Formal Methods for Event Processing

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

\textsuperscript{1}NCSR “Demokritos”, Greece
\textsuperscript{2}IBM Haifa Research Lab, Isreal

a.artikis@iit.demokritos.gr, opher@il.ibm.com
Event Recognition (Event Pattern Matching)

Input:
▶ LLE come from different sources/sensors.
▶ Very large amounts of input LLE.

Output:
▶ *High-level events* (HLE), i.e. collections of LLE and/or HLE that satisfy some pattern (temporal/spatial/logical constraints).
▶ Humans understand HLE easier than LLE.

Scope:
▶ Symbolic event recognition, not signal processing.
Cardiac Arrhythmia Recognition

- **Input**: electrocardiograms. E.g., P and QRS waves, representing heart activity.
- **Output**: cardiac arrhythmias.

A cardiac arrhythmia is recognised given a stream of P and QRS waves (events) that satisfy a set of temporal constraints.
Cardiac Arrhythmia Recognition

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>16338</td>
<td>qrs[normal]</td>
</tr>
<tr>
<td>17091</td>
<td>p_wave[normal]</td>
</tr>
<tr>
<td>17250</td>
<td>qrs[normal]</td>
</tr>
<tr>
<td>17952</td>
<td>p_wave[normal]</td>
</tr>
<tr>
<td>18913</td>
<td>p_wave[normal]</td>
</tr>
<tr>
<td>19066</td>
<td>qrs[normal]</td>
</tr>
<tr>
<td>19838</td>
<td>p_wave[normal]</td>
</tr>
<tr>
<td>20713</td>
<td>p_wave[normal]</td>
</tr>
<tr>
<td>20866</td>
<td>qrs[normal]</td>
</tr>
<tr>
<td>21413</td>
<td>qrs[abnormal]</td>
</tr>
<tr>
<td>21926</td>
<td>p_wave[normal]</td>
</tr>
<tr>
<td>22496</td>
<td>qrs[normal]</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
## Cardiac Arrhythmia Recognition

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>16338 qrs[normal]</td>
<td>[17091, 19066] mobitzII</td>
</tr>
<tr>
<td>17091 p_wave[normal]</td>
<td></td>
</tr>
<tr>
<td>17250 qrs[normal]</td>
<td></td>
</tr>
<tr>
<td>17952 p_wave[normal]</td>
<td></td>
</tr>
<tr>
<td>18913 p_wave[normal]</td>
<td></td>
</tr>
<tr>
<td>19066 qrs[normal]</td>
<td></td>
</tr>
<tr>
<td>19838 p_wave[normal]</td>
<td></td>
</tr>
<tr>
<td>20713 p_wave[normal]</td>
<td></td>
</tr>
<tr>
<td>20866 qrs[normal]</td>
<td></td>
</tr>
<tr>
<td>21413 qrs[abnormal]</td>
<td></td>
</tr>
<tr>
<td>21926 p_wave[normal]</td>
<td></td>
</tr>
<tr>
<td>22496 qrs[normal]</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>
## Cardiac Arrhythmia Recognition

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>77091 qrs[normal]</td>
<td></td>
</tr>
<tr>
<td>77250 p_wave[normal]</td>
<td></td>
</tr>
<tr>
<td>77952 qrs[normal]</td>
<td></td>
</tr>
<tr>
<td>78913 qrs[abnormal]</td>
<td></td>
</tr>
<tr>
<td>79066 p_wave[normal]</td>
<td></td>
</tr>
<tr>
<td>79838 qrs[normal]</td>
<td></td>
</tr>
<tr>
<td>80000 qrs[abnormal]</td>
<td></td>
</tr>
<tr>
<td>80713 p_wave[normal]</td>
<td></td>
</tr>
<tr>
<td>80866 qrs[normal]</td>
<td></td>
</tr>
<tr>
<td>81413 qrs[abnormal]</td>
<td></td>
</tr>
<tr>
<td>81926 p_wave[normal]</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>
## Cardiac Arrhythmia Recognition

