Formal Methods for Event Processing

Alexander Artikis and Georgios Paliouras

Complex Event Recognition Group
NCSR "Demokritos", Greece

{a.artikis, paliourg}@iit.demokritos.gr

http://cer.iit.demokritos.gr

Contributors

- Marek Sergot, Imperial College London.
- Jason Filippou, University of Maryland.
- Anastasios Skarlatidis, NCSR "Demokritos".
- Nikos Katzouris, NCSR "Demokritos".

Tutorial Resources

- Alexander Artikis, Anastasios Skarlatidis, Francois Portet, Georgios Paliouras: Logic-based event recognition. Knowledge Engineering Review 27(4): 469-506 (2012).
- Software, datasets, slides & papers at `cer.iit.demokritos.gr`

Event Recognition (Event Pattern Matching)

Input:

- Symbolic representation of time-stamped, low-level events (LLE) coming from (geographically distributed) sources.
- Big Data.

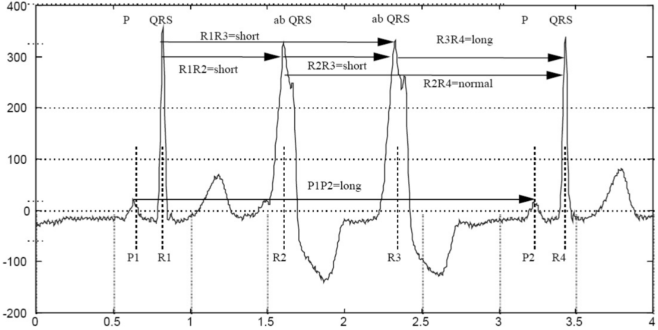
Output:

- High-level events (HLE) — collections of LLE and/or HLE that satisfy some pattern (temporal, spatial, logical constraints).
  - Not restricted to aggregates.
- Humans understand HLE easier than LLE.

Tutorial scope:

- Systems with a formal semantics.

Cardiac Arrhythmia Recognition



- LLE: P and QRS waves representing heart activity.
- HLE: Cardiac arrhythmias.

A cardiac arrhythmia is defined as a temporal combination of P and QRS waves.

Cardiac Arrhythmia Recognition

Input	Output
16338 qrs[normal]	
17091 p_wave[normal]	
17250 qrs[normal]	
17952 p_wave[normal]	
18913 p_wave[normal]	
19066 qrs[normal]	
19838 p_wave[normal]	
20713 p_wave[normal]	
20866 qrs[normal]	
21413 qrs[abnormal]	
21926 p_wave[normal]	
22496 qrs[normal]	
...	

Cardiac Arrhythmia Recognition

Input	Output
16338 qrs[normal]	[17091, 19066] mobitzII
17091 p_wave[normal]	
17250 qrs[normal]	
17952 p_wave[normal]	
18913 p_wave[normal]	
19066 qrs[normal]	
19838 p_wave[normal]	
20713 p_wave[normal]	
20866 qrs[normal]	
21413 qrs[abnormal]	
21926 p_wave[normal]	
22496 qrs[normal]	
...	

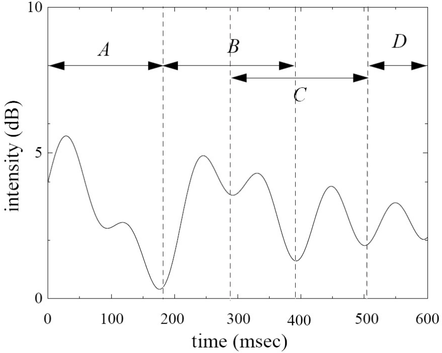
Cardiac Arrhythmia Recognition

Input	Output
77091 qrs[normal]	
77250 p_wave[normal]	
77952 qrs[normal]	
78913 qrs[abnormal]	
79066 p_wave[normal]	
79838 qrs[normal]	
80000 qrs[abnormal]	
80713 p_wave[normal]	
80866 qrs[normal]	
81413 qrs[abnormal]	
81926 p_wave[normal]	
. . .	

Cardiac Arrhythmia Recognition

Input	Output
77091 qrs[normal]	[78913, 81413] bigeminy
77250 p_wave[normal]	
77952 qrs[normal]	
78913 qrs[abnormal]	
79066 p_wave[normal]	
79838 qrs[normal]	
80000 qrs[abnormal]	
80713 p_wave[normal]	
80866 qrs[normal]	
81413 qrs[abnormal]	
81926 p_wave[normal]	
...	

Humpback Whale Song Recognition



- LLE: Song units representing whale sounds.
- HLE: Whale songs.

A whale song is defined as a temporal combination of songs units.

Humpback Whale Song Recognition

Input	Output
$[200, 400]$	A
$[400, 500]$	B
$[500, 550]$	C
$[600, 700]$	B
$[700, 800]$	D
$[800, 1000]$	A
$[1050, 1200]$	E
$[1300, 1500]$	B
$[1600, 1800]$	E
$[1800, 1900]$	C
$[1900, 2000]$	B
...	

Humpback Whale Song Recognition

Input		Output	
[200, 400]	A	[200, 550]	S_1
[400, 500]	B	[700, 1200]	S_2
[500, 550]	C	[1600, 2000]	S_3
[600, 700]	B	...	
[700, 800]	D		
[800, 1000]	A		
[1050, 1200]	E		
[1300, 1500]	B		
[1600, 1800]	E		
[1800, 1900]	C		
[1900, 2000]	B		
...			

Event Recognition for Maritime Surveillance

LLE:

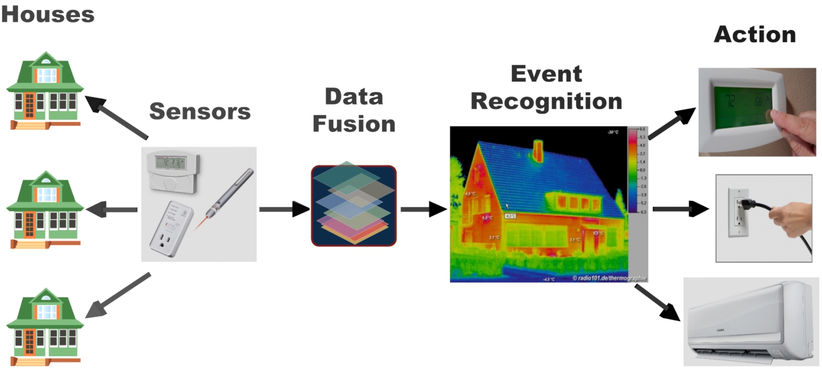
- Vessel movement.
- Entering/leaving port.
- Communication gap.

HLE:

- Shipping in protected areas.
- Shipping in unsafe areas.
- Loitering.
- Collision.
- Forbidden fishing.

HLE are spatio-temporal combinations of LLE and background knowledge.

Event Recognition for Energy Management



Credit Card Fraud Recognition

LLE:

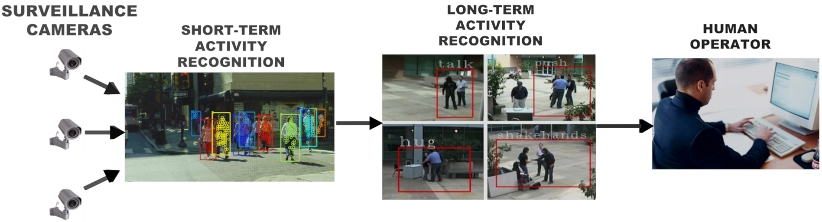
- Credit card transactions from all over the world.

HLE:

- Cloned card — a credit card is being used simultaneously in different countries.
- Spike usage — the 24-hour running sum is considerably higher than the monthly average of the last 6 months.
- New high use — the card is being frequently used in merchants or countries never used before.
- Potential batch fraud — many transactions from multiple cards in the same point-of-sale terminal in high amounts.

A fraud is a spatio-temporal combination of transactions and background knowledge.

Running Example I: Event Recognition for Public Space Surveillance



Event Recognition for Public Space Surveillance

Input	Output
340 $inactive(id_0)$	
340 $p(id_0) = (20.88, -11.90)$	
340 $appear(id_0)$	
340 $walking(id_2)$	
340 $p(id_2) = (25.88, -19.80)$	
340 $active(id_1)$	
340 $p(id_1) = (20.88, -11.90)$	
340 $walking(id_3)$	
340 $p(id_3) = (24.78, -18.77)$	
380 $walking(id_3)$	
380 $p(id_3) = (27.88, -9.90)$	
380 $walking(id_2)$	
380 $p(id_2) = (28.27, -9.66)$	

Event Recognition for Public Space Surveillance

Input	Output
340 $inactive(id_0)$	340 $leaving\_object(id_1, id_0)$
340 $p(id_0) = (20.88, -11.90)$	
340 $appear(id_0)$	
340 $walking(id_2)$	
340 $p(id_2) = (25.88, -19.80)$	
340 $active(id_1)$	
340 $p(id_1) = (20.88, -11.90)$	
340 $walking(id_3)$	
340 $p(id_3) = (24.78, -18.77)$	
380 $walking(id_3)$	
380 $p(id_3) = (27.88, -9.90)$	
380 $walking(id_2)$	
380 $p(id_2) = (28.27, -9.66)$	

Event Recognition for Public Space Surveillance

Input	Output
340 $inactive(id_0)$	340 $leaving\_object(id_1, id_0)$
340 $p(id_0) = (20.88, -11.90)$	$since(340)$ $moving(id_2, id_3)$
340 $appear(id_0)$	
340 $walking(id_2)$	
340 $p(id_2) = (25.88, -19.80)$	
340 $active(id_1)$	
340 $p(id_1) = (20.88, -11.90)$	
340 $walking(id_3)$	
340 $p(id_3) = (24.78, -18.77)$	
380 $walking(id_3)$	
380 $p(id_3) = (27.88, -9.90)$	
380 $walking(id_2)$	
380 $p(id_2) = (28.27, -9.66)$	

Event Recognition for Public Space Surveillance

Input	Output
420 $active(id_4)$	
420 $p(id_4) = (10.88, -71.90)$	
420 $inactive(id_3)$	
420 $p(id_3) = (5.8, -50.90)$	
420 $abrupt(id_5)$	
420 $p(id_5) = (11.80, -72.80)$	
420 $active(id_6)$	
420 $p(id_6) = (7.8, -52.90)$	
480 $abrupt(id_4)$	
480 $p(id_4) = (20.45, -12.90)$	
480 $abrupt(id_5)$	
480 $p(id_5) = (17.88, -11.90)$	
...	

