandable by knowledge workers).

Can we do it simpler?

The delivery Controller assigns delivery to a driver.

The driver drives to pick up the package.

The driver delivers and the customer confirms.

The delivery.

New driver initiation.
The vision: lift the center of gravity up in event-based systems from the programmer/code level to the knowledge worker/model level.

Center of gravity of the life-cycle is in the level of the execution code: Development, modification.

TODAY

Center of gravity of the life-cycle is in the level of the computing independent model: Development, modification.

TOMORROW
The concept computing principle – seeking a formal model that can bridge the gap

The application logic should be expressed by a semantically declarative, directly executable, implementation independent, and rigorously structured knowledge model.

The term was coined by Mills Davis in 2012.
There have been previous attempts to close the gap between IT and Business

Making the syntax close to natural language

Still need to think programming

Complexity in “programming in the large”

Why didn’t it work?
Moving from a process model of event-driven logic to a pure event model

Process Model

The Event Model
# The Event-driven thinking
## Request driven vs. event driven

<table>
<thead>
<tr>
<th>Question</th>
<th>Request driven</th>
<th>Event driven</th>
</tr>
</thead>
<tbody>
<tr>
<td>Why is an action be taken?</td>
<td>As a response to a specific request</td>
<td>Triggered by the fact that a situation has been detected</td>
</tr>
<tr>
<td>When is the action been taken?</td>
<td>When the request is being processed</td>
<td>Determined according to the context of a situation</td>
</tr>
<tr>
<td>What happens when request/event occurs?</td>
<td>A response is always produced</td>
<td>The event can be ignored, increment the state, trigger an internal derived event, or trigger a situation</td>
</tr>
</tbody>
</table>
What makes it difficult to express the “derive” part in procedural logic?

The event-driven vs. request-driven nature

Effectiveness and Efficiency issues

The temporal oriented Behavior

The hidden state handling
The temporal oriented behavior

- The logic is sensitive to timing of events: A delivery should be confirmed by the deadline.
- The logic is sensitive to the order of events: The winner in the bid is the first one who made the highest bid.
- Determination by timing considerations: Driver ranking increase and decrease are determined every 20 assignments.

Why? What? When?
The hidden state handling

Pattern “more than 5 deadlines are missed within 20 assignments”. A state counts the occurrences of missed delivery is hidden – only the end result is interesting.
Efficiency and effectiveness issues

It is possible to use the traditional way of inserting the event in a database, and make it a conventional data-driven application using queries and rules.

Efficient

The processing may not be efficient – many of the requests will not yield results.

Effective

The processing may not be effective – the time to react may be missed.

H owever

If the processing is periodic or by request then:
The “Structural Aspect” of the Event Model for Compute Driver Ranking

The delivery Controller assigns delivery to a driver

The driver drives to pick up the package

The driver delivers and the customer confirms the delivery

New driver initiation
The event model - basic building blocks of the formal model

TEM Concepts

TEM Glossary

TEM Principles

TEM Diagrams

TEM Logic Specification

<table>
<thead>
<tr>
<th>Row #</th>
<th>Expression</th>
<th>Start</th>
<th>End</th>
<th>Account ID</th>
<th>Event Logic</th>
<th>Conclusion</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Deposit occurrence + 3 days</td>
<td>same</td>
<td>same</td>
<td>Account ID</td>
<td>Big cash deposit</td>
<td>Occurs before Transaction transfer abroad</td>
</tr>
</tbody>
</table>
Anything is a concept

Glossary
- Events
- Facts
- States
- Actors
- IT elements

Logic
- Event Derivation Logic
- Transitions
- Goals
- Computation Logic
The Event Model for Compute Driver Ranking: “Detailed Logic Aspect” can generate code (1/4)
The Event Model for Compute Driver Ranking: “Detailed Logic Aspect” can generate code (2/4)
The Event Model for Compute Driver Ranking: “Detailed Logic Aspect” can generate code (3/4)
The Event Model for Compute Driver Ranking: “Detailed Logic Aspect” can generate code (4/4)

<table>
<thead>
<tr>
<th>Row Id</th>
<th>Pattern</th>
<th>Context</th>
<th>Conditions</th>
<th>Conclusion</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>When?</td>
<td>Group by Driver Id</td>
<td>Driver ranking is detected</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>When?</td>
<td>Group by Driver Id</td>
<td>Driver ranking is detected</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>When?</td>
<td>Group by Driver Id</td>
<td>Driver ranking is detected</td>
</tr>
</tbody>
</table>
In complex systems all the models play well together
The Conclusion

When a conclusion creates a derived event – it is denoted by the clause “is derived”

