Logic-Based Event Recognition

Alexander Artikis[1], Georgios Paliouras[1], François Portet[2] and Anastasios Skarlatidis[1 3]

[1]Institute of Informatics & Telecommunications,
NCSR "Demokritos", Greece
[2]Laboratoire d'Informatique de Grenoble, Grenoble Universités, France
[3]Department of Information & Communication Systems Engineering,
University of the Aegean, Greece

{a.artikis, paliourg, anskarl}@iit.demokritos.gr,
Francois.Portet@imag.fr

INTRODUCTION

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. combinations of LLE and/or HLE.
- Humans understand HLE easier than LLE.

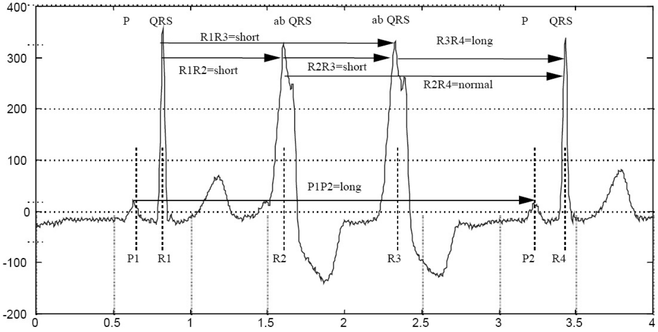
Tutorial scope:

- Symbolic event recognition, not signal processing.

Event recognition can be:

- On-line (run-time).
- Off-line (retrospective).

Medical Applications



- 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.

Medical Applications

Input
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]
. . .

Medical Applications

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]	
...	

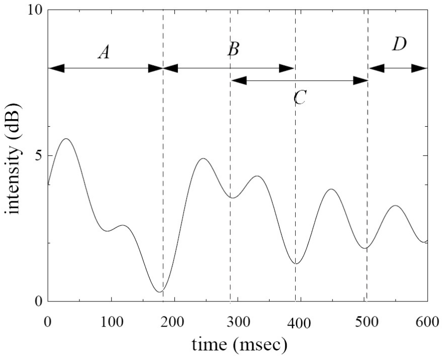
Medical Applications

Input
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]
...

Medical Applications

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



- 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.

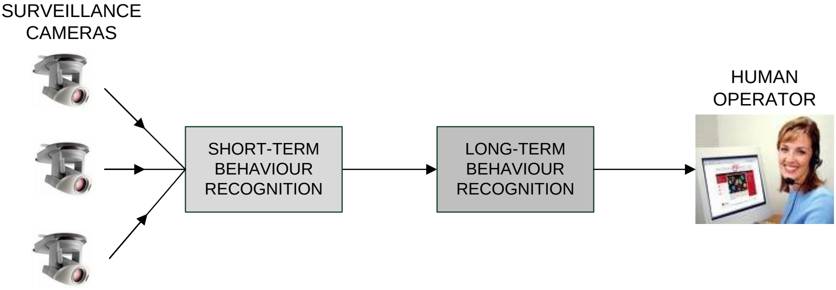
Humpback Whale Song Recognition

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 Public Space Surveillance



Event Recognition for Public Space Surveillance

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

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

Event Recognition for Public Space Surveillance

Input
340 $\mathit{inactive}(id_0)$
340 $p(id_0) = (20.88, -11.90)$
340 $\mathit{appear}(id_0)$
340 $\mathit{walking}(id_2)$
340 $p(id_2) = (25.88, -19.80)$
340 $\mathit{active}(id_1)$
340 $p(id_1) = (20.88, -11.90)$
340 $\mathit{walking}(id_3)$
340 $p(id_3) = (24.78, -18.77)$
380 $\mathit{walking}(id_3)$
380 $p(id_3) = (27.88, -9.90)$
380 $\mathit{walking}(id_2)$
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 $p(id_5) = (17.88, -11.90)$	
...	