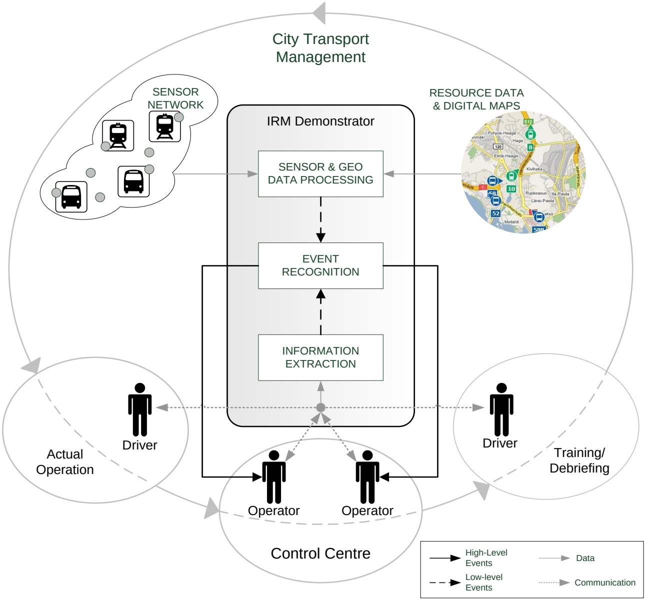
Event Recognition for Public Space Surveillance

Input	Output
420 $active(id_4)$	$[420, 480]$ $fighting(id_4, id_5)$
420 $p(id_4) = (10.88, -71.90)$	
420 $inactive(id_3)$	
420 $p(id_3) = (5.8, -50.90)$	
420 $abrupt(id_5)$	
420 $p(id_5) = (11.80, -72.80)$	
420 $active(id_6)$	
420 $p(id_6) = (7.8, -52.90)$	
480 $abrupt(id_4)$	
480 $p(id_4) = (20.45, -12.90)$	
480 $abrupt(id_5)$	
480 $p(id_5) = (17.88, -11.90)$	
...	

Event Recognition for Public Space Surveillance

Input	Output
420 $active(id_4)$	$[420, 480]$ $fighting(id_4, id_5)$
420 $p(id_4) = (10.88, -71.90)$	$since(420)$ $meeting(id_3, id_6)$
420 $inactive(id_3)$	
420 $p(id_3) = (5.8, -50.90)$	
420 $abrupt(id_5)$	
420 $p(id_5) = (11.80, -72.80)$	
420 $active(id_6)$	
420 $p(id_6) = (7.8, -52.90)$	
480 $abrupt(id_4)$	
480 $p(id_4) = (20.45, -12.90)$	
480 $abrupt(id_5)$	
480 $p(id_5) = (17.88, -11.90)$	
...	

Running Example II



Event Recognition for Transport & Traffic Management

	Input	Output
200	scheduled stop enter	
215	late stop leave	
[215, 400]	abrupt acceleration	
[350, 600]	sharp turn	
620	flow=low	
	density=high	
700	scheduled stop enter	
720	flow=low	
	density=average	
820	scheduled stop leave	
...		

Event Recognition for Transport & Traffic Management

Input		Output	
200	scheduled stop enter		
215	late stop leave	*since*(215)	non-punctual
[215, 400]	abrupt acceleration		
[350, 600]	sharp turn	[215, 600]	uncomfortable driving
620	flow=low		
	density=high	since(620)	congestion
700	scheduled stop enter		
720	flow=low		
	density=average		
820	scheduled stop leave		
...			

Event Recognition for Transport & Traffic Management

Input		Output	
200	scheduled stop enter		
215	late stop leave	*since*(215)	non-punctual
[215, 400]	abrupt acceleration		
[350, 600]	sharp turn	[215, 600]	uncomfortable driving
620	<u>flow=low</u>		
	<u>density=high</u>	since(620)	congestion
700	scheduled stop enter		
720	<u>flow=low</u>		
	<u>density=average</u>	[620,720]	congestion
820	scheduled stop leave	[215,820]	non-punctual
...			

Event Recognition

Requirements:

- Efficient reasoning
  - to support real-time decision-making in large-scale, (geographically) distributed applications.
- Reasoning under uncertainty
  - to deal with various types of noise.
- Automated knowledge construction
  - to avoid the time-consuming, error-prone manual HLE definition development.

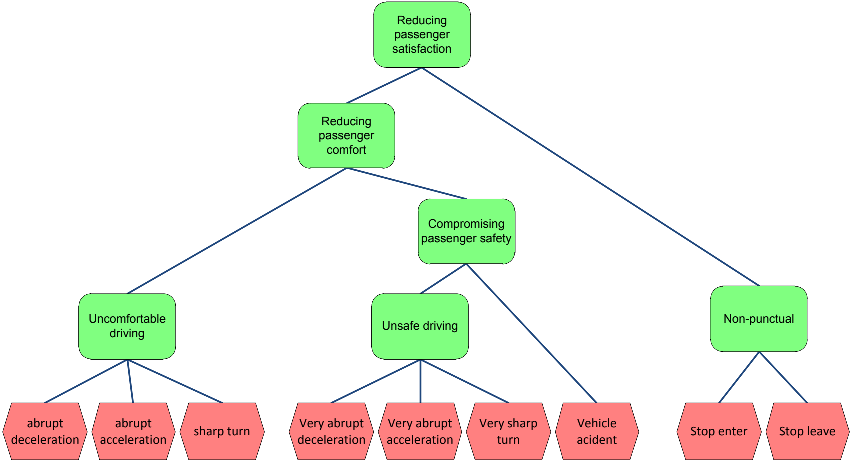
Tutorial Structure

- Temporal reasoning systems.
- Event recognition under uncertainty.
- Machine learning for event recognition.
- Open issues.

Tutorial Structure

- Temporal reasoning systems.
- Event recognition under uncertainty.
- Machine learning for event recognition.
- Open issues.

HLE Definition



HLE as Chronicle

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

Predicate	Meaning
`event(E, T)`	Event `E` takes place at time `T`
`event(F:(?V1,?V2),T)`	An event takes place at time `T` changing the value of property `F` from `?V1` to `?V2`
`noevent(E, (T1,T2))`	Event `E` does not take place between [`T1,T2`)
`noevent(F:(?V1,?V2), (T1,T2))`	No event takes place between [`T1,T2`) that changes the value of property `F` from `?V1` to `?V2`
`hold(F:?V, (T1,T2))`	The value of property `F` is `?V` between [`T1,T2`)
`occurs(N,M,E,(T1,T2))`	Event `E` takes place at least `N` times and at most `M` times between [`T1,T2`)

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

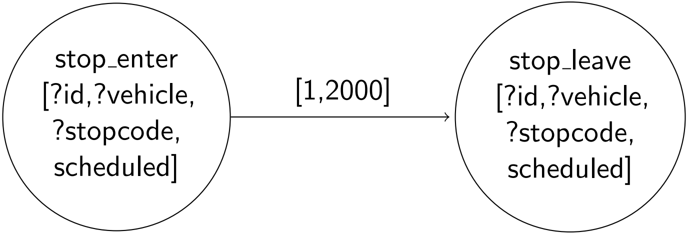
- Mathematical operators in the atemporal constraints of the language are not allowed:
  - cannot express that passenger safety is compromised more when a vehicle accident takes place far from a hospital or a police station.
- Universal quantification is not allowed:
  - cannot express that a route is punctual if all buses of the route are punctual.

CRS is a purely temporal reasoning system.

It is also a very efficient and scalable system.

Chronicle Recognition System: Semantics

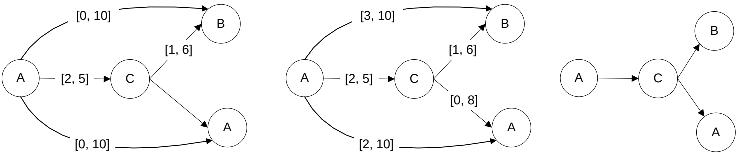
Each HLE definition is represented as a Temporal Constraint Network. Eg:



Chronicle Recognition System: Consistency Checking

Compilation stage:

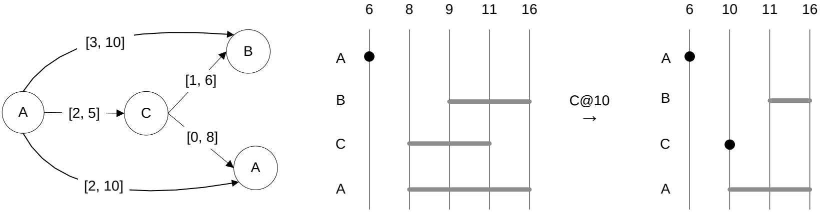
- Constraint propagation in the Temporal Constraint Network.
- Consistency checking.



Chronicle Recognition System: Recognition

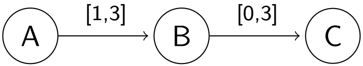
Recognition stage:

- Partial HLE instance evolution.
- Forward (predictive) recognition.