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>77091 qrs[normal]</td>
<td>[78913, 81413] bigeminy</td>
</tr>
<tr>
<td>77250 p_wave[normal]</td>
<td></td>
</tr>
<tr>
<td>77952 qrs[normal]</td>
<td></td>
</tr>
<tr>
<td>78913 qrs[abnormal]</td>
<td></td>
</tr>
<tr>
<td>79066 p_wave[normal]</td>
<td></td>
</tr>
<tr>
<td>79838 qrs[normal]</td>
<td></td>
</tr>
<tr>
<td>80000 qrs[abnormal]</td>
<td></td>
</tr>
<tr>
<td>80713 p_wave[normal]</td>
<td></td>
</tr>
<tr>
<td>80866 qrs[normal]</td>
<td></td>
</tr>
<tr>
<td>81413 qrs[abnormal]</td>
<td></td>
</tr>
<tr>
<td>81926 p_wave[normal]</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>
Humpback Whale Song Recognition

- **Input**: whale sounds as song units.
- **Output**: whale songs.

A whale song is recognised given a stream of unit songs that satisfy a set of temporal constraints.
<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>[200, 400]</td>
<td>A</td>
</tr>
<tr>
<td>[400, 500]</td>
<td>B</td>
</tr>
<tr>
<td>[500, 550]</td>
<td>C</td>
</tr>
<tr>
<td>[600, 700]</td>
<td>B</td>
</tr>
<tr>
<td>[700, 800]</td>
<td>D</td>
</tr>
<tr>
<td>[800, 1000]</td>
<td>A</td>
</tr>
<tr>
<td>[1050, 1200]</td>
<td>E</td>
</tr>
<tr>
<td>[1300, 1500]</td>
<td>B</td>
</tr>
<tr>
<td>[1600, 1800]</td>
<td>E</td>
</tr>
<tr>
<td>[1800, 1900]</td>
<td>C</td>
</tr>
<tr>
<td>[1900, 2000]</td>
<td>B</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Humpback Whale Song Recognition

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>[200, 400]</td>
<td>A [200, 550]</td>
</tr>
<tr>
<td>[400, 500]</td>
<td>B [700, 1200]</td>
</tr>
<tr>
<td>[600, 700]</td>
<td>B ...</td>
</tr>
<tr>
<td>[700, 800]</td>
<td>D</td>
</tr>
<tr>
<td>[800, 1000]</td>
<td>A</td>
</tr>
<tr>
<td>[1050, 1200]</td>
<td>E</td>
</tr>
<tr>
<td>[1300, 1500]</td>
<td>B</td>
</tr>
<tr>
<td>[1600, 1800]</td>
<td>E</td>
</tr>
<tr>
<td>[1800, 1900]</td>
<td>C</td>
</tr>
<tr>
<td>[1900, 2000]</td>
<td>B</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>
Other Event Recognition Applications

Computer Networks:
- Input: TCP/IP messages.
- Output: denial of service attacks, worms.

Financial transaction monitoring:
- Input: messages exchanged between brokers and clients, brokers’ transactions.
- Output: brokers’ long-term activities.

Emergency Rescue Operations:
- Input: messages exchanged between rescue workers, information concerning water and fuel availability.
- Output: operation criticality, operation status.
Running Example I:
Event Recognition for Public Space Surveillance

SURVEILLANCE CAMERAS

SHORT-TERM ACTIVITY RECOGNITION

LONG-TERM ACTIVITY RECOGNITION

HUMAN OPERATOR
### Event Recognition for Public Space Surveillance

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>inactive((id_0))</td>
<td>(340 \ p(id_0) = (20.88, -11.90))</td>
</tr>
<tr>
<td>appear((id_0))</td>
<td>(340 \ appear(id_0))</td>
</tr>
<tr>
<td>walking((id_2))</td>
<td>(340 \ p(id_2) = (25.88, -19.80))</td>
</tr>
<tr>
<td>active((id_1))</td>
<td>(340 \ active(id_1))</td>
</tr>
<tr>
<td>(p(id_1))</td>
<td>(340 \ p(id_1) = (20.88, -11.90))</td>
</tr>
<tr>
<td>walking((id_3))</td>
<td>(340 \ walking(id_3))</td>
</tr>
<tr>
<td>(p(id_3))</td>
<td>(340 \ p(id_3) = (24.78, -18.77))</td>
</tr>
<tr>
<td>walking((id_3))</td>
<td>(380 \ walking(id_3))</td>
</tr>
<tr>
<td>(p(id_3))</td>
<td>(380 \ p(id_3) = (27.88, -9.90))</td>
</tr>
<tr>
<td>walking((id_2))</td>
<td>(380 \ walking(id_2))</td>
</tr>
<tr>
<td>(p(id_2))</td>
<td>(380 \ p(id_2) = (28.27, -9.66))</td>
</tr>
</tbody>
</table>
## Event Recognition for Public Space Surveillance