Other Event Recognition Applications

Computer Networks:

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

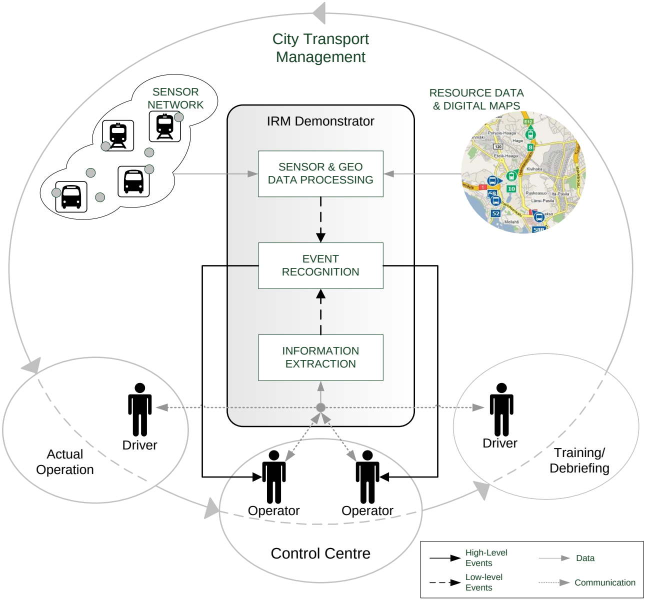
Financial transaction monitoring:

- 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



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	700	non-punctual
705	passenger density change to high	$since(705)$	reducing passenger comfort
715	scheduled stop leave		
820	scheduled stop enter		
815	passenger density change to low	$[705, 815]$	reducing passenger comfort
...			

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	$since(705)$	reducing passenger comfort
715	scheduled stop leave		
820	scheduled stop enter		
815	passenger density change to low	$[705, 815]$	reducing passenger comfort
...			

Tutorial Scope

Logic-based event recognition systems:

- Formal semantics
  - Verification, traceability.
- Declarative semantics
  - More easily applied to a variety of settings, easier to be understood by end users.
- High expressiveness
  - Compact representation.

At the same time, logic-based event recognition systems:

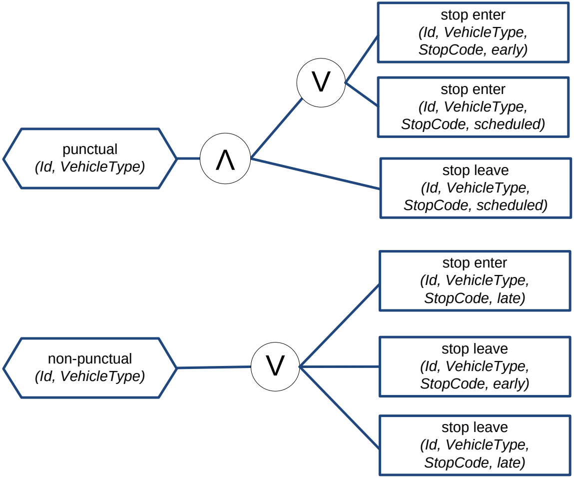
- can be very efficient;
- interoperate with non-logic based enterprise event processing infrastructures and middleware.

Tutorial Scope

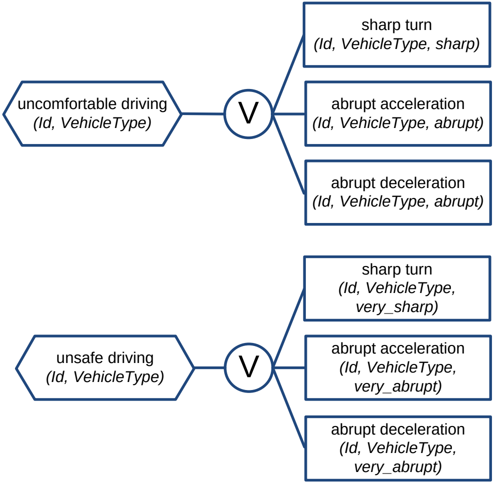
- We will present:
  - A purely temporal reasoning system.
  - A system for temporal and atemporal reasoning.
  - A system for temporal and atemporal reasoning, explicitly modelling uncertainty.
- For each system we will review:
  - the representation language,
  - reasoning algorithms,
  - machine learning techniques.
- Other systems have of course been used for event recognition.
- Other systems may be used for event recognition.