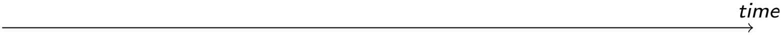


Chronicle Recognition System: Partial instances

HLE definition: Reduce tram endurance

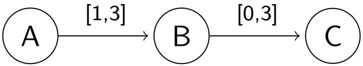


A: enter tram intersection
B: abrupt deceleration
C: abrupt acceleration

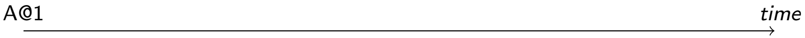


Chronicle Recognition System: Partial instances

HLE definition: Reduce tram endurance

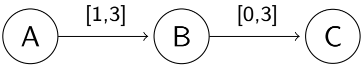


A: enter tram intersection
B: abrupt deceleration
C: abrupt acceleration

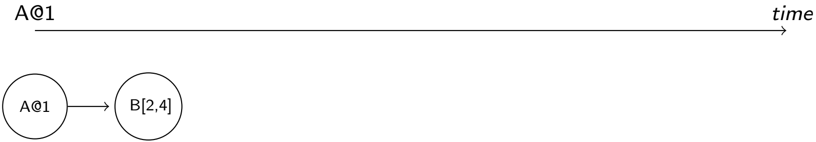


Chronicle Recognition System: Partial instances

HLE definition: Reduce tram endurance

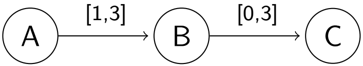


A: enter tram intersection
B: abrupt deceleration
C: abrupt acceleration

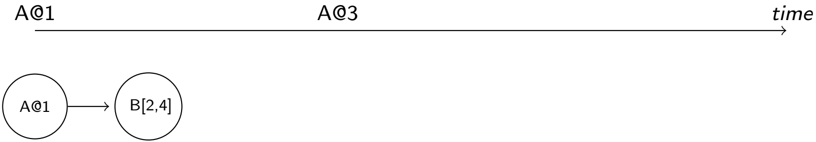


Chronicle Recognition System: Partial instances

HLE definition: Reduce tram endurance

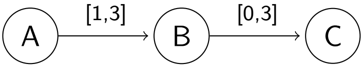


A: enter tram intersection
B: abrupt deceleration
C: abrupt acceleration

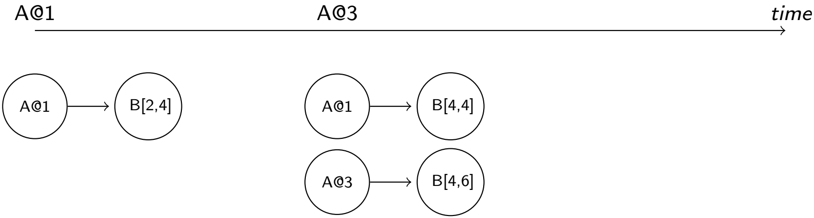


Chronicle Recognition System: Partial instances

HLE definition: Reduce tram endurance

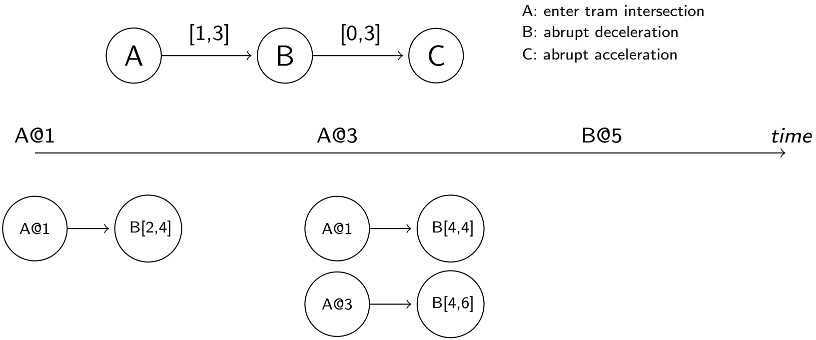


A: enter tram intersection
B: abrupt deceleration
C: abrupt acceleration



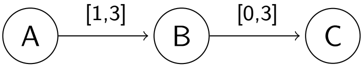
Chronicle Recognition System: Partial instances

HLE definition: Reduce tram endurance

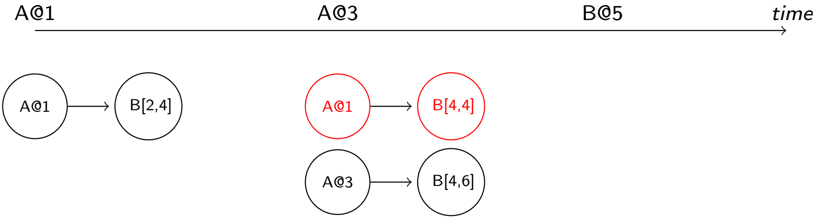


Chronicle Recognition System: Partial instances

HLE definition: Reduce tram endurance

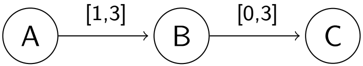


A: enter tram intersection
B: abrupt deceleration
C: abrupt acceleration

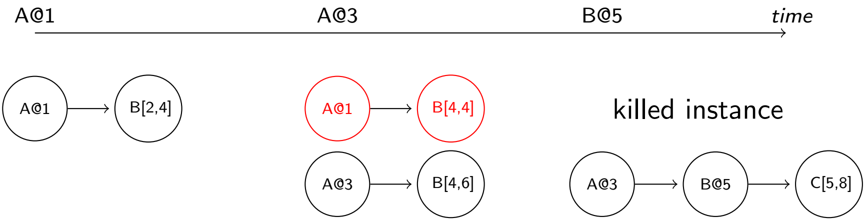


Chronicle Recognition System: Partial instances

HLE definition: Reduce tram endurance

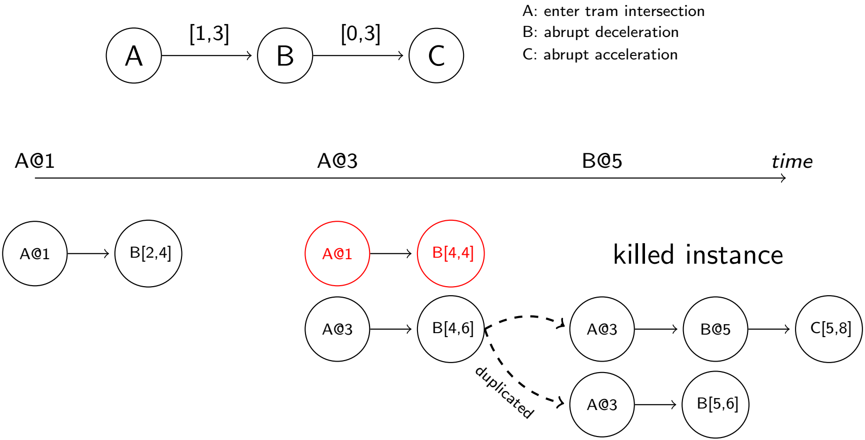


A: enter tram intersection
B: abrupt deceleration
C: abrupt acceleration



Chronicle Recognition System: Partial instances

HLE definition: Reduce tram endurance



Chronicle Recognition System

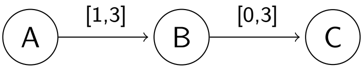
Recognition stage — partial HLE instance management:

- 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.
- For K HLE instances, each having n subevents, the complexity of processing each incoming event or a clock update is $O(Kn^2)$.
- To deal with out-of-order LLE streams, CRS keeps in memory partial HLE instances longer.

Chronicle Recognition System: Optimisation

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

- Distinguish between very rare events and frequent events based on a priori knowledge.
- 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.
- 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.
- Being AI-based, it has been largely overlooked by the event processing community.
- 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:

$$
\begin{aligned}
&\mathit{leaving\_object}(P, Obj)\ \textbf{initiated}\ \text{ iff} \\
&\qquad \mathit{appear}(Obj)\ \ \textbf{happens}, \\
&\qquad \mathit{inactive}(Obj)\ \ \textbf{holds}, \\
&\qquad \mathit{close}(P, Obj)\ \ \textbf{holds}, \\
&\qquad \mathit{person}(P)\ \ \textbf{holds}
\end{aligned}
$$

$$
\begin{aligned}
&\mathit{leaving\_object}(P, Obj)\ \textbf{terminated}\ \text{ iff} \\
&\qquad \mathit{disappear}(Obj)\ \ \textbf{happens}
\end{aligned}
$$

HLE recognition:

- $\mathit{leaving\_object}(P, Obj)$ **holdsFor** I

HLE Definitions in the Event Calculus

HLE definition:

$punctuality(ID) = non\_punctual$ **initiated** iff
$enter\_stop(ID, Stop, late)$ **happens** or
$leave\_stop(ID, Stop, early)$ **happens**

$punctuality(ID) = non\_punctual$ **terminatedAt** T iff
$enter\_stop(ID, Stop, scheduled)$ **happensAt** T',
$leave\_stop(ID, Stop, scheduled)$ **happensAt** T,
$T > T'$

HLE recognition:

- $punctuality(ID) = non\_punctual$ **holdsFor** I

HLE Definitions in the Event Calculus

HLE definition:

$$\begin{aligned}&\mathit{driving\_quality}(ID) = \mathit{low} \text{ iff}\\ &\quad \mathit{punctuality}(ID) = \mathit{non\_punctual} \text{ or}\\ &\quad \mathit{driving\_style}(ID) = \mathit{unsafe}\end{aligned}$$

Compiled HLE definition:

$$\begin{aligned}&\mathit{driving\_quality}(ID) = \mathit{low} \;\; \textbf{holdsFor} \;\; I_1 \cup I_2 \;\; \text{iff}\\ &\quad \mathit{punctuality}(ID) = \mathit{non\_punctual} \;\; \textbf{holdsFor} \;\; I_1,\\ &\quad \mathit{driving\_style}(ID) = \mathit{unsafe} \;\; \textbf{holdsFor} \;\; I_2\end{aligned}$$

HLE Definitions in the Event Calculus

HLE definition:

$$\begin{aligned}&driving\_quality(ID) = medium \text{ iff}\\&\quad punctuality(ID) = punctual,\\&\quad driving\_style(ID) = uncomfortable\end{aligned}$$

Compiled HLE definition:

$$\begin{aligned}&driving\_quality(ID) = medium \ \ \textbf{holdsFor} \ \ I_1 \cap I_2 \ \ \text{iff}\\&\quad punctuality(ID) = punctual \ \ \textbf{holdsFor} \ \ I_1,\\&\quad driving\_style(ID) = uncomfortable \ \ \textbf{holdsFor} \ \ I_2\end{aligned}$$

HLE Definitions in the Event Calculus

HLE definition:

$$\begin{aligned}&\mathit{fighting}(P_1, P_2) \text{ iff}\\&\quad(\mathit{abrupt}(P_1) \text{ or } \mathit{abrupt}(P_2)),\\&\quad\mathit{close}(P_1, P_2),\\&\quad\text{not } (\mathit{inactive}(P_1) \text{ or } \mathit{inactive}(P_2))\end{aligned}$$

Compiled HLE definition:

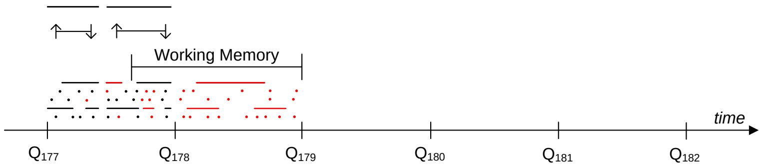
$$\begin{aligned}&\mathit{fighting}(P_1, P_2) \ \textbf{holdsFor} \ ((I_1 \cup I_2) \cap I_3) \setminus (I_4 \cup I_5) \ \text{ iff}\\&\quad\mathit{abrupt}(P_1) \ \textbf{holdsFor} \ I_1,\\&\quad\mathit{abrupt}(P_2) \ \textbf{holdsFor} \ I_2,\\&\quad\mathit{close}(P_1, P_2) \ \textbf{holdsFor} \ I_3,\\&\quad\mathit{inactive}(P_1) \ \textbf{holdsFor} \ I_4,\\&\quad\mathit{inactive}(P_2) \ \textbf{holdsFor} \ I_5\end{aligned}$$

Run-Time Event Recognition

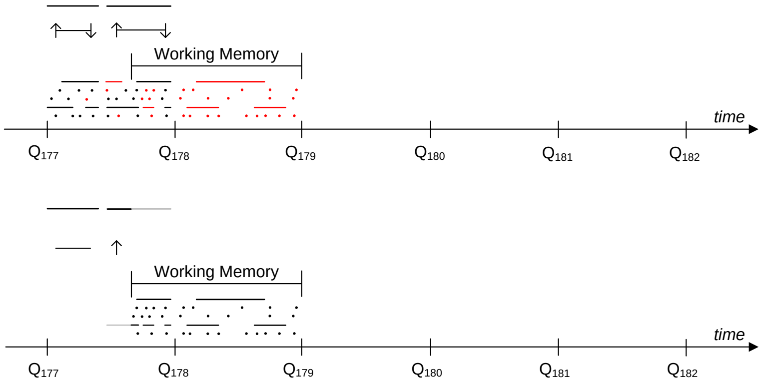
Real-time decision-support in the presence of:

- Very large LLE streams.
- Non-sorted LLE streams.
- LLE revision.
- Very large HLE numbers.