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>340 inactive($id_0$)</td>
<td>340 leaving_object($id_1$, $id_0$)</td>
</tr>
<tr>
<td>340 $p(id_0) = (20.88, -11.90)$</td>
<td></td>
</tr>
<tr>
<td>340 appear($id_0$)</td>
<td></td>
</tr>
<tr>
<td>340 walking($id_2$)</td>
<td></td>
</tr>
<tr>
<td>340 $p(id_2) = (25.88, -19.80)$</td>
<td></td>
</tr>
<tr>
<td>340 active($id_1$)</td>
<td></td>
</tr>
<tr>
<td>340 $p(id_1) = (20.88, -11.90)$</td>
<td></td>
</tr>
<tr>
<td>340 walking($id_3$)</td>
<td></td>
</tr>
<tr>
<td>340 $p(id_3) = (24.78, -18.77)$</td>
<td></td>
</tr>
<tr>
<td>380 walking($id_3$)</td>
<td></td>
</tr>
<tr>
<td>380 $p(id_3) = (27.88, -9.90)$</td>
<td></td>
</tr>
<tr>
<td>380 walking($id_2$)</td>
<td></td>
</tr>
<tr>
<td>380 $p(id_2) = (28.27, -9.66)$</td>
<td></td>
</tr>
</tbody>
</table>
### Event Recognition for Public Space Surveillance

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>340 inactive($id_0$)</td>
<td>340 leaving object($id_1$, $id_0$)</td>
</tr>
<tr>
<td>340 $p(id_0) = (20.88, -11.90)$</td>
<td>340 since(340) moving($id_2$, $id_3$)</td>
</tr>
<tr>
<td>340 appear($id_0$)</td>
<td></td>
</tr>
<tr>
<td>340 walking($id_2$)</td>
<td></td>
</tr>
<tr>
<td>340 $p(id_2) = (25.88, -19.80)$</td>
<td></td>
</tr>
<tr>
<td>340 active($id_1$)</td>
<td></td>
</tr>
<tr>
<td>340 $p(id_1) = (20.88, -11.90)$</td>
<td></td>
</tr>
<tr>
<td>340 walking($id_3$)</td>
<td></td>
</tr>
<tr>
<td>340 $p(id_3) = (24.78, -18.77)$</td>
<td></td>
</tr>
<tr>
<td>380 walking($id_3$)</td>
<td></td>
</tr>
<tr>
<td>380 $p(id_3) = (27.88, -9.90)$</td>
<td></td>
</tr>
<tr>
<td>380 walking($id_2$)</td>
<td></td>
</tr>
<tr>
<td>380 $p(id_2) = (28.27, -9.66)$</td>
<td></td>
</tr>
</tbody>
</table>
## Event Recognition for Transport & Traffic 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>late stop leave</td>
</tr>
<tr>
<td>[215, 400]</td>
<td>abrupt acceleration</td>
</tr>
<tr>
<td>[350, 600]</td>
<td>sharp turn</td>
</tr>
<tr>
<td>620</td>
<td>flow=low</td>
</tr>
<tr>
<td></td>
<td>density=high</td>
</tr>
<tr>
<td>700</td>
<td>scheduled stop enter</td>
</tr>
<tr>
<td>720</td>
<td>flow=low</td>
</tr>
<tr>
<td></td>
<td>density=average</td>
</tr>
<tr>
<td>820</td>
<td>scheduled stop leave</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>
## Event Recognition for Transport & Traffic 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>late stop leave</td>
</tr>
<tr>
<td>[215, 400]</td>
<td>abrupt acceleration</td>
</tr>
<tr>
<td>[350, 600]</td>
<td>sharp turn</td>
</tr>
<tr>
<td>620</td>
<td>flow=low</td>
</tr>
<tr>
<td></td>
<td>density=high</td>
</tr>
<tr>
<td></td>
<td>since(620) congestion</td>
</tr>
<tr>
<td>700</td>
<td>scheduled stop enter</td>
</tr>
<tr>
<td>720</td>
<td>flow=low</td>
</tr>
<tr>
<td></td>
<td>density=average</td>
</tr>
<tr>
<td>820</td>
<td>scheduled stop leave</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>
### Event Recognition for Transport & Traffic 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 late stop leave</td>
<td>since(215) non-punctual</td>
</tr>
<tr>
<td>[215, 400] abrupt acceleration</td>
<td></td>
</tr>
<tr>
<td>[350, 600] sharp turn</td>
<td>[215, 600] uncomfortable driving</td>
</tr>
<tr>
<td>620 flow=low</td>
<td>since(620) congestion</td>
</tr>
<tr>
<td>700 scheduled stop enter</td>
<td></td>
</tr>
<tr>
<td>720 flow=low</td>
<td>[620, 720] congestion</td>
</tr>
<tr>
<td>820 scheduled stop leave</td>
<td>[215, 820] non-punctual</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>
Event Recognition