PART I:
Chronicle Recognition

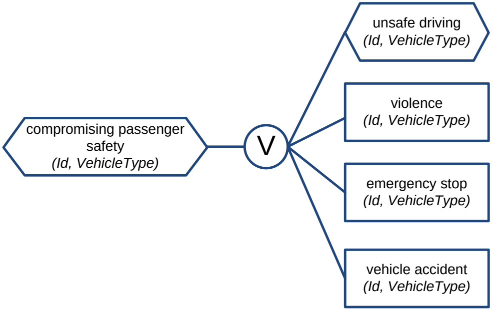
Event Definitions



Event Definitions



Event Definitions



High Level Event 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:

- IxTeT — LAAS.
- Chronicle Recognition System — CRS/Onera.
- Chronicle Recognition System — CRS/Orange-FT group.

CRS/Orange-FT group has 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`)

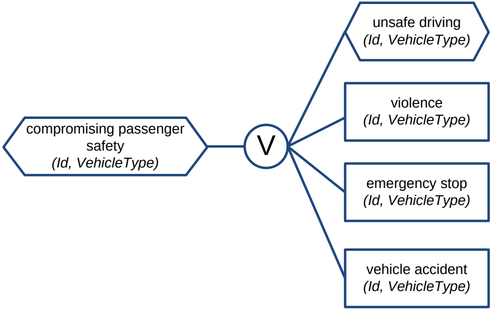
Chronicle Representation Language

```
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



- Passenger safety: difficult to express that violence, emergency stop, vehicle accident is more severe when taking place *far* from a hospital or a police station.
- No mathematical operators in the atemporal constraints of the CRS language.

Chronicle Representation Language

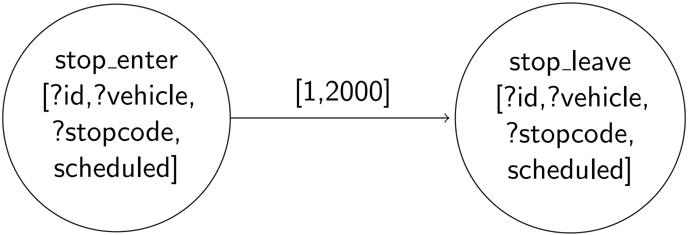
- Punctual line/route (as opposed to punctual vehicle): A route is said to be punctual if *all* vehicles of the route are punctual.
- We cannot express universal quantification in the CRS language.

CRS is a purely temporal reasoning system.

It is also a very efficient and scalable system.

Chronicle Recognition System

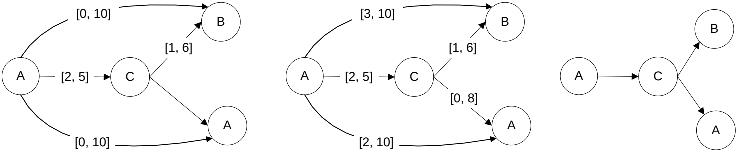
Each HLE definition is represented as a Temporal Constraint Network. Eg:



Chronicle Recognition System

Compilation stage:

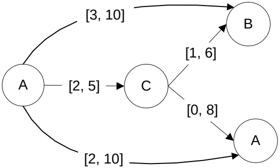
- Constraint propagation in the Temporal Constraint Network.
- Consistency checking.

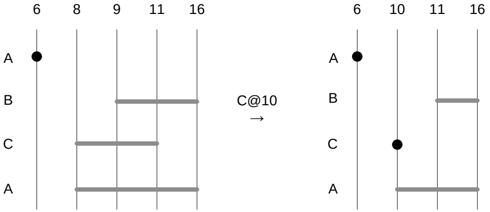


Chronicle Recognition System

Recognition stage:

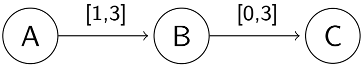
- Partial HLE instance evolution.
- Forward (predictive) recognition.





Chronicle Recognition System - Partial instances

HLE definition: Reduce tram endurance

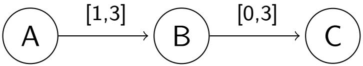


A: enter tram intersection
B: abrupt deceleration
C: abrupt acceleration

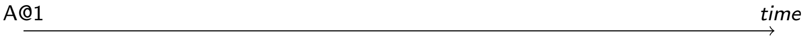
time

Chronicle Recognition System - Partial instances

HLE definition: Reduce tram endurance

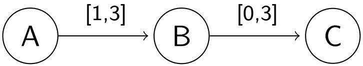


