Logic-Based Event Recognition

Georgios Paliouras

Institute of Informatics & Telecommunications,
NCSR "Demokritos", Greece
paliourg@iit.demokritos.gr

Joint work with

- Alexander Artikis (efficient event recognition)
- Jason Filippou (event recognition under uncertainty)
- Nikos Katzouris (structure learning for event recognition)
- Anastasios Skarlatidis (event recognition with MLNs)

Event Recognition

Input:

- Symbolic representation of time-stamped, *low-level events* (LLE).
- LLE come from different sources/sensors.
- Very large amounts of input LLE.

Output:

- *High-level events* (HLE), i.e. temporal/spatial/logical combinations of LLE and/or HLE.
- Humans understand HLE easier than LLE.

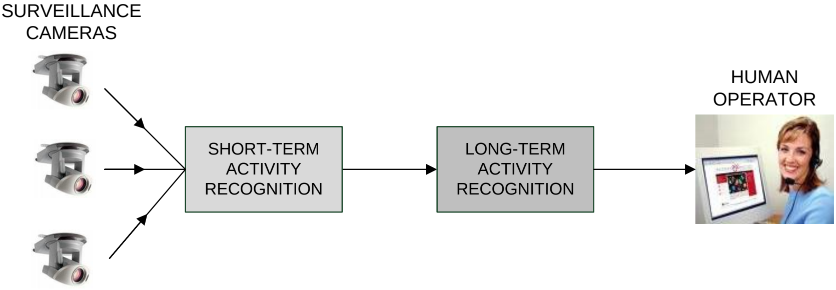
Scope:

- Symbolic event recognition, not signal processing.

Event recognition can be:

- Run-time.
- Retrospective.

Event Recognition for Public Space Surveillance



Event Recognition for Public Space Surveillance

- Input: short-term activities. Eg: someone is walking, running, stays inactive, becomes active, moves abruptly, etc.
- Output: long-term activities. Eg: two people are meeting, someone leaves an unattended object, two people are fighting, etc.

A long-term activity is recognised given a series of short-term activities that satisfy a set of temporal, logical and spatial constraints.

Event Recognition for Public Space Surveillance

Input
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 $p(id_5) = (17.88, -11.90)$	
...	

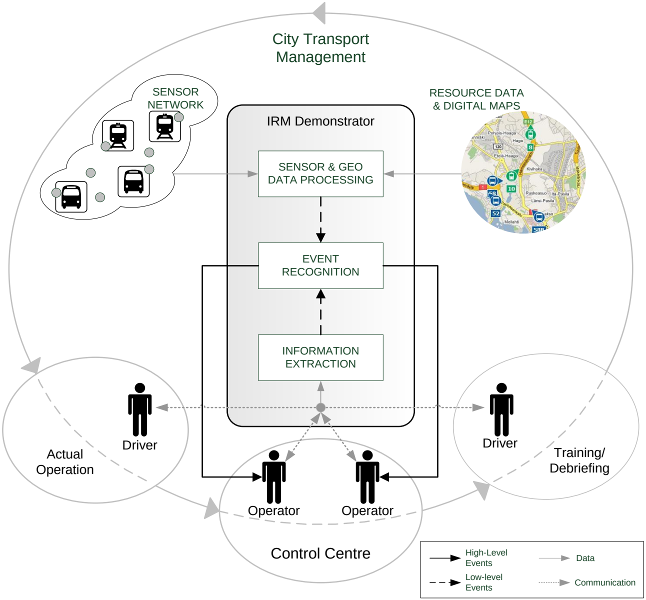
Event Recognition for Public Space Surveillance

Input	Output
$420\ \mathit{active}(\mathit{id}_4)$	$[420, 480]\ \mathit{fighting}(\mathit{id}_4, \mathit{id}_5)$
$420\ p(\mathit{id}_4) = (10.88, -71.90)$	
$420\ \mathit{inactive}(\mathit{id}_3)$	
$420\ p(\mathit{id}_3) = (5.8, -50.90)$	
$420\ \mathit{abrupt}(\mathit{id}_5)$	
$420\ p(\mathit{id}_5) = (11.80, -72.80)$	
$420\ \mathit{active}(\mathit{id}_6)$	
$420\ p(\mathit{id}_6) = (7.8, -52.90)$	
$480\ \mathit{abrupt}(\mathit{id}_4)$	
$480\ p(\mathit{id}_4) = (20.45, -12.90)$	
$480\ \mathit{abrupt}(\mathit{id}_5)$	
$480\ p(\mathit{id}_5) = (17.88, -11.90)$	
...	

Event Recognition for Public Space Surveillance

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

Event Recognition for City Transport Management



Event Recognition for City Transport Management

- Input: LLE coming from GPS, accelerometers, internal thermometers, microphones, internal cameras.
- Output: HLE concerning passenger and driver safety, passenger and driver comfort, passenger satisfaction, etc.
- Details at `http://www.ict-pronto.org/`

Event Recognition for City Transport Management

	Input	Output
200	scheduled stop enter	
215	scheduled stop leave	
$[215, 400]$	abrupt acceleration	
$[500, 600]$	very sharp turn	
700	late stop enter	
705	passenger density change to high	
715	scheduled stop leave	
820	scheduled stop enter	
815	passenger density change to low	
...		