Requirements:

▷ Efficient reasoning
  ▶ to support real-time decision-making in large-scale, distributed applications.

▷ Reasoning under uncertainty
  ▶ to deal with various types of noise.

▷ Automated knowledge construction
  ▶ to avoid the time-consuming, error-prone manual knowledge development.
Tutorial Structure

- Introduction.
- Temporal reasoning systems.
- Event recognition under uncertainty.
- Open issues.
Tutorial Structure

- Introduction.
- Temporal reasoning systems.
- Event recognition under uncertainty.
- Open issues.
uncomfortable driving
(Id, VehicleType)

sharp turn
(Id, VehicleType, sharp)

sharp turn
(Id, VehicleType, very_sharp)

abrupt acceleration
(Id, VehicleType, abrupt)

abrupt deceleration
(Id, VehicleType, very_abrupt)

abrupt acceleration
(Id, VehicleType, very_abrupt)

vehicle accident
(Id, VehicleType)

non-punctual
(Id, VehicleType)

reducing passenger comfort
(Id, VehicleType)

compromising passenger safety
(Id, VehicleType)

reducing passenger satisfaction
(Id, VehicleType)

abrupt deceleration
(Id, VehicleType, abrupt)

abrupt acceleration
(Id, VehicleType, abrupt)

sharp turn
(Id, VehicleType, sharp)

sharp turn
(Id, VehicleType, very_sharp)

abrupt acceleration
(Id, VehicleType, very_abrupt)

abrupt deceleration
(Id, VehicleType, very_abrupt)

vehicle accident
(Id, VehicleType)

stop enter
(Id, VehicleType, StopCode, late)

stop leave
(Id, VehicleType, StopCode, early)
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.
## Chronicle Representation Language

<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]
}
Chronicle Representation Language

- 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.
- 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:
Chronicle Recognition System: Consistency Checking

Compilation stage:

- Constraint propagation in the Temporal Constraint Network.
- Consistency checking.
Recognition stage:

- Partial HLE instance evolution.
- Forward (predictive) recognition.
HLE definition: Reduce tram endurance

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

\[ [1,3] \quad \rightarrow \quad [0,3] \quad \rightarrow \quad \text{time} \]
Chronicle Recognition System: Partial instances

HLE definition: Reduce tram endurance

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

\[ A \stackrel{1,3}{\rightarrow} B \stackrel{0,3}{\rightarrow} C \]

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

HLE definition: Reduce tram endurance

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

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

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

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

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

A@3 → B[2,4] → C
A@1 → B[4,6] → C

A@1 → [1,3] → B → [0,3] → C

A@1 → time → B@5 → C

A@1 → time → B@5 → C

A@1 → time → B@5 → C

A@1 → time → B@5 → C

A@1 → time → B@5 → C
Chronicle Recognition System: Partial instances

HLE definition: Reduce tram endurance

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

A[1,3] → B[0,3] → C

A@1 → B[2,4]
A@3 → B[4,4]
B@5 → C[5,8]

killed instance
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] → C[5,8]

A@1 → B[4,4] → C[5,8]

A@1 → B[5,6] → C[5,8]

killed instance

duplicated
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

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 recognition.
- **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:

\[ \text{meeting}(P_1, P_2) \text{ initiated} \iff \text{active}(P_1) \text{ happens,} \]
\[ \text{active}(P_2) \text{ happens,} \]
\[ \text{close}(P_1, P_2) \text{ holds} \]

\[ \text{meeting}(P_1, P_2) \text{ terminated} \text{ if} \]
\[ \text{stop(close}(P_1, P_2)) \text{ happens} \]