A: enter tram intersection
B: abrupt deceleration
C: abrupt acceleration

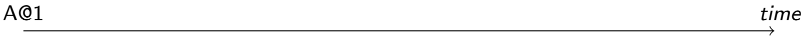


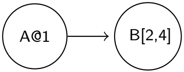
Chronicle Recognition System - Partial instances

HLE definition: Reduce tram endurance



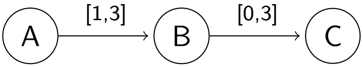
A: enter tram intersection
B: abrupt deceleration
C: abrupt acceleration



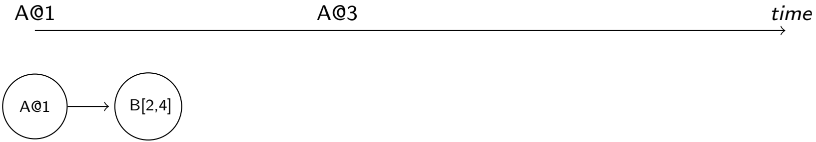


Chronicle Recognition System - Partial instances

HLE definition: Reduce tram endurance

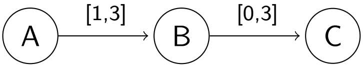


A: enter tram intersection
B: abrupt deceleration
C: abrupt acceleration

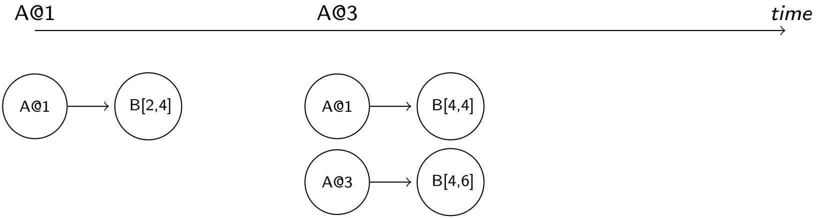


Chronicle Recognition System - Partial instances

HLE definition: Reduce tram endurance

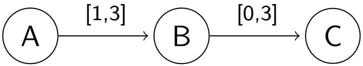


A: enter tram intersection
B: abrupt deceleration
C: abrupt acceleration

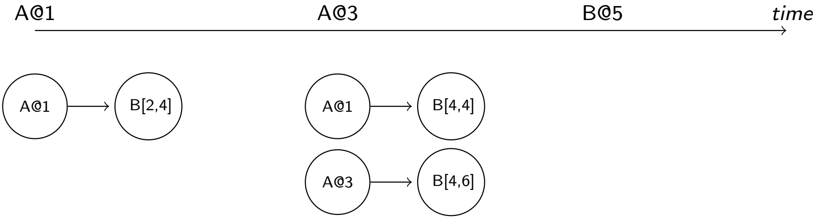


Chronicle Recognition System - Partial instances

HLE definition: Reduce tram endurance

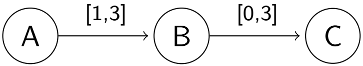


A: enter tram intersection
B: abrupt deceleration
C: abrupt acceleration

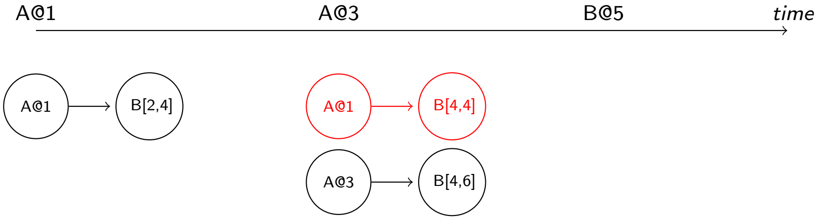


Chronicle Recognition System - Partial instances

HLE definition: Reduce tram endurance

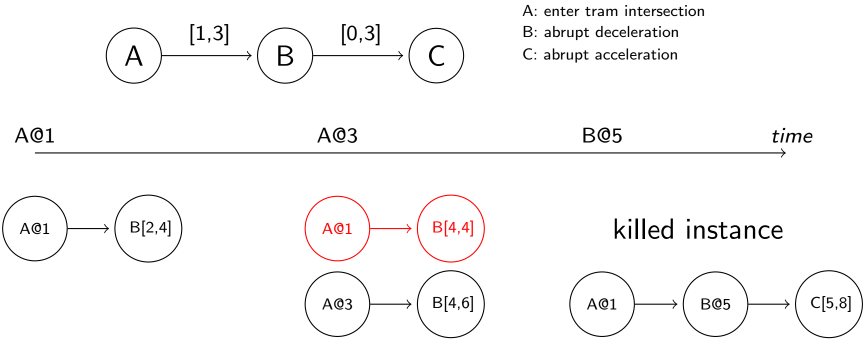


A: enter tram intersection
B: abrupt deceleration
C: abrupt acceleration



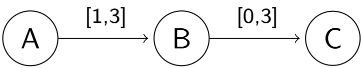
Chronicle Recognition System - Partial instances

HLE definition: Reduce tram endurance

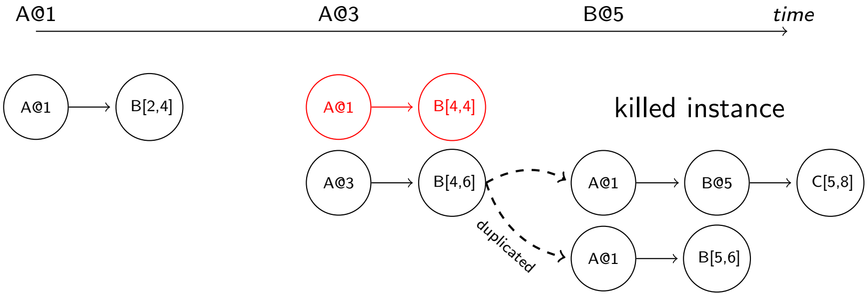


Chronicle Recognition System - Partial instances

HLE definition: Reduce tram endurance



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



Chronicle Recognition System

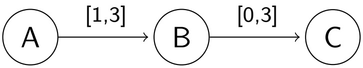
Recognition stage — partial HLE instance management:

- 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

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

- Temporal focusing leads to backward recognition.
- CRS thus now offers hybrid recognition: it mixes forward and backward recognition.