<table>
<thead>
<tr>
<th>Conclusion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frequent big cash deposits</td>
</tr>
<tr>
<td>is</td>
</tr>
<tr>
<td>Derived</td>
</tr>
</tbody>
</table>

The conclusion is a derived fact – event or stored fact
Uniqueness of EDL for conclusion

This principle reduces the number of such tables, and ease the validation

For each conclusion there is a single EDL table
A condition can assert that an event did not happen

<table>
<thead>
<tr>
<th>Regular Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transaction</td>
</tr>
<tr>
<td>is Absent</td>
</tr>
</tbody>
</table>

A condition can also count the number of occurrences of an event

<table>
<thead>
<tr>
<th>Multiple events Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count(Big cash deposit)</td>
</tr>
<tr>
<td>&gt; 3</td>
</tr>
</tbody>
</table>
The conditions (2/3)

<table>
<thead>
<tr>
<th>Regular Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>transaction cash deposit indicator</td>
</tr>
<tr>
<td>= Y</td>
</tr>
</tbody>
</table>

A condition may relate to the value of a fact, the fact can be part of an event, or a stored fact.

The relationships among conditions is “AND”.

Conditions are Assertions on events and facts.
When the relations between the conditions is “OR” – they are split into two rows, each of them is a different pattern

<table>
<thead>
<tr>
<th>Regular Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frequent big cash deposits</td>
</tr>
<tr>
<td>Frequent deposits followed by transfers abroad</td>
</tr>
<tr>
<td>Lack of account activity</td>
</tr>
</tbody>
</table>

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>is</td>
<td>Detected</td>
</tr>
<tr>
<td>is</td>
<td>Detected</td>
</tr>
<tr>
<td>is</td>
<td>Detected</td>
</tr>
</tbody>
</table>
## Context

<table>
<thead>
<tr>
<th>Context</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Expression</strong></td>
</tr>
<tr>
<td>Deposit occurrence</td>
</tr>
</tbody>
</table>

*Context consists of temporal context (When) and segmentation context (Partition By)*
Introduction

Temporal reasoning systems

Event processing under uncertainty

Ontology- and user-oriented modeling of event processing

Open issues
Challenges and open issues

In May 2010 there has been a meeting of the leading Persons in industry and academia in event processing in Dagstuhl. The conclusion: Event-driven computing will be the dominant paradigm of the future. But – there are many challenges to make it happen!

Some of the challenges:
1. Creating the event fabric
2. Make event processing more standard
3. Nature of events is uncertain
4. Extending the reactive world to become proactive
5. Reduce complexity and make it pervasive for use by non IT people

We have talked about uncertainty and some aspects of complexity – we’ll describe briefly some of the other challenges
Internet scale event fabric – the Dagstuhl challenge

Scale – quantity of producers, consumers, event processing agents, events, geographical areas…

Management and orchestration – getting the right event to the right agent/subscriber at the right granularity in the right time

With privacy and security considerations

Blackbox and whitebox
Optimizations, distributed and parallel infrastructure

Agent architecture
Event processing network
Real-time scheduling
Use of multiple channels

Security and protection
Privacy enforcement
Proactive event-driven computing is a new paradigm aimed at predicting the occurrence of problems or opportunities before they occur, and changing the course of actions to mitigate or leverage them.
Proactive event-driven computing: An example - Critical Shipment Logistics

Detect
Monitor shipment progress and various related alerts (traffic, cargo handling time at airport, carriers being late)

Predict
According to current route, the shipment will be 3 hours late and we will incur high penalty

Decide
Find alternative route which (given new condition) is faster than previous route

Act
Generate cargo reservations, reroute shipment
Dynamic event forecasting

Time Series Prediction

Graphical models

Temporal Graphical models
Why a correctness directives are required as part of the model – an example

Bid scenario- ground rules:
1. All bidders that issued a bid within the validity interval participate in the bid.
2. The highest bid wins. In the case of tie between bids, the first accepted bid wins the auction

Trace:
===Input Bid===
Bid Start 12:55:00
credit bid id=2, occurrence time=12:55:32, price=4
credit bid id=29, occurrence time=12:55:33, price=4
cash bid id=33, occurrence time=12:55:34, price=3
credit bid id=66, occurrence time=12:55:36, price=4
credit bid id=56, occurrence time=12:55:59, price=5
Bid End 12:56:00

===Winning Bid===
cash bid id=29, occurrence time=12:55:33, price=4

Race conditions:
Between events;
Between events and Window start/end
A lot of challenges relate to AI disciplines:

Statistical reasoning
Semantic models
Planning models
Knowledge representation
Knowledge elicitation

This tutorial dealt with various aspects of event processing and formal models: temporal, uncertainty and concept model.
Tutorial Resources

- Slides on slideshare.