Event Recognition for City Transport Management

Input		Output	
200	scheduled stop enter		
215	scheduled stop leave	215	punctual
$[215, 400]$	abrupt acceleration	$[215, 400]$	uncomfortable driving
$[500, 600]$	very sharp turn	$[500, 600]$	unsafe driving
700	late stop enter		
705	passenger density change to high		
715	scheduled stop leave		
820	scheduled stop enter		
815	passenger density change to low		
...			

Event Recognition for City Transport Management

Input		Output	
200	scheduled stop enter		
215	scheduled stop leave	215	punctual
$[215, 400]$	abrupt acceleration	$[215, 400]$	uncomfortable driving
$[500, 600]$	very sharp turn	$[500, 600]$	unsafe driving
700	late stop enter	700	non-punctual
705	passenger density change to high	$\mathit{since}(705)$	reducing passenger comfort
715	scheduled stop leave		
820	scheduled stop enter		
815	passenger density change to low	$[705, 815]$	reducing passenger comfort
...			

Overview

Event Calculus

Event Recognition using the Event Calculus

- Run-time event recognition with caching.
- Probabilistic event calculus (ProbLog, MLNs).

Adaptable Event Recognition

- MLN weight learning.
- Incremental structure learning with abduction.

Event Calculus

- Formalism for representing events and their effects.
- Originally expressed as a logic (Prolog) program.
- Built-in representation of law of inertia.

Predicate	Meaning
happensAt$(E,\ T)$	Event E is occurring at time T
initially$(F=V)$	The value of fluent F is V at time 0
initiatedAt$(F=V,\ T)$	At time T a period of time for which $F=V$ is initiated
terminatedAt$(F=V, T)$	At time T a period of time for which $F=V$ is terminated
holdsAt$(F=V,\ T)$	The value of fluent F is V at time T
holdsFor$(F=V,\ I)$	I is the list of the maximal intervals for which $F=V$ holds continuously

Event Calculus

$$\text{initially}(\ \mathit{punctuality}(\_,\_) = \mathit{punctual}\ )$$

$$\begin{aligned}&\text{initiatedAt}(\ \mathit{punctuality}(\mathit{Id}, \mathit{Vehicle}) = \mathit{punctual},\ \ T\ ) \leftarrow\\ &\qquad \text{happensAt}(\ \mathit{punctual}(\mathit{Id}, \mathit{Vehicle}),\ \ T\ )\end{aligned}$$

$$\begin{aligned}&\text{initiatedAt}(\ \mathit{punctuality}(\mathit{Id}, \mathit{Vehicle}) = \mathit{non\_punctual},\ \ T\ ) \leftarrow\\ &\qquad \text{happensAt}(\ \mathit{non\_punctual}(\mathit{Id}, \mathit{Vehicle}),\ \ T\ )\end{aligned}$$

$$\begin{aligned}&\text{happensAt}(\ \mathit{punctuality\_change}(\mathit{Id}, \mathit{Vehicle}, \mathit{non\_punctual}),\ \ T\ ) \leftarrow\\ &\qquad \text{holdsFor}(\ \mathit{punctuality}(\mathit{Id}, \mathit{Vehicle}) = \mathit{non\_punctual},\ \ I\ ),\\ &\qquad (T, \_) \in I,\\ &\qquad T \neq 0\end{aligned}$$

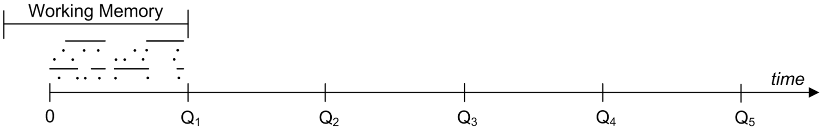
Event Calculus

$$\text{holdsFor}(\ \mathit{driving\_quality}(\mathit{Id}, \mathit{Vehicle}) = \mathit{low},\ \ \mathit{LQDI}\ ) \leftarrow$$
$$\text{holdsFor}(\ \mathit{punctuality}(\mathit{Id}, \mathit{Vehicle}) = \mathit{non\_punctual},\ \ \mathit{NPI}\ ),$$
$$\text{holdsFor}(\ \mathit{driving\_style}(\mathit{Id}, \mathit{Vehicle}) = \mathit{unsafe},\ \ \mathit{USI}\ ),$$
$$\text{union\_all}(\ [\mathit{NPI}, \mathit{USI}],\ \ \mathit{LQDI}\ )$$