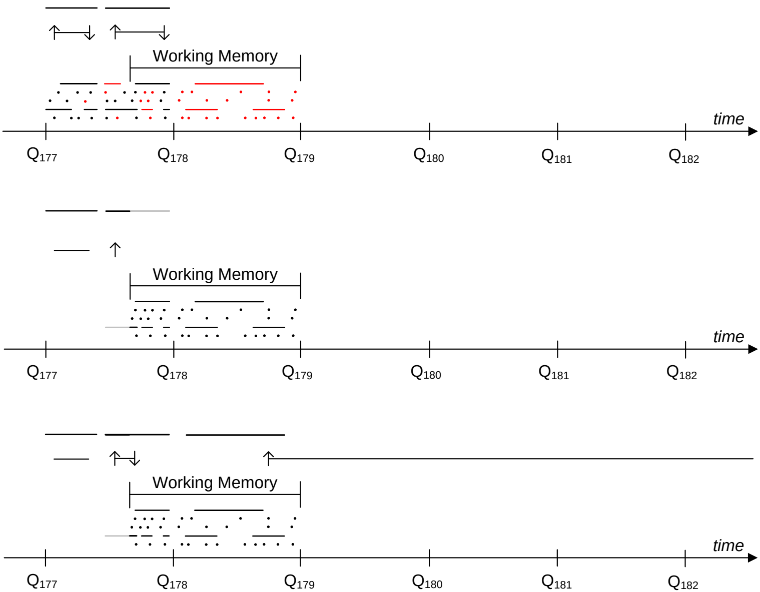
Event Calculus: Run-Time Event Recognition



Event Calculus: Run-Time Event Recognition



Event Calculus: Run-Time Event Recognition



Event Calculus: Summary

- Representation of complex temporal phenomena.
  - Succinct representation $\rightarrow$ code maintenance.
  - Intuitive representation $\rightarrow$ facilitates interaction with domain experts unfamiliar with programming.
- The full power of logic programming is available.
  - Complex atemporal computations in HLE definitions.
  - Combination of streaming data with historical knowledge.
- Very efficient reasoning.
  - Even when LLE arrive with a delay and are revised.
  - Even in the presence of large HLE hierarchies.
- But:
  - The Event Calculus has to deal with uncertainty.

Tutorial Structure

- Temporal reasoning systems.
- Event recognition under uncertainty.
- Machine learning for event recognition.
- Open issues.

Common Problems of Event Recognition

- Limited dictionary of LLE and context variables.
  - No explicit representation of hand shaking, falling down, etc.
- Incomplete LLE stream.
  - Abrupt acceleration was not detected.
- Erroneous LLE detection.
  - Abrupt acceleration was classified as sharp turn.
- Inconsistent ground truth (HLE & LLE annotation).
  - Disagreement between (human) annotators.

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

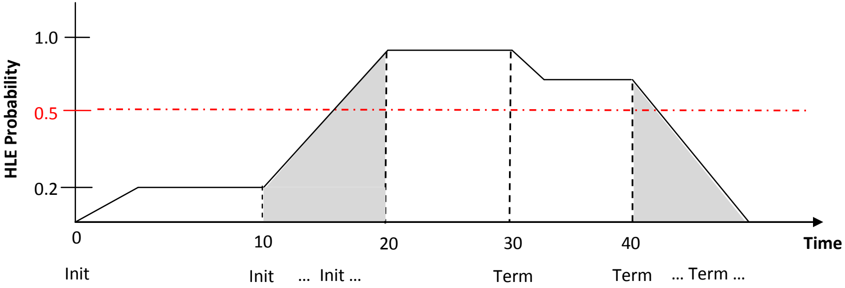
- A Probabilistic Logic Programming language.
- Allows for independent ‘probabilistic facts’ *prob::fact*.
- *Prob* indicates the probability that *fact* is part of a possible world.
- Rules are written as in classic Prolog.
- The probability of a query q imposed on a ProbLog database *(success probability)* is computed by the following formula:

$$P_s(q) = P\Big(\bigvee_{e \in Proofs(q)} \bigwedge_{f_i \in e} f_i \Big)$$

Event Recognition using ProbLog

Input	Output
340 0.45 :: $inactive(id_0)$	340 0.41 :: $leaving\_object(id_1, id_0)$
340 0.80 :: $p(id_0) = (20.88, -11.90)$	340 0.55 :: $moving(id_2, id_3)$
340 0.55 :: $appear(id_0)$	
340 0.15 :: $walking(id_2)$	
340 0.80 :: $p(id_2) = (25.88, -19.80)$	
340 0.25 :: $active(id_1)$	
340 0.66 :: $p(id_1) = (20.88, -11.90)$	
340 0.70 :: $walking(id_3)$	
340 0.46 :: $p(id_3) = (24.78, -18.77)$	
...	

Event Calculus in ProbLog

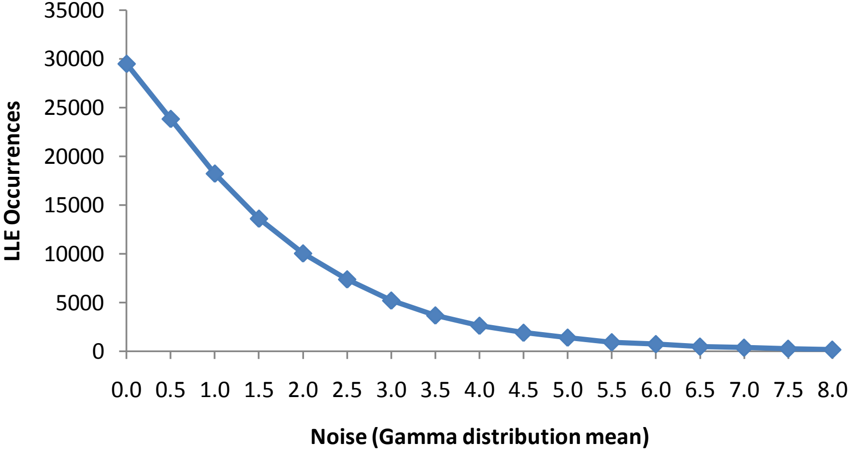


Event Calculus in ProbLog: Experimental Evaluation

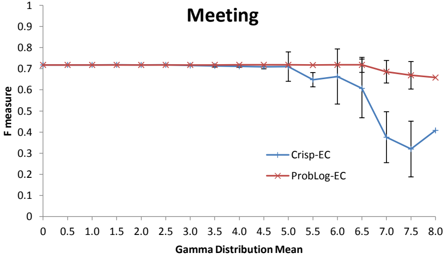
To compare ProbLog-EC to Crisp-EC:

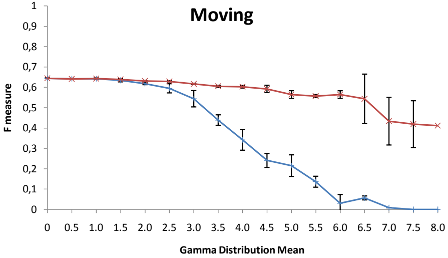
- We add noise (probabilities) in LLE:
  - Crisp-EC: LLE with probability < 0.5 are discarded.
  - ProbLog-EC: all LLE are kept with their probabilities.
- In ProbLog-EC we accept as recognised the HLE that have probability > 0.5.

Event Calculus in ProbLog: Experimental Evaluation



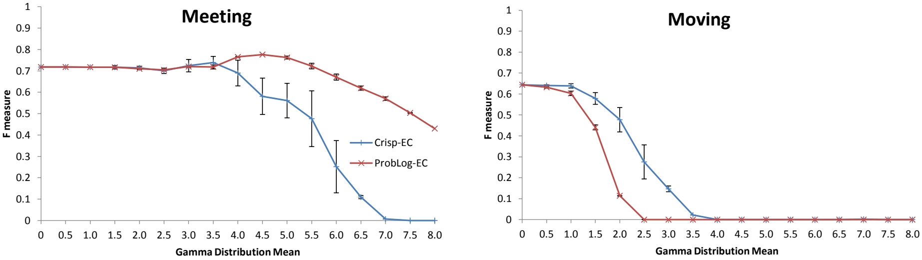
Event Calculus in ProbLog: Experimental Evaluation





$$
\begin{array}{l}
moving(P_1, P_2)\ \ \textbf{initiated}\ \ \text{iff} \\
\quad walking(P_1)\ \ \textbf{happens}, \\
\quad walking(P_2)\ \ \textbf{happens}, \\
\quad close(P_1, P_2)\ \ \textbf{holds}, \\
\quad orientation(P_1) = O_1\ \ \textbf{holds}, \\
\quad orientation(P_2) = O_2\ \ \textbf{holds}, \\
\quad |O_1 - O_2| < threshold
\end{array}
$$

Event Calculus in ProbLog: Experimental Evaluation

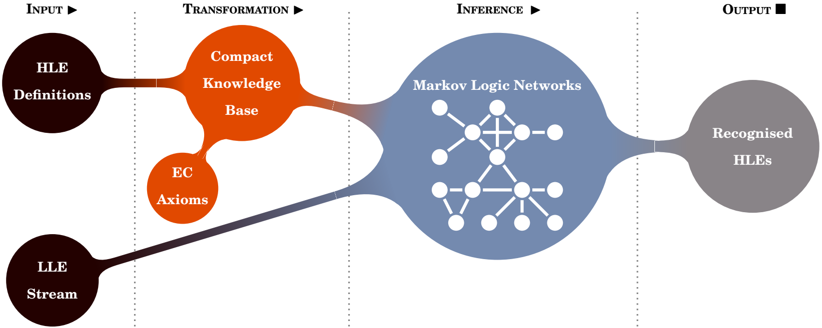


$moving(P_1, P_2)$ **initiated** iff
$walking(P_1)$ **happens**,
$walking(P_2)$ **happens**,
$close(P_1, P_2)$ **holds**,
$orientation(P_1) = O_1$ **holds**,
$orientation(P_2) = O_2$ **holds**,
$|O_1 - O_2| < threshold$

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.