HLE recognition:

- \[ \text{meeting}(P_1, P_2) \text{ holdsFor } I \]
HLE Definitions in the Event Calculus

HLE definition:

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

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

HLE recognition:

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

HLE definition:

\[
\text{driving\_quality}(ID) = \text{low} \iff \\
\text{punctuality}(ID) = \text{non\_punctual} \text{ or} \\
\text{driving\_style}(ID) = \text{unsafe}
\]

Compiled HLE definition:

\[
\text{driving\_quality}(ID) = \text{low} \text{ holdsFor } I_1 \cup I_2 \iff \\
\text{punctuality}(ID) = \text{non\_punctual} \text{ holdsFor } I_1, \\
\text{driving\_style}(ID) = \text{unsafe} \text{ holdsFor } I_2
\]
HLE Definitions in the Event Calculus

HLE definition:

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

Compiled HLE definition:

\[ \text{driving}_\text{quality}(ID) = \text{medium} \text{ holdsFor } l_1 \cap l_2 \text{ iff } \]
\[ \text{punctuality}(ID) = \text{punctual} \text{ holdsFor } l_1, \]
\[ \text{driving}_\text{style}(ID) = \text{uncomfortable} \text{ holdsFor } l_2 \]
HLE Definitions in the Event Calculus

HLE definition:

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

Compiled HLE definition:

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

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 Recognition
Event Calculus: Run-Time Event Recognition

---

**Working Memory**

---

**time**

Q177  Q178  Q179  Q180  Q181  Q182

---

**Working Memory**

---

**time**

Q177  Q178  Q179  Q180  Q181  Q182
Event Calculus: Run-Time Event Recognition

[Diagram showing time-axis with annotations and timelines labeled as Q177, Q178, Q179, Q180, Q181, and Q182, along with the term "Working Memory"]
Event Calculus: Summary

- Complex temporal representation.
  - Succinct representation → code maintenance.
  - Intuitive representation → 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 recognition under uncertainty.
- Open issues.
Common Problems of Event Recognition

- 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
ProbLog

- 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 Recognition using ProbLog

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>340 0.45 :: <em>inactive</em>(\textit{id}_0)</td>
<td>340 0.41 :: <em>leaving_object</em>(\textit{id}_1, \textit{id}_0)</td>
</tr>
<tr>
<td>340 0.80 :: \textit{p}(\textit{id}_0) = (20.88, -11.90)</td>
<td>340 0.55 :: <em>moving</em>(\textit{id}_2, \textit{id}_3)</td>
</tr>
<tr>
<td>340 0.55 :: <em>appear</em>(\textit{id}_0)</td>
<td></td>
</tr>
<tr>
<td>340 0.15 :: <em>walking</em>(\textit{id}_2)</td>
<td></td>
</tr>
<tr>
<td>340 0.80 :: \textit{p}(\textit{id}_2) = (25.88, -19.80)</td>
<td></td>
</tr>
<tr>
<td>340 0.25 :: <em>active</em>(\textit{id}_1)</td>
<td></td>
</tr>
<tr>
<td>340 0.66 :: \textit{p}(\textit{id}_1) = (20.88, -11.90)</td>
<td></td>
</tr>
<tr>
<td>340 0.70 :: <em>walking</em>(\textit{id}_3)</td>
<td></td>
</tr>
<tr>
<td>340 0.46 :: \textit{p}(\textit{id}_3) = (24.78, -18.77)</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>
Event Calculus in ProbLog
To compare ProbLog-EC to Crisp-EC:

- We add noise (probabilities) in LLE:
  - Crisp-EC: LLE with probability < 0.5 are discarded.
  - ProbLog-EC: all LLE are kept with their probabilities.
- In ProbLog-EC we accept as recognised the HLE that have probability > 0.5.
Event Calculus in ProbLog: Experimental Evaluation

![Graph showing the relationship between LLE Occurrences and Noise (Gamma distribution mean).]
**Event Calculus in ProbLog: Experimental Evaluation**

![Graphs showing comparison between Crisp-EC and ProbLog-EC for Meeting and Moving events.]