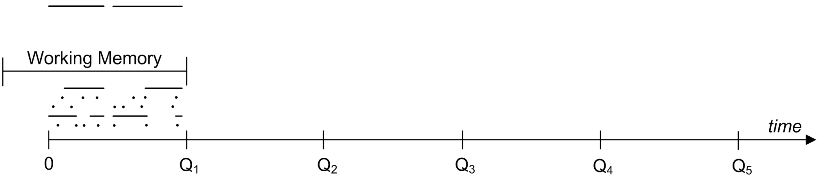
$$\text{holdsFor}(\ \mathit{driving\_quality}(\mathit{Id}, \mathit{Vehicle}) = \mathit{medium},\ \ \mathit{MQDI}\ ) \leftarrow$$
$$\text{holdsFor}(\ \mathit{punctuality}(\mathit{Id}, \mathit{Vehicle}) = \mathit{punctual},\ \ \mathit{PunctualI}\ ),$$
$$\text{holdsFor}(\ \mathit{driving\_style}(\mathit{Id}, \mathit{Vehicle}) = \mathit{uncomfortable},\ \ \mathit{UCI}\ ),$$
$$\text{intersect\_all}(\ [\mathit{PunctualI}, \mathit{UCI}],\ \ \mathit{MQDI}\ )$$

$$\text{holdsFor}(\ \mathit{driving\_quality}(\mathit{Id}, \mathit{Vehicle}) = \mathit{high},\ \ \mathit{HQDI}\ ) \leftarrow$$
$$\text{holdsFor}(\ \mathit{punctuality}(\mathit{Id}, \mathit{Vehicle}) = \mathit{punctual},\ \ \mathit{PunctualI}\ ),$$
$$\text{holdsFor}(\ \mathit{driving\_style}(\mathit{Id}, \mathit{Vehicle}) = \mathit{unsafe},\ \ \mathit{USI}\ ),$$
$$\text{holdsFor}(\ \mathit{driving\_style}(\mathit{Id}, \mathit{Vehicle}) = \mathit{uncomfortable},\ \ \mathit{UCI}\ ),$$
$$\text{relative\_complement\_all}(\ \mathit{PunctualI},\ \ [\mathit{USI}, \mathit{UCI}],\ \ \mathit{HQDI}\ )$$

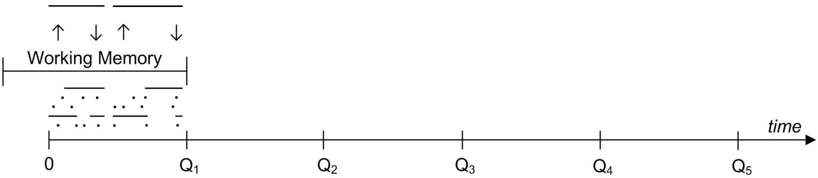
Event Calculus: Run-Time Event Recognition



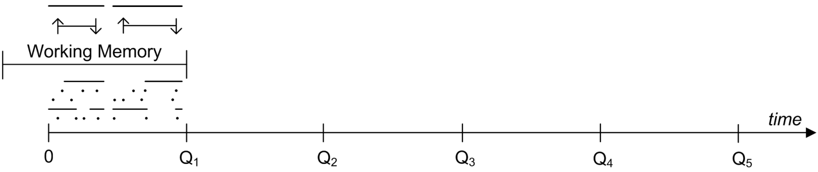
Event Calculus: Run-Time Event Recognition



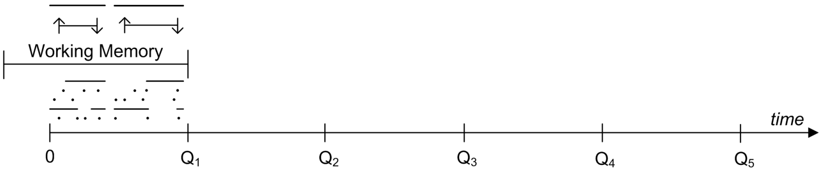
Event Calculus: Run-Time Event Recognition