Markov Logic: Representation

Example definition of HLE 'uncomfortable_driving' :

$$w_1 \quad \begin{aligned} & abrupt\_movement(Id, Vehicle, T) \leftarrow \\ & \quad abrupt\_acceleration(Id, Vehicle, T) \vee \\ & \quad abrupt\_deceleration(Id, Vehicle, T) \vee \\ & \quad sharp\_turn(Id, Vehicle, T) \end{aligned}$$

$$w_2 \quad \begin{aligned} & uncomfortable\_driving(Id, Vehicle, T_2) \leftarrow \\ & \quad approach\_intersection(Id, Vehicle, T_1) \wedge \\ & \quad abrupt\_movement(Id, Vehicle, T_2) \wedge \\ & \quad before(T_1, T_2) \end{aligned}$$

Markov Logic: Representation

- Weight: a real-valued number.
- Higher weight $\longrightarrow$ Stronger constraint
- Hard constraints
  - Infinite weight values.
  - Background knowledge.
- 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:

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

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

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

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

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:
$T = \{100, 101\}$
$Id = \{tr_0\}$
$Vehicle = \{tram\}$

Markov Logic: Network Construction

For example, the clause:

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

produces the following groundings:

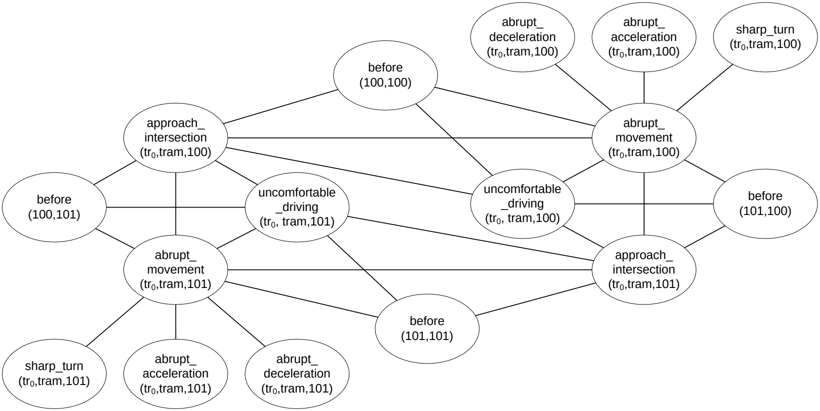
$$w_2 \quad \neg approach\_intersection(tr_0, tram, 100) \ \lor \neg abrupt\_movement(tr_0, tram, 100) \lor \neg before(100, 100) \lor uncomfortable\_driving(tr_0, tram, 100)$$

$$w_2 \quad \neg approach\_intersection(tr_0, tram, 100) \ \lor \neg abrupt\_movement(tr_0, tram, 101) \lor \neg before(100, 101) \lor uncomfortable\_driving(tr_0, tram, 101)$$

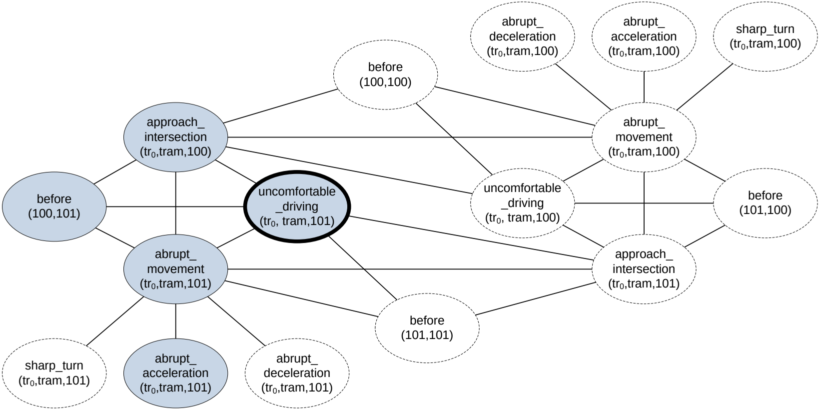
$$w_2 \quad \neg approach\_intersection(tr_0, tram, 101) \ \lor \neg abrupt\_movement(tr_0, tram, 100) \lor \neg before(101, 100) \lor uncomfortable\_driving(tr_0, tram, 100)$$

$$w_2 \quad \neg approach\_intersection(tr_0, tram, 101) \ \lor \neg abrupt\_movement(tr_0, tram, 101) \lor \neg before(101, 101) \lor uncomfortable\_driving(tr_0, tram, 101)$$

Markov Logic: Network Construction

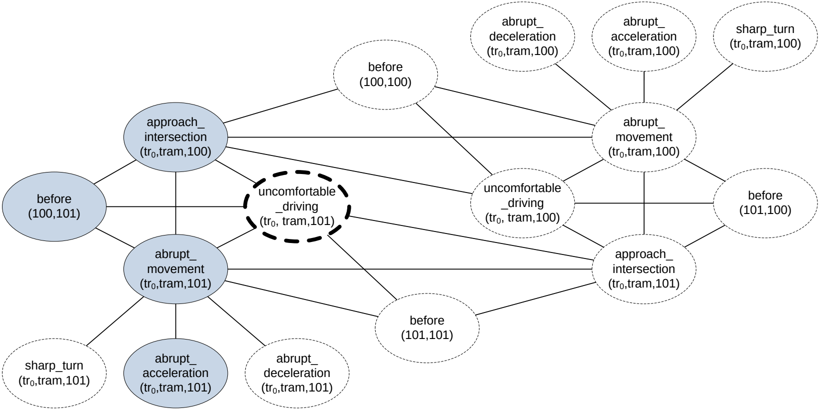


Markov Logic: World state discrimination



$$P(X = x_1) = \frac{1}{Z} exp(\frac{1}{3} w_1 \cdot 2 + \frac{1}{3} w_1 \cdot 2 + \frac{1}{3} w_1 \cdot 2 + w_2 \cdot 4) = \frac{1}{Z} e^{2w_1 + 4w_2}$$

Markov Logic: World state discrimination



$$P(X = x_1) = \frac{1}{Z} exp(\frac{1}{3} w_1 \cdot 2 + \frac{1}{3} w_1 \cdot 2 + \frac{1}{3} w_1 \cdot 2 + w_2 \cdot 4) = \frac{1}{Z} e^{2w_1 + 4w_2}$$

$$P(X = x_2) = \frac{1}{Z} exp(\frac{1}{3} w_1 \cdot 2 + \frac{1}{3} w_1 \cdot 2 + \frac{1}{3} w_1 \cdot 2 + w_2 \cdot 3) = \frac{1}{Z} e^{2w_1 + 3w_2}$$

Markov Logic: Inference

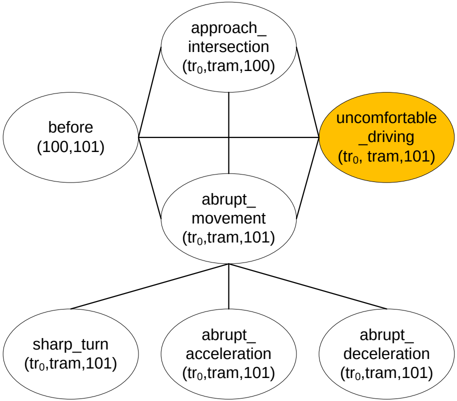
- Event recognition involves querying about HLE.
- Given a ground Markov network, apply standard probabilistic inference methods.
- Markov network may be large and have a complex structure
  - Inference may become infeasible.
- MLN combine logical and probabilistic inference methods.

Markov Logic: Conditional inference

Query: Which trams are driven in an uncomfortable manner?

- Query variables Q: HLE

$$P(Q \mid E = e) = \frac{P(Q, E = e, H)}{P(E = e, H)}$$

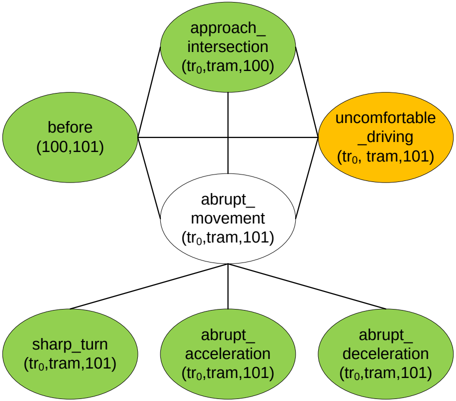


Markov Logic: Conditional inference

Query: Which trams are driven in an uncomfortable manner?

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

$$P(Q \mid E = e) = \frac{P(Q, E = e, H)}{P(E = e, H)}$$

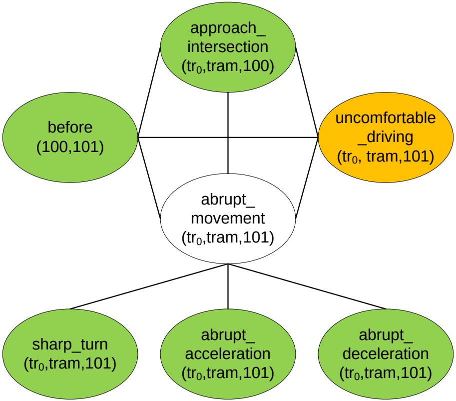


Markov Logic: Conditional inference

Query: Which trams are driven in an uncomfortable manner?

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

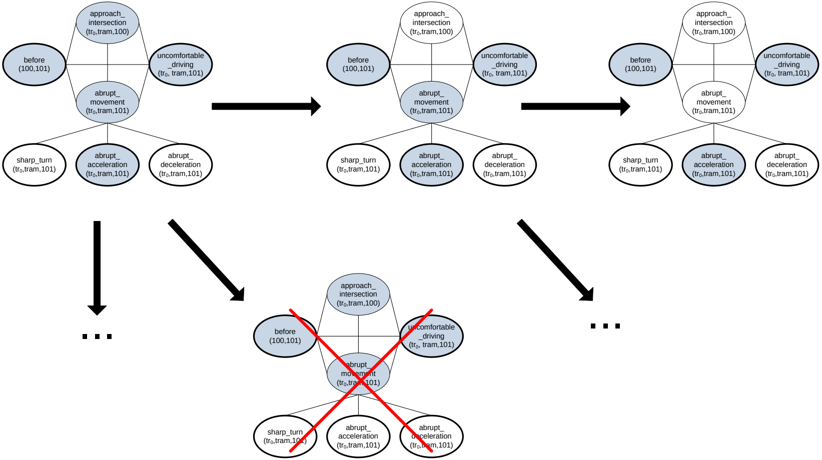
$$P(Q \mid E = e) = \frac{P(Q, E = e, H)}{P(E = e, H)}$$



Markov Logic: Conditional inference

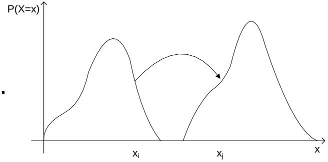
- Efficiently approximated with sampling.
- Markov Chain Monte Carlo (MCMC): e.g Gibbs sampling.
- Random walks in state space.
- Reject all states where $E = e$ does not hold.