**Equation:**

\[
\text{moving}(P_1, P_2) \quad \text{initiated} \quad \text{iff} \\
\text{walking}(P_1) \quad \text{happens}, \\
\text{walking}(P_2) \quad \text{happens}, \\
\text{close}(P_1, P_2) \quad \text{holds}, \\
\text{orientation}(P_1) = O_1 \quad \text{holds}, \\
\text{orientation}(P_2) = O_2 \quad \text{holds}, \\
|O_1 - O_2| < \text{threshold}
\]
Event Calculus in ProbLog: Experimental Evaluation

moving($P_1, P_2$) initiated iff
walking($P_1$) happens,
walking($P_2$) happens,
close($P_1, P_2$) holds,
orientation($P_1$) = $O_1$ holds,
orientation($P_2$) = $O_2$ holds,
$|O_1 - O_2| < \text{threshold}$
Event Calculus in ProbLog: Summary

- ProbLog-EC clearly outperforms Crisp-EC when:
  - The environment is highly noisy (LLE < 0.5) — realistic assumption in many domains,
  - there are successive initiations that allow the HLE’s probability to increase and eventually exceed the specified (0.5) threshold, and
  - the amount of probabilistic conjuncts in an initiation condition is limited.

- Note that:
  - we also need to deal with uncertainty in the HLE definitions.
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’:

\[
\begin{align*}
 \text{abrupt\_movement}(Id, \text{Vehicle}, T) & \leftarrow w_1 \quad \text{abrupt\_acceleration}(Id, \text{Vehicle}, T) \lor \\
 & \quad \text{abrupt\_deceleration}(Id, \text{Vehicle}, T) \lor \\
 & \quad \text{sharp\_turn}(Id, \text{Vehicle}, T) \\
 w_2 \quad \text{uncomfortable\_driving}(Id, \text{Vehicle}, T_2) & \leftarrow \text{approach\_intersection}(Id, \text{Vehicle}, T_1) \land \\
 & \quad \text{abrupt\_movement}(Id, \text{Vehicle}, T_2) \land \\
 & \quad \text{before}(T_1, T_2)
\end{align*}
\]
Markov Logic: Representation

- **Weight**: a real-valued number.
- **Higher weight** $\rightarrow$ **Stronger constraint**
- **Hard constraints**
  - Infinite weight values.
  - Background knowledge.
- **Soft constraints**
  - Strong weight values: almost always true.
  - Weak weight values: describe exceptions.
Formulas are translated into clausal form.

Weights are divided equally among clauses:

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

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

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

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

Template that produces ground Markov network:
- Given a set of constants from the input LLE stream
  - Ground all clauses.
- Boolean nodes: ground predicates.
- Each ground clause:
  - Forms a clique in the network.
  - Is 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

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

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

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

\( w_2 \quad \neg approach\_intersection(Id, Vehicle, T_1) \vee \neg abrupt\_movement(Id, Vehicle, T_2) \vee \\
\neg before(T_1, T_2) \vee uncomfortable\_driving(Id, Vehicle, T_2) \)

**LLE:**

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

**Constants:**

- \( T = \{100, 101\} \)
- \( Id = \{tr_0\} \)
- \( Vehicle = \{tram\} \)
For example, the clause:
\[
\begin{align*}
w_2 \quad & \neg \text{approach\_intersection}(Id, \text{Vehicle}, T_1) \lor \neg \text{abrupt\_movement}(Id, \text{Vehicle}, T_2) \lor \\
& \neg \text{before}(T_1, T_2) \lor \text{uncomfortable\_driving}(Id, \text{Vehicle}, T_2)
\end{align*}
\]