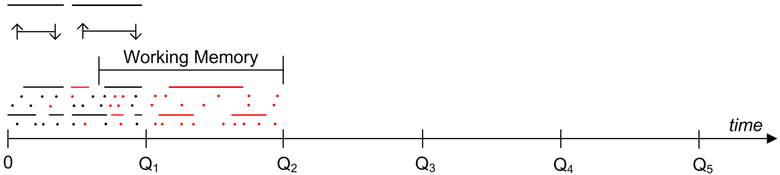


Event Calculus: Run-Time Event Recognition

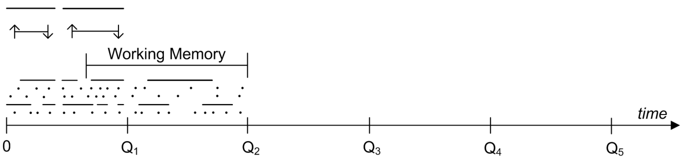


Event Calculus: Run-Time Event Recognition

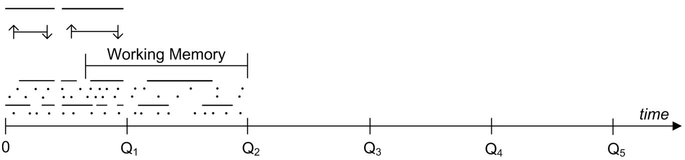


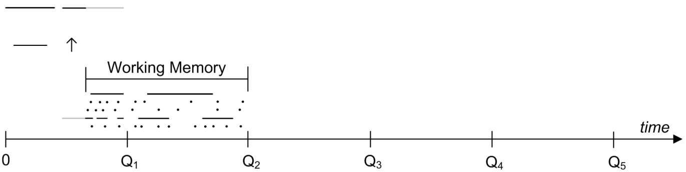


Event Calculus: Run-Time Event Recognition

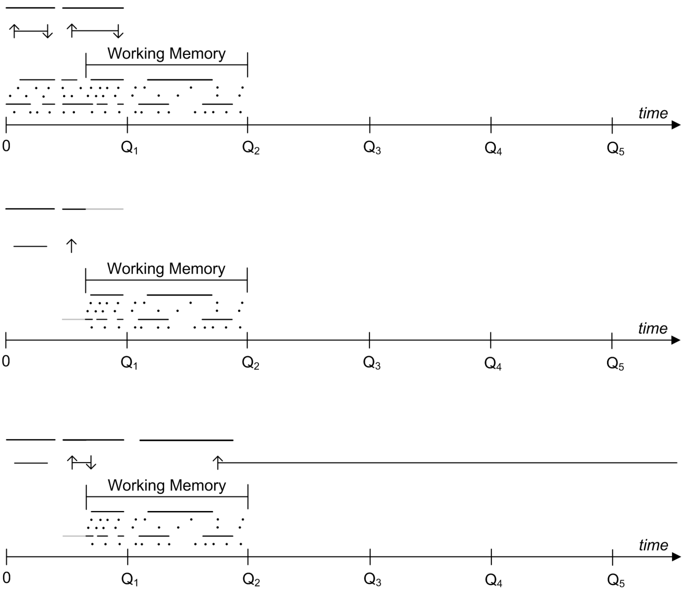


Event Calculus: Run-Time Event Recognition

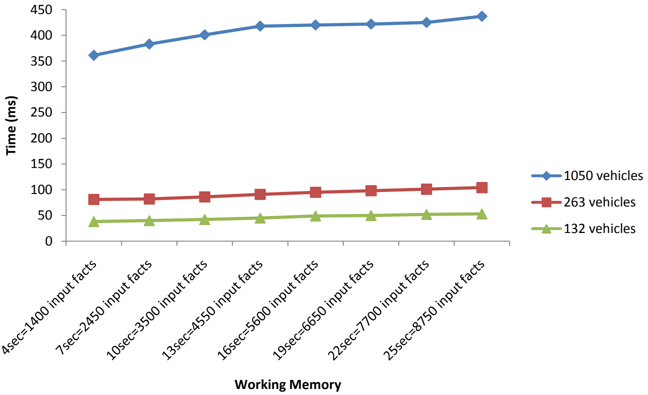




Event Calculus: Run-Time Event Recognition



Run-Time Event Recognition for City Transport Management



Probabilistic Event Recognition

Event recognition methods:

- Logic-based methods
- Probabilistic methods

Event recognition requirements:

- Formal representation language
- Handle uncertainty

Probabilistic Event Calculus combines:

- Event Calculus — representation
- Probabilistic inference (ProbLog or MLNs) — uncertainty

Probabilistic Event Recognition: ProbLog

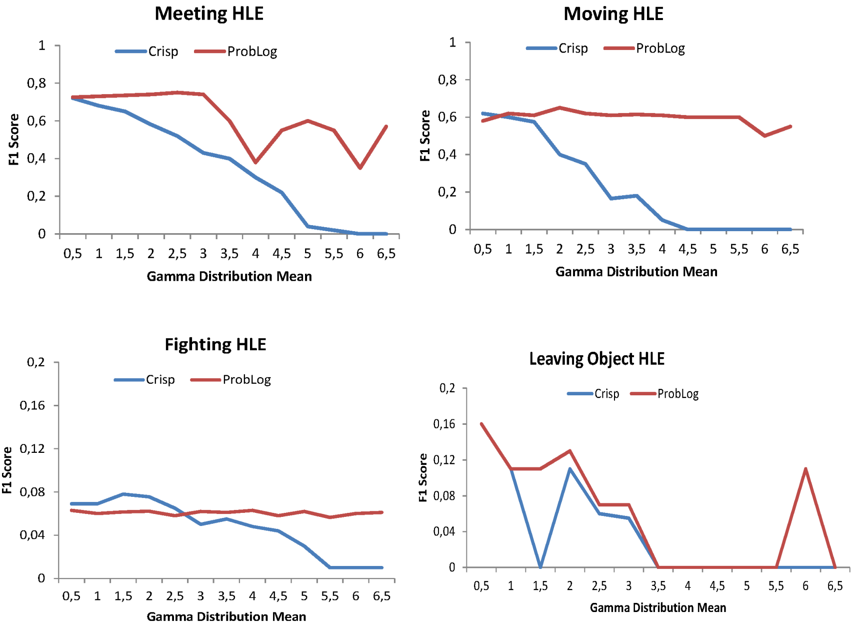
- A Probabilistic Logic Programming language.
- Allows for independent "probabilistic facts", i.e facts of form: *prob::fact*.
- *Prob* indicates the probability that *fact* is part of a possible world.
- Rules are written as in classic Prolog: $Head \leftarrow Body$
- The probability of a query q imposed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

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

Probabilistic Event Recognition: 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 :: $p(id_3) = (24.78, -18.77)$	
...	