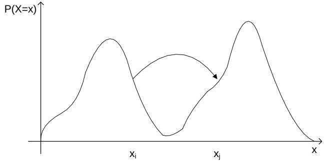
Markov Logic: MCMC



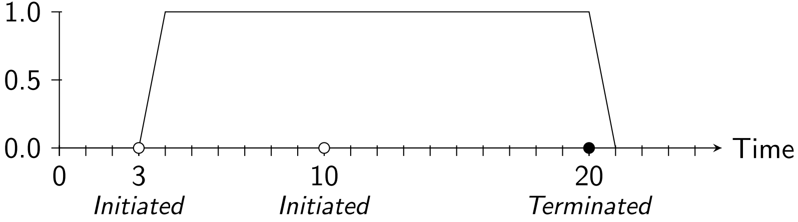
Markov Logic: Deterministic dependencies

- 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.

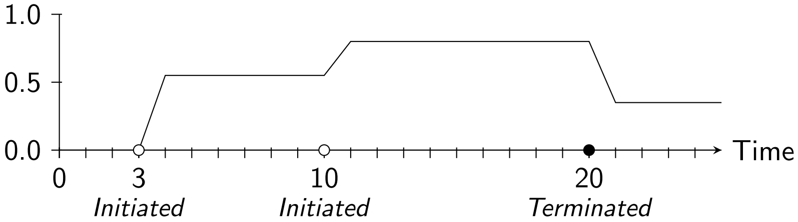




Event Calculus



Event Calculus in MLN



Hard-constrained inertia rules:

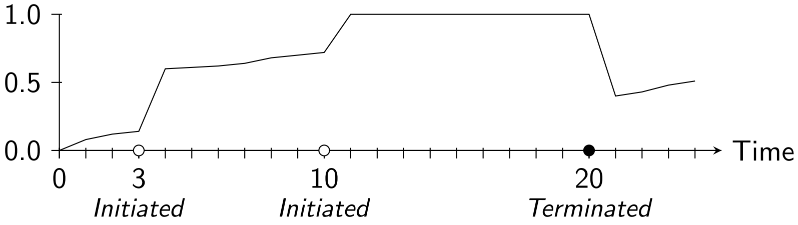
2.3 HLE **initiatedAt** T if
$[\text{Conditions}]$

$\neg(HLE$ **holdsAt** $T)$ iff
$\neg(HLE$ **holdsAt** $T-1)$,
$\neg(HLE$ **initiatedAt** $T-1)$

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

HLE **holdsAt** T iff
HLE **holdsAt** $T-1$,
$\neg(HLE$ **terminatedAt** $T-1)$

Event Calculus in MLN



Soft-constrained initiation inertia rules:

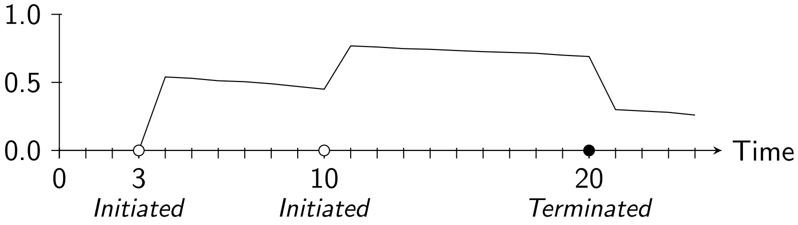
2.3 *HLE* **initiatedAt** *T* if
[Conditions]

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

2.8 $\neg$(*HLE* **holdsAt** *T*) iff
$\neg$(*HLE* **holdsAt** $T-1$),
$\neg$(*HLE* **initiatedAt** $T-1$)

HLE **holdsAt** *T* iff
HLE **holdsAt** $T-1$,
$\neg$(*HLE* **terminatedAt** $T-1$)

Event Calculus in MLN



Soft-constrained termination inertia rules:

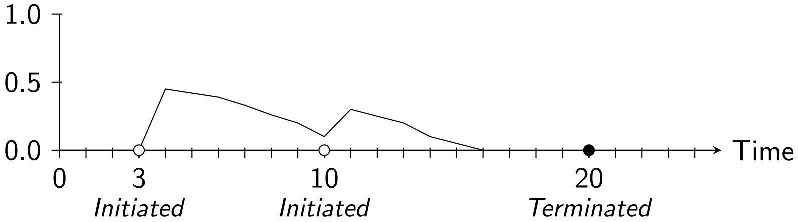
2.3 $\textit{HLE}$ **initiatedAt** T if
$[\text{Conditions}]$

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

$\neg(\textit{HLE}$ **holdsAt** $T)$ iff
$\neg(\textit{HLE}$ **holdsAt** $T{-}1)$,
$\neg(\textit{HLE}$ **initiatedAt** $T{-}1)$

2.8 $\textit{HLE}$ **holdsAt** T iff
$\textit{HLE}$ **holdsAt** $T{-}1$,
$\neg(\textit{HLE}$ **terminatedAt** $T{-}1)$

Event Calculus in MLN



Soft-constrained termination inertia rules:

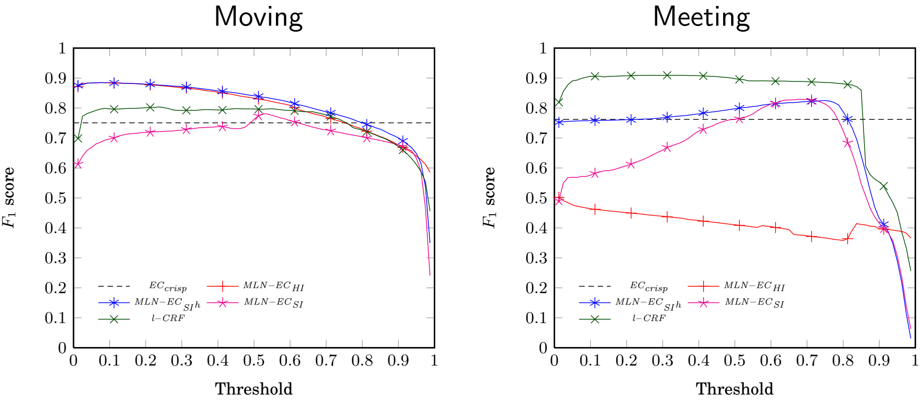
2.3   *HLE* **initiatedAt** *T* if
[Conditions]

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

¬(*HLE* **holdsAt** *T*) iff
¬(*HLE* **holdsAt** $T-1$),
¬(*HLE* **initiatedAt** $T-1$)

0.8   *HLE* **holdsAt** *T* iff
HLE **holdsAt** $T-1$,
¬(*HLE* **terminatedAt** $T-1$)

Event Calculus in MLN: Experimental Evaluation



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

$busReportedCongestion(Lon, Lat)$ **initiated** iff
$move(Bus, Lon_B, Lat_B, 1)$ **happens**,
$close(Lon_B, Lat_B, Lon, Lat)$

$busReportedCongestion(Lon, Lat)$ **terminated** iff
$move(Bus, Lon_B, Lat_B, 0)$ **happens**,
$close(Lon_B, Lat_B, Lon, Lat)$

Self-Adaptive Event Recognition: Identifying Mismatches among Different Streams

$noisy(Bus)$ **initiated** iff
$move(Bus, Lon_B, Lat_B, 1)$ **happens**,
$close(Lon_B, Lat_B, Lon_S, Lat_S)$,
$\neg$ ($scatsReportedCongestion(Lon_S, Lat_S)$ **holds**)

$noisy(Bus)$ **terminated** if
$move(Bus, Lon_B, Lat_B, 1)$ **happens**,
$close(Lon_B, Lat_B, Lon_S, Lat_S)$,
$scatsReportedCongestion(Lon_S, Lat_S)$ **holds**

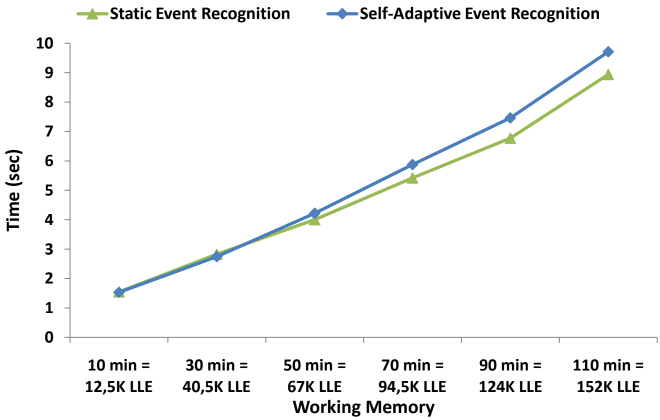
$noisy(Bus)$ **terminated** if
$move(Bus, Lon_B, Lat_B, 0)$ **happens**,
$close(Lon_B, Lat_B, Lon_S, Lat_S)$,
$\neg$ ($scatsReportedCongestion(Lon_S, Lat_S)$ **holds**)

Self-Adaptive Event Recognition: Discard Temporarily Unreliable Event Sources

$busReportedCongestion(Lon, Lat)$ **initiated** iff
  $move(Bus, Lon_B, Lat_B, 1)$ **happens**,
  $\neg$ ($noisy(Bus)$ **holds**),
  $close(Lon_B, Lat_B, Lon, Lat)$

$busReportedCongestion(Lon, Lat)$ **terminated** iff
  $move(Bus, Lon_B, Lat_B, 0)$ **happens**,
  $\neg$ ($noisy(Bus)$ **holds**),
  $close(Lon_B, Lat_B, Lon, Lat)$

Self-Adaptive Event Recognition in Dublin



Event Recognition Under Uncertainty: Summary

- Uncertainty in the input:
  - Probabilistic reasoning.
  - Using variety for veracity (when possible).
- Uncertainty in the HLE definitions:
  - Probabilistic reasoning.
- But:
  - We are still missing a framework for real-time, probabilistic event recognition.

Tutorial Structure

- Temporal reasoning systems.
- Event recognition under uncertainty.
- Machine learning for event recognition.
- Open issues.

Machine Learning for Event Recognition

Manual development of HLE definitions:

- Time consuming.
- Error-prone.

Automated construction for HLE definitions:

- Learn complex HLE definitions
  - Structure learning
- Learn from noisy data
  - Parameter learning
- Learn with incomplete or missing annotation
  - Semi-supervised, unsupervised learning
- Learn from large amounts of data
  - Scalable algorithms, incremental learning

Learning the Structure of HLE Definitions

Inductive Logic Programming (ILP):

- Input:
  - LLE streams annotated with HLE
    - Examples E^+, E^-.
  - Event recognition engine
    - Background knowledge B.
  - Syntax of event recognition language
    - Language bias M.
- Output:
  - A HLE definition
    - Hypothesis H in the language of M such that $B \cup H$ entails all positive and none of the negative examples.

Learning the Structure of HLE Definitions with ILP

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**,
- $|O_1 - O_2| < threshold$

↑

Background Knowledge

↑

Examples

. . .