produces the following groundings:
\[
\begin{align*}
w_2 \quad & \neg \text{approach\_intersection}(tr_0, \text{tram}, 100) \lor \neg \text{abrupt\_movement}(tr_0, \text{tram}, 100) \lor \\
& \neg \text{before}(100, 100) \lor \text{uncomfortable\_driving}(tr_0, \text{tram}, 100)
\end{align*}
\]
\[
\begin{align*}
w_2 \quad & \neg \text{approach\_intersection}(tr_0, \text{tram}, 100) \lor \neg \text{abrupt\_movement}(tr_0, \text{tram}, 101) \lor \\
& \neg \text{before}(100, 101) \lor \text{uncomfortable\_driving}(tr_0, \text{tram}, 101)
\end{align*}
\]
\[
\begin{align*}
w_2 \quad & \neg \text{approach\_intersection}(tr_0, \text{tram}, 101) \lor \neg \text{abrupt\_movement}(tr_0, \text{tram}, 100) \lor \\
& \neg \text{before}(101, 100) \lor \text{uncomfortable\_driving}(tr_0, \text{tram}, 100)
\end{align*}
\]
\[
\begin{align*}
w_2 \quad & \neg \text{approach\_intersection}(tr_0, \text{tram}, 101) \lor \neg \text{abrupt\_movement}(tr_0, \text{tram}, 101) \lor \\
& \neg \text{before}(101, 101) \lor \text{uncomfortable\_driving}(tr_0, \text{tram}, 101)
\end{align*}
\]
Markov Logic: Network Construction
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

\[
P(X = x_2) = \frac{1}{2} \exp\left(\frac{1}{3} w_1 \cdot 2 + \frac{1}{3} w_1 \cdot 2 + \frac{1}{3} w_1 \cdot 2 + w_2 \cdot 3\right) = \frac{1}{2} e^{2w_1+3w_2}
\]
Markov Logic: Inference

- Event recognition involves querying about HLE.
- Given a ground Markov network, apply standard probabilistic inference methods.
- Markov network may be large network and have a complex structure
  - Inference may become infeasible.
- MLN combine logical and probabilistic inference methods.
Markov Logic: Conditional inference

Query: Which trams are driven in an uncomfortable manner?

- Query variables $Q$: HLE

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

Query: Which trams are driven in an uncomfortable manner?

- 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: Which trams are driven in an uncomfortable manner?

- 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: MCMC

- Uncomfortable driving
- Abrupt movement
- Approach intersection
- Sharp turn
- Abrupt deceleration
- Abrupt acceleration

...
Markov Logic: Deterministic dependencies

- MCMC is a pure statistical method.
- MLN combine logic and probabilistic models.
- Hard constrained formulas:
  - Deterministic dependencies.
  - Isolated regions in state space.
- Strong constrained formulas:
  - Near-deterministic dependencies.
  - Difficult to cross regions.
- Combination of satisfiability testing with MCMC.
Event Calculus

Initiated

Initiated

Terminated

Time

0.0

0.5

1.0

0.0

3

10

20

Initiated

Initiated

Terminated
Event Calculus in MLN

Hard-constrained inertia rules:

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

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

Soft-constrained initiation inertia rules:

2.3 \( HLE \) initiated\(_{At} \) \( T \) if [Conditions]

2.5 \( HLE \) terminated\(_{At} \) \( T \) if [Conditions]

2.8 \( \neg(HLE \ holds\_{At} \ T) \) iff

\( \neg(HLE \ holds\_{At} \ T-1), \)

\( \neg(HLE \ initiated\_{At} \ T-1) \)

\( HLE \ holds\_{At} \ T \) iff

\( HLE \ holds\_{At} \ T-1, \)

\( \neg(HLE \ terminated\_{At} \ T-1) \)
Event Calculus in MLN

Soft-constrained termination inertia rules:

2.3 $\text{HLE initialedAt } T$ if [Conditions]

$\neg (\text{HLE holdsAt } T)$ iff
$\neg (\text{HLE holdsAt } T - 1)$,
$\neg (\text{HLE initialedAt } T - 1)$

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

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

Soft-constrained termination inertia rules:

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

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

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

\[ HLE \ \text{holdsAt} \ T-1, \]
\[ \neg (HLE \ \text{terminatedAt} \ T-1) \]
Event Calculus in MLN: Experimental Evaluation

![Graph 1](image1.png)

![Graph 2](image2.png)
Event Calculus in MLN: Summary

- We can deal with both:
  - Uncertainty in the HLE definitions, and
  - uncertainty in the input.
- Customisable inertia behaviour to meet the requirements of different applications.
- But:
  - There is room for improvement with respect to efficiency.
Event Recognition under Uncertainty