Preliminary Experimental Results (ProbLog)



Markov Logic Networks (MLN) — in a nutshell

- First-order logic $\rightarrow$ set of hard constraints
- Syntactically: weighted first-order logic formulas (F_i, w_i)
- Semantically: (F_i, w_i) represents a probability distribution over possible worlds (or Herbrand interpretations)

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

- possible world
- partition function
- number of satisfied ground formulas

A world violating formulas becomes less probable, but not impossible!

Representing Event Calculus in MLN

Some axioms of Event Calculus in First-Order Logic:

$$
\left.\begin{aligned}
holdsAt(F,\ T) \leftarrow\ & happens(E,\ T_0) \wedge \\
& initiates(E,\ F,\ T_0) \wedge \\
& T_0 < T \wedge \\
& \neg clipped(F,\ T_0,\ T)
\end{aligned}\right\} \quad F \times E \times\ T \times T_0
$$

$$
\left.\begin{aligned}
clipped(F,\ T_0,\ T_1) \leftrightarrow \exists\ E, T\ & \\
& happens(E,\ T) \wedge \\
& T_0 \leq T < T_1 \wedge \\
& terminates(E,\ F,\ T)
\end{aligned}\right\} \quad F \times E \times\ T \times T_0 \times T_1
$$

Representing Event Calculus in MLN

Some axioms of Event Calculus in First-Order Logic:

$$\left.\begin{aligned} holdsAt(F,\ T) \leftarrow\ & happens(E,\ T_0) \wedge \\ & initiates(E,\ F,\ T_0) \wedge \\ & T_0 < T \wedge \\ & \neg clipped(F,\ T_0,\ T) \end{aligned}\right\} \quad F \times E \times \boxed{T \times T_0}$$

$$\left.\begin{aligned} clipped(F,\ T_0,\ T_1) \leftrightarrow\ & \boxed{\exists\ E, T} \\ & happens(E,\ T) \wedge \\ & T_0 \leq T < T_1 \wedge \\ & terminates(E,\ F,\ T) \end{aligned}\right\} \quad F \times E \times \boxed{T \times T_0 \times T_1}$$

- Huge number of groundings
- Combinatorial explosion

Simplified Discrete Event Calculus

When a fluent holds:

$$\left.\begin{array}{l} holdsAt(F, T+1) \leftarrow \\ \qquad initiatedAt(F, T) \end{array}\right\} \quad F \times T$$

$$\left.\begin{array}{l} holdsAt(F, T+1) \leftarrow \\ \qquad holdsAt(F, T) \wedge \\ \qquad \neg teminatedAt(F, T) \end{array}\right\} \quad F \times T$$

When a fluent does not hold:

$$\left.\begin{array}{l} \neg holdsAt(F, T+1) \leftarrow \\ \qquad terminatedAt(F, T) \end{array}\right\} \quad F \times T$$

$$\left.\begin{array}{l} \neg holdsAt(F, T+1) \leftarrow \\ \qquad \neg holdsAt(F, T) \wedge \\ \qquad \neg initiatedAt(F, T) \end{array}\right\} \quad F \times T$$

Example: HLE definition

When the fluent 'meeting' is initiated:

$$\begin{aligned}\textbf{initiatedAt}(\textbf{meeting}, \textbf{T}) \leftarrow\ & \\ & happens(event_1, T) \wedge \\ & \neg happens(event_2, T) \wedge \\ & distance(close, T)\end{aligned}$$

$$\begin{aligned}\textbf{initiatedAt}(\textbf{meeting}, \textbf{T}) \leftarrow\ & \\ & happens(event_3, T) \wedge \\ & \neg happens(event_1, T) \wedge \\ & \neg happens(event_2, T) \wedge \\ & distance(close, T)\end{aligned}$$

When the fluent 'meeting' is terminated:

$$\begin{aligned}\textbf{terminatedAt}(\textbf{meeting}, \textbf{T}) \leftarrow\ & \\ & happens(event_4, T)\end{aligned}$$

...

Open-world semantics in MLN

Domain-dependent definitions:

- Conditions under which HLE are initiated or terminated
- Open-world assumption for non-evidence predicates: initiatedAt, terminatedAt and holdsAt

When something is happening that it is not defined in the domain-dependent definitions:

- Cannot determine whether a fluent holds or not
- Loss of the inertia
- This is also known as the *frame problem*
- Solution: predicate completion

Predicate completion and MLN

$$
\text{HLE definitions} = \begin{cases}
\mathbf{initiatedAt(meeting, T)} \leftarrow \\
\quad \mathit{happens}(\mathit{event}_1, T) \wedge \\
\quad \neg \mathit{happens}(\mathit{event}_2, T) \wedge \\
\quad \mathit{distance}(\mathit{close}, T) \\
\mathbf{initiatedAt(meeting, T)} \leftarrow \\
\quad \mathit{happens}(\mathit{event}_3, T) \wedge \\
\quad \neg \mathit{happens}(\mathit{event}_1, T) \wedge \\
\quad \neg \mathit{happens}(\mathit{event}_2, T) \wedge \\
\quad \mathit{distance}(\mathit{close}, T) \\
\ldots
\end{cases}
$$