$+$

moving(*alice*, *bob*) **holdsAt** 10

walking(*alice*) **happensAt** 10,
walking(*bob*) **happensAt** 10,
close(*alice*, *bob*) **holdsAt** 10,
orientation(*alice*) $= O_1$ **holdsAt** 10,
orientation(*bob*) $= O_2$ **holdsAt** 10,
$|O_1 - O_2| < threshold$

. . .

$-$

moving(*mary*, *jim*) **not holdsAt** 10

standing(*mary*) **happensAt** 10,
running(*jim*) **happensAt** 10,
close(*mary*, *jim*) **not holdsAt** 10,
orientation(*mary*) $= O_1$ **holdsAt** 10,
orientation(*jim*) $= O_2$ **holdsAt** 10,
$|O_1 - O_2| > threshold$

. . .

Learning HLE definitions with ILP

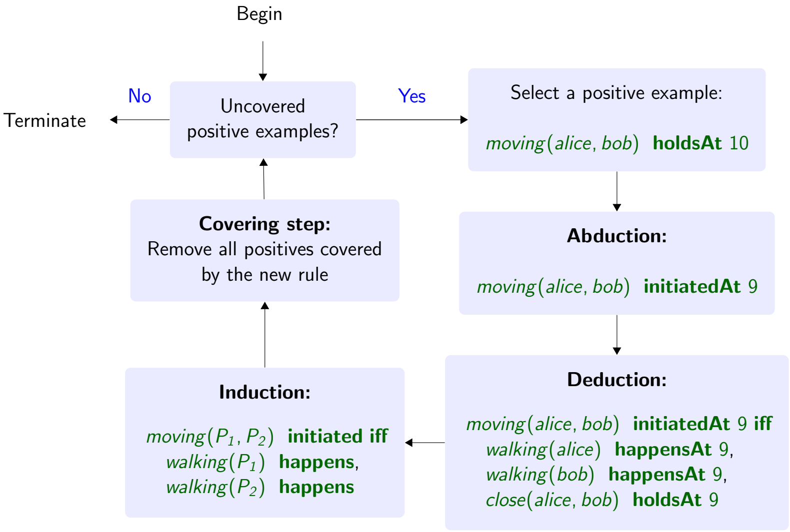
Non-Observational Predicate Learning:

- Supervision
  - **holdsAt**
- Target
  - **initiated, terminated**
- Traditional ILP systems cannot handle this

Solution:

- Obtain missing supervision by computing possible explanations of the examples (Abduction).

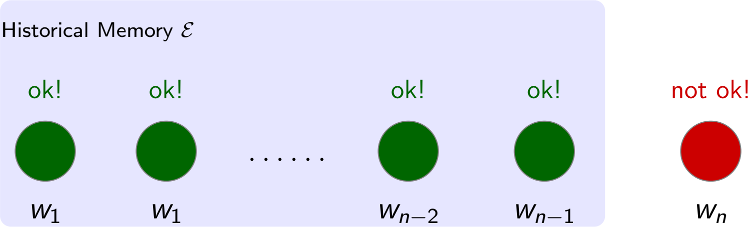
eXtended Hybrid Abductive-Inductive Learning – XHAIL



Incremental Learning

Given:

- A LLE stream $\mathcal{E}$ annotated with HLE (historical memory)
- A HLE definition H which is correct w.r.t $\mathcal{E}$
- A new LLE batch in which H is incorrect



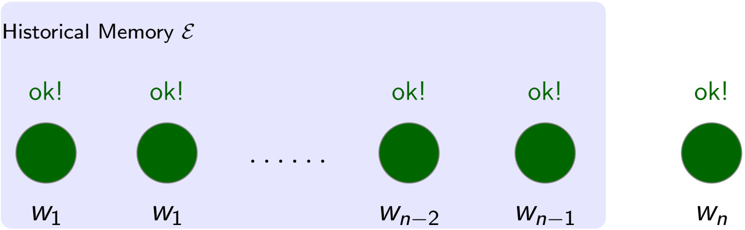
H :

fighting(P_1, P_2) **initiated iff**
active(P_1) **happens**,
abrupt(P_2) **happens**

Incremental Learning

Goal:

- Revise H to an H' that is correct w.r.t all examples



H' :

fighting(P_1, P_2) **initiated iff**	*fighting*(P_1, P_2) **initiated iff**
active(P_1) **happens**,	*abrupt*(P_1) **happens**,
abrupt(P_2) **happens**,	*abrupt*(P_2) **happens**,
close(P_1, P_2) **holds**	*close*(P_1, P_2) **holds**

Incremental Learning

Specialisation:

- Reject negative examples

H :

fighting(P_1, P_2) **initiated iff**
active(P_1) **happens**,
abrupt(P_2) **happens**

→

H' :

fighting(P_1, P_2) **initiated iff**
active(P_1) **happens**,
abrupt(P_2) **happens**,
close(P_1, P_2) **holds**

Incremental Learning

Generalisation:

- Cover more positive examples

H :

fighting(P_1, P_2) **initiated iff**
  active(P_1) **happens**,
  abrupt(P_2) **happens**,
  close(P_1, P_2) **holds**

↓

H' :

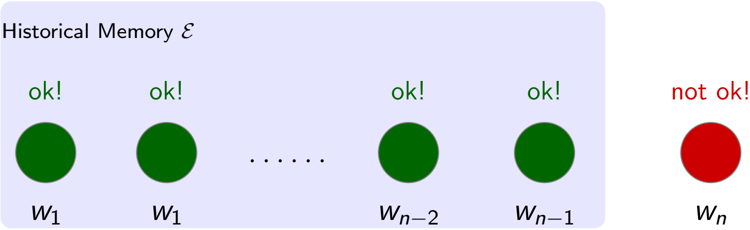
fighting(P_1, P_2) **initiated iff**
  active(P_1) **happens**,
  abrupt(P_2) **happens**,
  close(P_1, P_2) **holds**

fighting(P_1, P_2) **initiated iff**
  abrupt(P_1) **happens**,
  abrupt(P_2) **happens**,
  close(P_1, P_2) **holds**

Incremental Learning is Hard

Example:

- Specialise a HLE definition



Negative examples covered

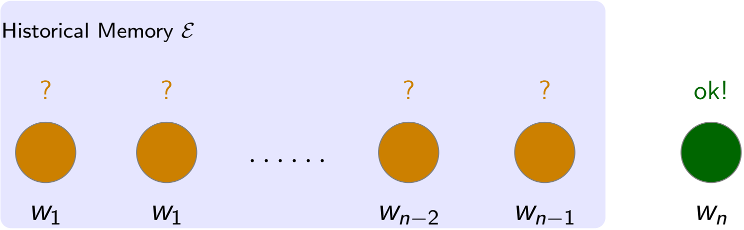
H :

fighting(P_1, P_2) **initiated iff**
active(P_1) **happens**,
abrupt(P_2) **happens**

Incremental Learning is Hard

Example:

- Specialise a HLE definition



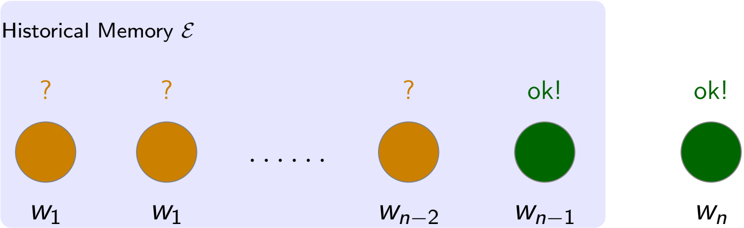
H' :

fighting(P_1, P_2) **initiated iff**
active(P_1) **happens**,
abrupt(P_2) **happens**,
close(P_1, P_2) **holds**

Incremental Learning is Hard

Example:

- Specialise a HLE definition



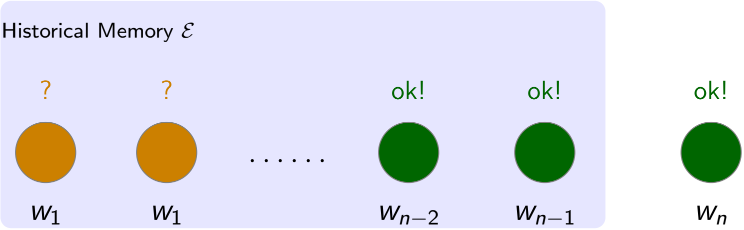
H' :

fighting(P_1, P_2) **initiated iff**
  active(P_1) **happens**,
  abrupt(P_2) **happens**,
  close(P_1, P_2) **holds**

Incremental Learning is Hard

Example:

- Specialise a HLE definition



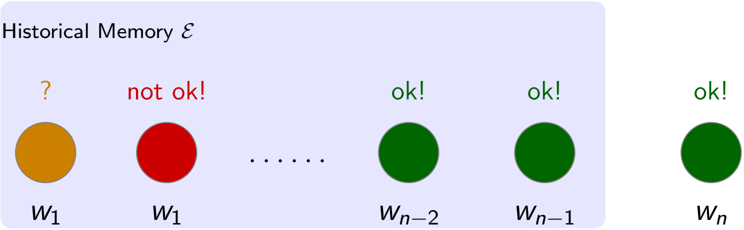
H' :

fighting(P_1, P_2) **initiated iff**
  active(P_1) **happens**,
  abrupt(P_2) **happens**,
  close(P_1, P_2) **holds**

Incremental Learning is Hard

Example:

- Specialise a HLE definition



Positive examples not covered

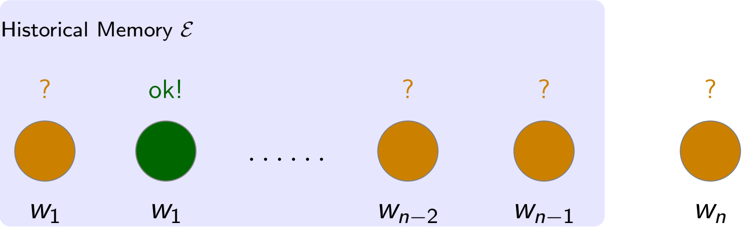
H' :

fighting(P_1, P_2) **initiated iff**
active(P_1) **happens**,
abrupt(P_2) **happens**,
close(P_1, P_2) **holds**

Incremental Learning is Hard

Example:

- Specialise a HLE definition



We must start all over again...