- Probabilistic reasoning improves recognition accuracy.
- But probabilistic reasoning often does not allow for real-time event recognition.
- Solution: self-adaptive event recognition
  - Streams from multiple sources are matched against each other to identify mismatches that indicate uncertainty in the sources.
  - Temporal regions of uncertainty are identified from which the system autonomously decides to adapt its event sources in order to deal with uncertainty, without compromising efficiency.
  - Data variety is used to handle veracity.
Self-Adaptive Event Recognition

\[\text{busReportedCongestion}(Lon, Lat) \text{ initiated iff move}(Bus, Lon_B, Lat_B, 1) \text{ happens, close}(Lon_B, Lat_B, Lon, Lat)\]

\[\text{busReportedCongestion}(Lon, Lat) \text{ terminated iff move}(Bus, Lon_B, Lat_B, 0) \text{ happens, close}(Lon_B, Lat_B, Lon, Lat)\]
Self-Adaptive Event Recognition:
Identifying Mismatches among Different Streams

\[\text{noisy}(Bus) \text{ initiated iff}\]
\[\begin{align*}
&\text{move}(Bus, Lon_B, Lat_B, 1) \text{ happens,} \\
&\text{close}(Lon_B, Lat_B, Lon_S, Lat_S), \\
&\neg (\text{scatsReportedCongestion}(Lon_S, Lat_S) \text{ holds})
\end{align*}\]

\[\text{noisy}(Bus) \text{ terminated if}\]
\[\begin{align*}
&\text{move}(Bus, Lon_B, Lat_B, 1) \text{ happens,} \\
&\text{close}(Lon_B, Lat_B, Lon_S, Lat_S), \\
&\text{scatsReportedCongestion}(Lon_S, Lat_S) \text{ holds}
\end{align*}\]

\[\text{noisy}(Bus) \text{ terminated if}\]
\[\begin{align*}
&\text{move}(Bus, Lon_B, Lat_B, 0) \text{ happens,} \\
&\text{close}(Lon_B, Lat_B, Lon_S, Lat_S), \\
&\neg (\text{scatsReportedCongestion}(Lon_S, Lat_S) \text{ holds})
\end{align*}\]
Self-Adaptive Event Recognition: Discard Temporarily Unreliable Event Sources

busReportedCongestion(Lon, Lat) initiated iff
move(Bus, Lon_B, Lat_B, 1) happens,
¬ (noisy(Bus) holds),
close(Lon_B, Lat_B, Lon, Lat)

busReportedCongestion(Lon, Lat) terminated iff
move(Bus, Lon_B, Lat_B, 0) happens,
¬ (noisy(Bus) holds),
close(Lon_B, Lat_B, Lon, Lat)
Self-Adaptive Event Recognition in Dublin

<table>
<thead>
<tr>
<th>Time (sec)</th>
<th>Working Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>10 min</td>
<td>12.5K LLE</td>
</tr>
<tr>
<td>30 min</td>
<td>40.5K LLE</td>
</tr>
<tr>
<td>50 min</td>
<td>67K LLE</td>
</tr>
<tr>
<td>70 min</td>
<td>94.5K LLE</td>
</tr>
<tr>
<td>90 min</td>
<td>124K LLE</td>
</tr>
<tr>
<td>110 min</td>
<td>152K LLE</td>
</tr>
</tbody>
</table>

Graph showing the comparison between Static Event Recognition and Self-Adaptive Event Recognition in terms of time (sec) and LLE usage.
Event Recognition Under Uncertainty: Summary

- Uncertainty in the input:
  - Probabilistic reasoning.
  - Using variety for veracity.

- Uncertainty in the HLE definitions:
  - Probabilistic reasoning.

- But:
  - We are still missing a framework for real-time, probabilistic event recognition under uncertainty.
Introduction.

Temporal reasoning systems.

Event recognition under uncertainty.

Open issues.
Open Issues

- User-friendly authoring tools enabling non-programmers to use event recognition.
- Real-time, probabilistic event recognition under uncertainty
  - including distributed/parallel processing.
- Internet scale event fabric
  - Management and orchestration—getting the right event to the right agent at the right granularity in the right time in very large, geographically distributed systems.
  - Comply with privacy and security considerations.
Open Issues

- Machine learning for the automated construction and refinement of HLE definitions.
- Event forecasting under uncertainty
  - allowing for proactive event-driven computing: recognise-forecast-decide-act.
Tutorial Resources


▶ Slides, papers & datasets on cer.iit.demokritos.gr