Predicate completion and MLN

$$
\text{HLE definitions} = \left\{
\begin{array}{l}
\quad \mathbf{initiatedAt}(\mathbf{meeting}, \mathbf{T}) \leftarrow \\
\qquad \mathit{happens}(\mathit{event}_1, T) \wedge \\
\qquad \neg\mathit{happens}(\mathit{event}_2, T) \wedge \\
\qquad \mathit{distance}(\mathit{close}, T) \\
\quad \mathbf{initiatedAt}(\mathbf{meeting}, \mathbf{T}) \leftarrow \\
\qquad \mathit{happens}(\mathit{event}_3, T) \wedge \\
\qquad \neg\mathit{happens}(\mathit{event}_1, T) \wedge \\
\qquad \neg\mathit{happens}(\mathit{event}_2, T) \wedge \\
\qquad \mathit{distance}(\mathit{close}, T) \\
\ldots
\end{array}
\right.
$$

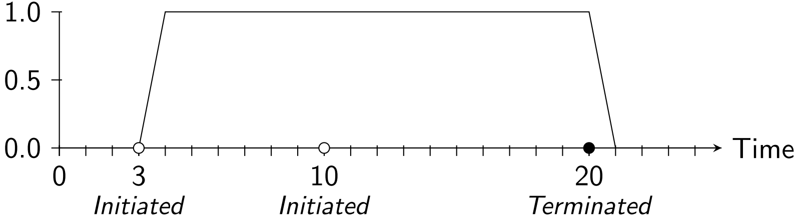
$$
\begin{array}{r}
\text{Completion constraints} = \\
\text{(automatically generated)}
\end{array}
\left\{
\begin{array}{l}
\quad \mathbf{initiatedAt}(\mathbf{meeting}, \mathbf{T}) \rightarrow \\
\qquad [\mathit{happens}(\mathit{event}_1, T) \wedge \\
\qquad \neg\mathit{happens}(\mathit{event}_2, T) \wedge \\
\qquad \mathit{distance}(\mathit{close}, T)\ ] \bigvee \\
\qquad [\mathit{happens}(\mathit{event}_3, T) \wedge \\
\qquad \neg\mathit{happens}(\mathit{event}_1, T) \wedge \\
\qquad \neg\mathit{happens}(\mathit{event}_2, T) \wedge \\
\qquad \mathit{distance}(\mathit{close}, T)\ ] \\
\ldots
\end{array}
\right.
$$

Predicate completion and MLN

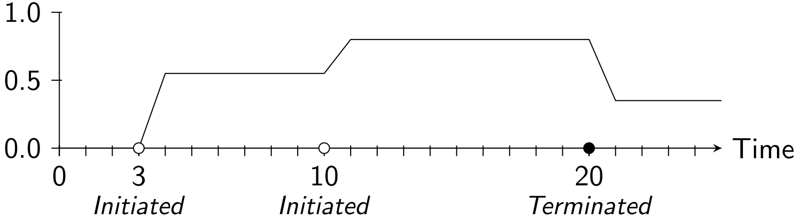
$$\text{HLE definitions} = \begin{cases} \mathbf{1.5} \quad \mathbf{initiatedAt}(\mathbf{meeting}, \mathbf{T}) \leftarrow \\ \qquad \mathit{happens}(\mathit{event}_1, T) \wedge \\ \qquad \neg \mathit{happens}(\mathit{event}_2, T) \wedge \\ \qquad \mathit{distance}(\mathit{close}, T) \\ \mathbf{0.25} \quad \mathbf{initiatedAt}(\mathbf{meeting}, \mathbf{T}) \leftarrow \\ \qquad \mathit{happens}(\mathit{event}_3, T) \wedge \\ \qquad \neg \mathit{happens}(\mathit{event}_1, T) \wedge \\ \qquad \neg \mathit{happens}(\mathit{event}_2, T) \wedge \\ \qquad \mathit{distance}(\mathit{close}, T) \\ \ldots \end{cases}$$

$$\begin{array}{l}\text{Completion constraints} = \\ \text{(automatically generated)}\end{array} \begin{cases} \mathbf{4.0} \quad \mathbf{initiatedAt}(\mathbf{meeting}, \mathbf{T}) \rightarrow \\ \qquad [\mathit{happens}(\mathit{event}_1, T) \wedge \\ \qquad \neg \mathit{happens}(\mathit{event}_2, T) \wedge \\ \qquad \mathit{distance}(\mathit{close}, T)\ ] \bigvee \\ \qquad [\mathit{happens}(\mathit{event}_3, T) \wedge \\ \qquad \neg \mathit{happens}(\mathit{event}_1, T) \wedge \\ \qquad \neg \mathit{happens}(\mathit{event}_2, T) \wedge \\ \qquad \mathit{distance}(\mathit{close}, T)\ ] \\ \ldots \end{cases}$$