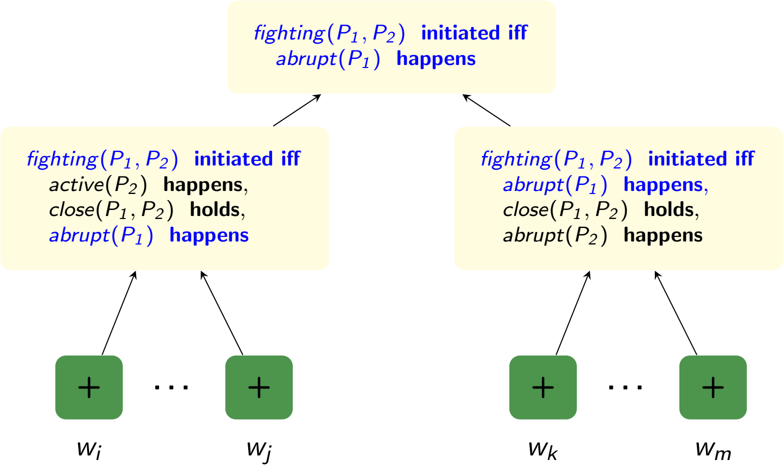
H'' :

fighting(P_1, P_2) **initiated iff**
active(P_1) **happens**,
abrupt(P_2) **happens**,
close(P_1, P_2) **holds**

fighting(P_1, P_2) **initiated iff**
abrupt(P_1) **happens**

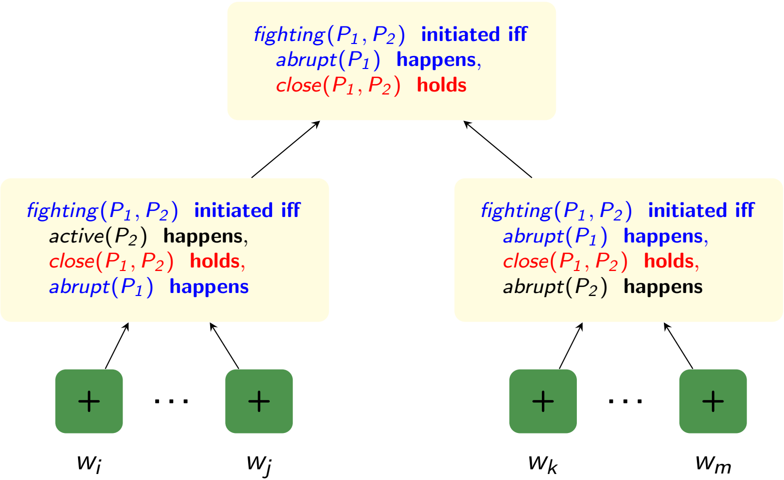
Efficient Incremental Learning: Support Set

- While constructing a HLE definition, summarize the positive examples it covers so far.
- This memory can be used for specialisation without having to look back.



Support Set

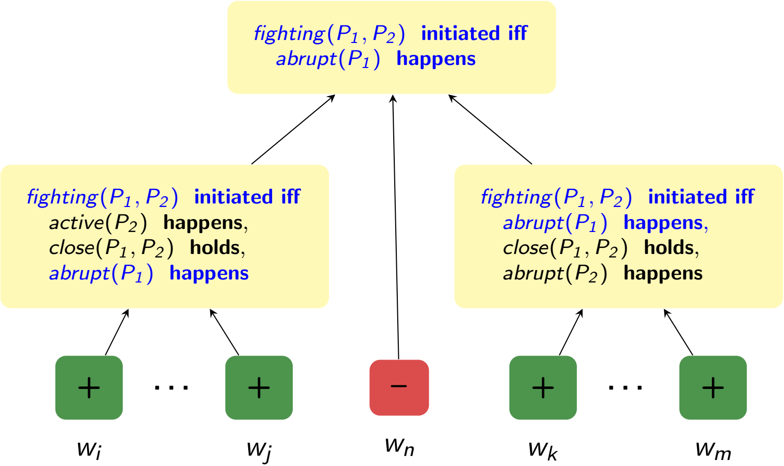
- To revise a HLE definition while preserving the positive examples it covers
  - It suffices for the revision to subsume the support set



Support Set Example

Find the smallest set of "supported" specialisations such that:

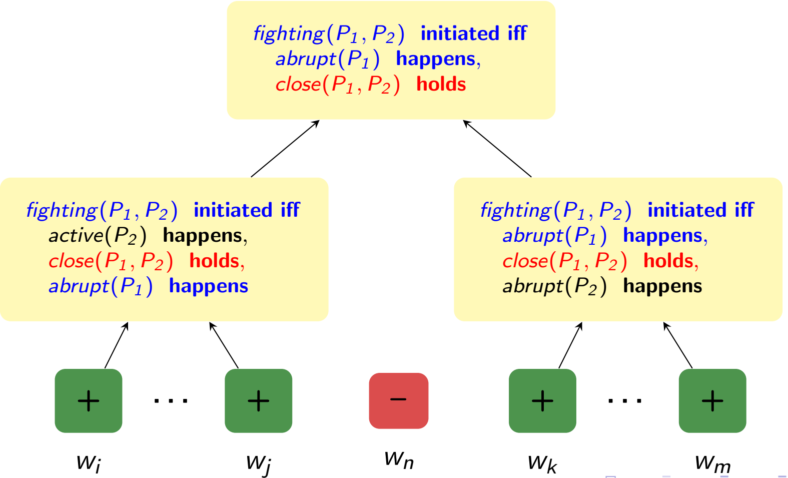
- All specialisations subsume the support set.
- Each specialisation rejects the negative examples.



Support Set Example

Find the smallest set of "supported" specialisations such that:

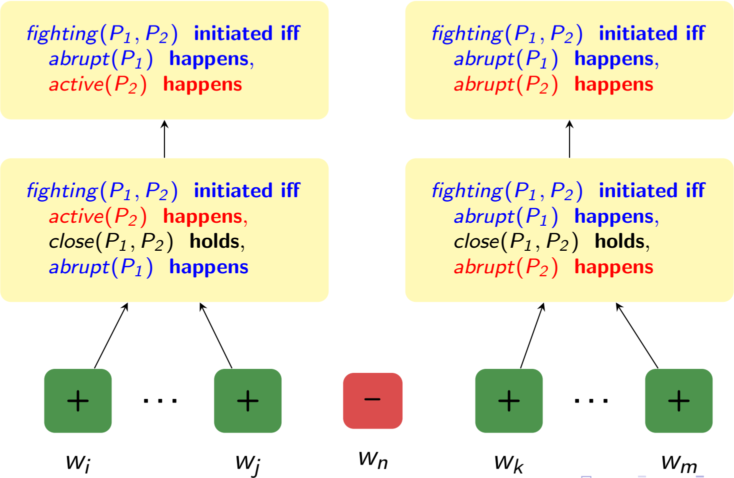
- All specialisations subsume the support set.
- Each specialisation rejects the negative examples.
- A single specialisation may suffice.



Support Set Example

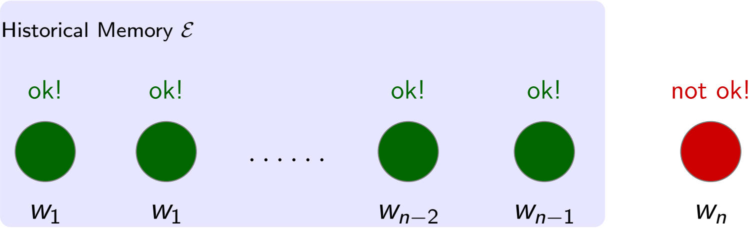
Find the smallest set of "supported" specialisations such that:

- All specialisations subsume the support set.
- Each specialisation rejects the negative examples.
- The HLE definition may need to "split".



What do we achieve?

- Without the support set

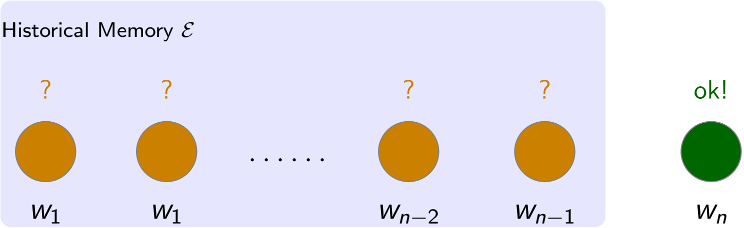


Negative examples covered

H : *fighting*(P_1, P_2) **initiated iff** *abrupt*(P_1) **happens**

What do we achieve?

- Without the support set

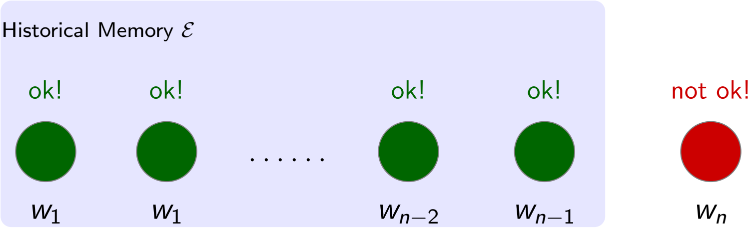


H' :

fighting(P_1, P_2) **initiated iff**
abrupt(P_1) **happens**,
active(P_2) **happens**

What do we achieve?

- With the support set

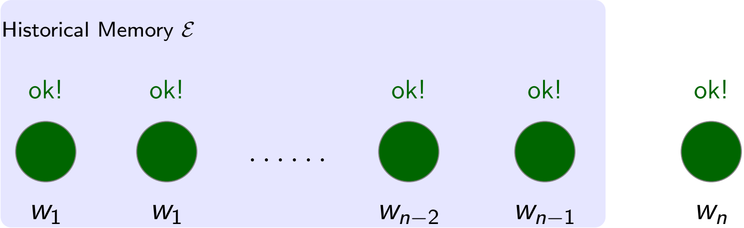


Negative examples covered

H : $\textit{fighting}(P_1, P_2)$ **initiated iff** $\textit{abrupt}(P_1)$ **happens**

What do we achieve?

- With the support set
  - Reject negative examples locally, preserve positive examples globally.
  - Reasoning within the support set, avoid redundant inference in the historical memory
  - At most one pass over the historical memory is required.



H' :

fighting(P_1, P_2) **initiated iff**
abrupt(P_1) **happens**,
active(P_2) **happens**

fighting(P_1, P_2) **initiated iff**
abrupt(P_1) **happens**,
abrupt(P_2) **happens**

Machine Learning for Event Recognition: Summary

- Automated construction & refinement of HLE definitions
  - Taking advantage of very large datasets.
  - Dealing with partial supervision.
- But:
  - We also need to deal with noise
    - Simultaneous optimisation of structure and parameters.

Tutorial Structure

- Temporal reasoning systems.
- Event recognition under uncertainty.
- Machine learning for event recognition.
- Open issues.

Open Issues

- Machine learning under uncertainty.
- Real-time event recognition under uncertainty.
- Distributed event recognition.
- Multi-scale temporal aggregation of events.
- Event forecasting under uncertainty.
- User-friendly authoring tools enabling non-programmers to use event recognition & forecasting.

Tutorial Resources

- Alexander Artikis, Anastasios Skarlatidis, Francois Portet, Georgios Paliouras: Logic-based event recognition. Knowledge Engineering Review 27(4): 469-506 (2012).
- Software, datasets, slides & papers at `cer.iit.demokritos.gr`