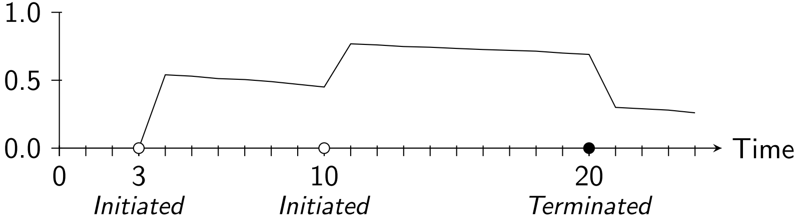
Inertia in probabilistic EC



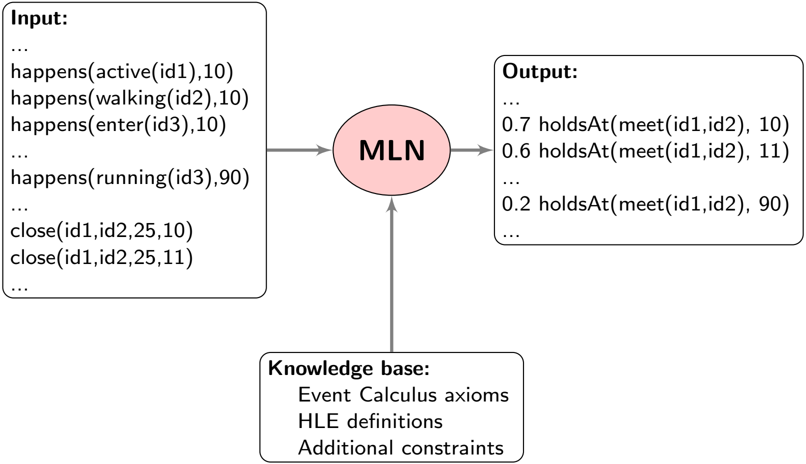
Inertia in probabilistic EC



Inertia in probabilistic EC



Experiments (MLN)



Experimental results (MLN)

- EC-LP:
  - Logic-programming based EC
  - Knowledge base of HLE for CAVIAR dataset
- Manualy adjusted weight values for the HLE *meeting*:
  - weak values — low confidence
  - strong values — high confidence
- DEC-MLN$_a$: soft-constrained HLE definitions
- DEC-MLN$_b$: soft-constrained HLE definitions and termination rules in the additional constraints

Method	TP	FP	FN	Precision	Recall
EC-LP	3099	2258	525	0.578	**0.855**
DEC-MLN$_a$	3048	1762	576	0.633	0.841
DEC-MLN$_b$	3048	1154	576	**0.725**	0.841

Source KB and dataset files can be found in http://www.iit.demokritos.gr/∼anskarl

MLN weight learning

- Manually adjusting the weight values is a tedious and error prone process
- Annotation is given in terms of ground holdsAt predicates

LLE	HLE
...	...
happens(walking(id1), 100)	
happens(walking(id2), 100)	
distance(close(id1, id2), 100)	holdsAt(moving(id1, id2), 100)
happens(walking(id1), 101)	
happens(walking(id2), 101)	
¬distance(close(id1, id2), 101)	holdsAt(moving(id1, id2), 101)
happens(walking(id1), 102)	
happens(walking(id2), 102)	
¬distance(close(id1, id2), 102)	¬holdsAt(moving(id1, id2), 102)
...	...

MLN Weight learning

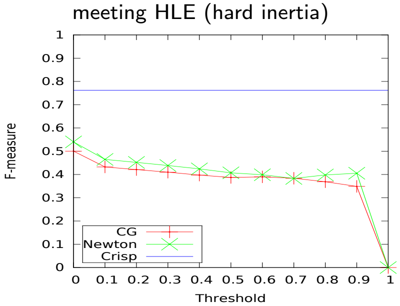
- Predicates initiatedAt/terminatedAt are not observable
- The KB can be further simplified by eliminating the initiatedAt/terminatedAt predicates — supervised learning
- Weight estimation using: Conjugate Gradient or Diagonal Newton

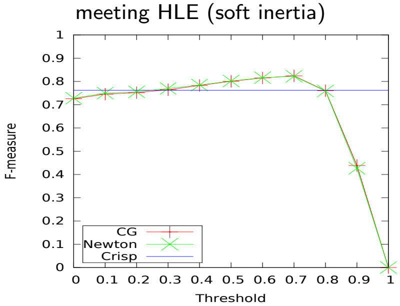
Conditional log-likelihood function:

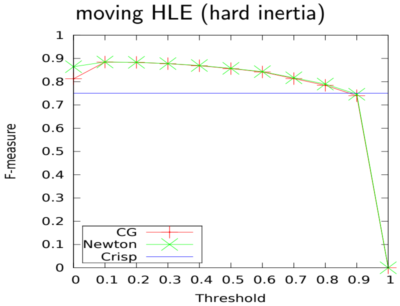
$$\log P(\boxed{Q = q} \mid \boxed{E = e}) = \sum_i w_i n_i(e, q) - \log Z_e$$

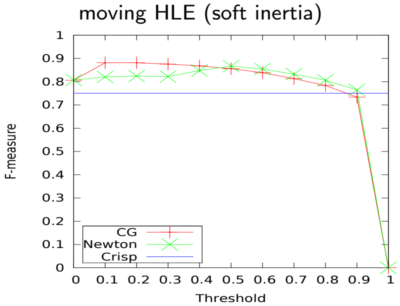
- Query prediates: HLE
- Evidence predicates: LLE

MLN weight learning results









Structure learning with abduction

Learning event structure (requirements):

- Represent and reason about time properties.
- Learning with partial supervision (Non-OPL - output clauses in which head literals do not appear in the training examples).
- Process large data streams.

Combining abduction with induction

Negation As Failure (stable model) semantics useful for:

- Handling of inertia.
- Learning with partial supervision (abduction).

Structure learning: Example of non-OPL learning

Compute a theory of the form:

$$\begin{aligned}&\mathit{initiatedAt}(\mathit{reducing\_passenger\_satisfaction}(\mathit{Id}, \mathit{VehicleType}) = \mathit{true}, T) \leftarrow\\ &\quad \mathit{happens}(\mathit{temperature\_change}(\mathit{Id}, \mathit{VehicleType}, \mathit{very\_warm}), T),\\ &\quad \mathit{holdsAt}(\mathit{punctuality}(\mathit{Id}, \mathit{VehicleType}) = \mathit{non\_punctual}), T)\end{aligned}$$

From examples of the form

$$\begin{aligned}&\mathit{holdsAt}(\mathit{reducing\_passenger\_satisfaction}(b1, \mathit{bus}) = \mathit{true}, 8)\\ &\mathit{not}\ \mathit{holdsAt}(\mathit{reducing\_passenger\_satisfaction}(b1, \mathit{bus}) = \mathit{true}, 16)\\ &\mathit{happens}(\mathit{temperature\_change}(b1, \mathit{bus}, \mathit{very\_warm}) = \mathit{true}, 8)\\ &\ldots\end{aligned}$$

And the axioms of the Event Calculus (as background knowledge)

Structure learning: three steps (XHAIL)

Example:

- **Abduction**

$$initiatedAt(reducing\_passenger\_satisfaction(b1, bus) = true, 8)$$

- **Deduction**

$$\begin{aligned} & initiatedAt(reducing\_passenger\_satisfaction(b1, bus) = true, 8) \leftarrow \\ & \quad happens(temperature\_change(b1, bus, very\_warm), 8), \\ & \quad holdsAt(punctuality(b1, bus) = non\_punctual), 8), \\ & \quad holdsAt(noise\_level(b1, bus) = high), 8) \end{aligned}$$

- **Induction**

$$\begin{aligned} & initiatedAt(reducing\_passenger\_satisfaction(Id, VehicleType) = true, T) \leftarrow \\ & \quad happens(temperature\_change(Id, VehicleType, very\_warm), T), \\ & \quad holdsAt(punctuality(Id, VehicleType) = non\_punctual), T) \end{aligned}$$

Structure learning from large data streams

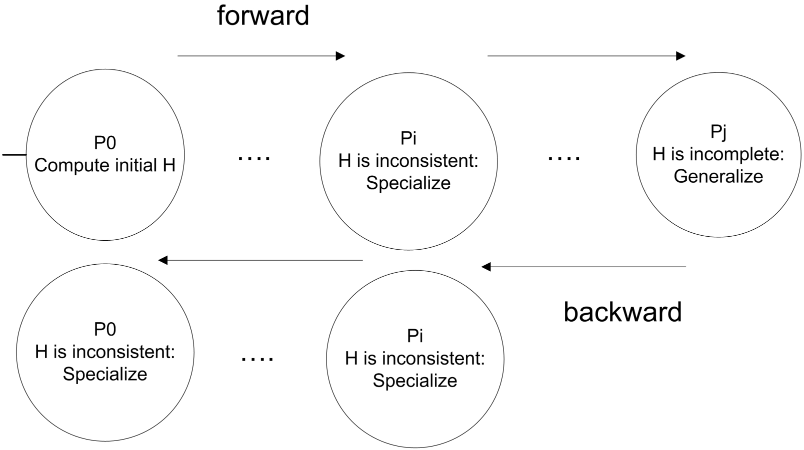
The problem

- All three steps implemented with Answer Set Programming.
- Computing answer sets is intractable in the general case.
  - **Complexity increases with the size of the data.**

Possible solution

- Break the dataset into smaller ones (partitions), e.g. time window.
- Learn a separate theory for each partition.
- Ensure completeness and consistency with all data (not minimality).

Structure learning from large data streams



Structure learning from large data streams

Preliminary hypothesis H**:** computed from p_0 (first partition)
Forward direction: new partition is being processed
Backward direction: past partitions are being revisited
Revision operators:

- Generalization (occurs in forward motion). Adds extra clauses to H. Fires backward specialization to retain consistency.
- Specialization (occurs in both forward and backward motion). Adds extra literals to inconsistent clauses.

Support Set: A structure associated with each clause $C \in H$.

- Compressive way to "remember" the examples that C covers from *each partition*.
- "Pool" for literals used to specialize an inconsistent clause.

Initial experiments: batch ($>$ 1 day), incremental ($<$ 1 minute)

Conclusions

- Event Calculus is a sound basis for event recognition.
- Very efficient event recognition can be achieved with caching.
- Uncertainty can be handled with probabilistic event calculus.
- Closed World semantics help.
- Probabilistic inertia in EC is particularly interesting.
- Weight learning in MLNs is effective.
- Structure learning with incremental theory revision is efficient.

Open issues

- Handling of intervals in probabilistic event recognition and learning.
- Simultaneous handling of uncertainty at the level of data and the knowledge base.
- Simultaneous optimization of structure and weights.
- Efficient semi-supervised learning in MLNs.
- Interaction with signal processing, providing low-level events.