- Note: there exist purely backward recognition systems.

Chronicle Recognition System: Machine Learning

- Defining a HLE can be difficult and time-consuming.
- Methods that have been used for automatically extracting HLE definitions in the CRS language:
  - Automata based learning.
  - Frequency based analysis of sequence of events.
  - Inductive Logic Programming (ILP).
    - ILP is well-suited to CRS because first-order logic programs can be straightforwardly translated into CRS definitions and vice-versa.
    - ILP makes use of domain knowledge.

Inductive Logic Programming

ILP is a search problem.
Given:

- A set of positive examples E^+ and a set of negative examples E^- (ground facts).
- A hypotheses language L_H.
- A background knowledge base B. B and H are sets of clauses of the form $h \leftarrow b_1 \wedge \cdots \wedge b_n$, where the head h and b_i are literals.

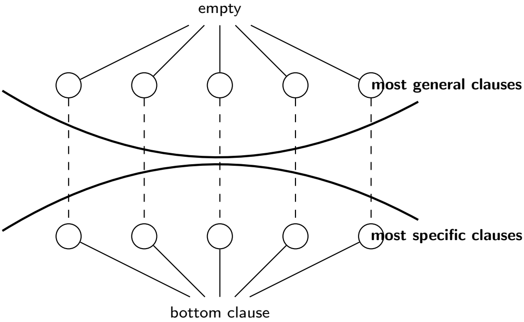
ILP searches for hypotheses $H \in L_H$ such that:

- $B \wedge H \models E^+$ (completeness).
- $B \wedge H \wedge E^- \not\models \Box$ (consistency).

Inductive Logic Programming

Walk the hypothesis space:

- states: hypotheses from L_H;
- stop: found a hypotheses set that satisfies completeness and consistency.



Inductive Logic Programming

A naïve generate-and-test algorithm would be far too computationally expensive

The search must be restricted:

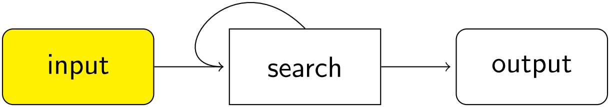
- Language bias to reduce the hypothesis space (how to express a hypothesis).
- Search bias to restrict the search (how a hypothesis will be selected).

Inductive Logic Programming: Learning Chronicles

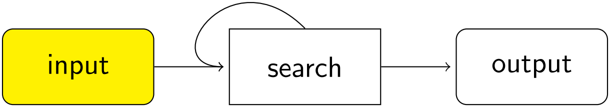
Try to learn the definition of 'punctual': $punctual(Id, V, T)$.

- Input:
  - A set of LLE.
  - Annotations of 'punctual' (ie, examples).
  - Language and search bias.
- Output: the definition of 'punctual'.

Inductive Logic Programming: Learning Chronicles



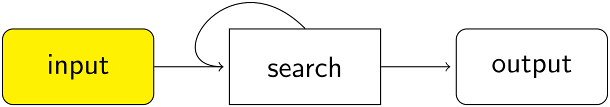
Inductive Logic Programming: Learning Chronicles



Background knowledge:

```
% LLE for tram tr1
event(stop_enter(tr1,tram,stop1,early),20,init,e1).
event(stop_leave(tr1,tram,stop1,scheduled),20.5,e1,e2).
...
% LLE for bus b2
event(stop_enter(b2,bus,stop4,scheduled),44,init,e1).
event(stop_leave(b2,bus,stop4,late),46,e1,e2).
...
```

Inductive Logic Programming: Learning Chronicles



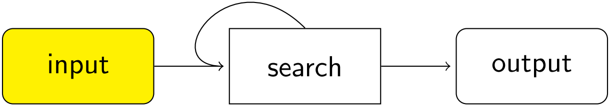
Background knowledge:

```
% LLE for tram tr1
event(stop_enter(tr1,tram,stop1,early),20,init,e1).
event(stop_leave(tr1,tram,stop1,scheduled),20.5,e1,e2).
...
% LLE for bus b2
event(stop_enter(b2,bus,stop4,scheduled),44,init,e1).
event(stop_leave(b2,bus,stop4,late),46,e1,e2).
...
```

Examples:

```
%E+
punctual(tr1,tram,20.5).   punctual(b2,bus,21).
...
%E-
punctual(tr1,tram,55).     punctual(b2,bus,46).
...
```

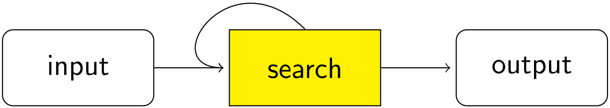
Inductive Logic Programming: Learning Chronicles



Language bias with mode declaration.

```
% mode declarations
:- modeh(*, punctual(+id,+vehicle,+float)).
:- modeb(*, event(stop_enter(+id,+vehicle,+stop,
#respected_time), -float, -evt, -evt)).
:- modeb(*, event(stop_leave(+id,+vehicle,+stop,
#respected_time), -float, -evt, -evt)).
:- modeb(*,event(abrupt_deceleration(+id,+vehicle),
-float, -evt, -evt)).
```

Inductive Logic Programming: Learning Chronicles



The ALEPH algorithm:

- Select an example from E^+, e.g. $\textit{punctual}(\textit{tr1}, \textit{tram}, \textit{20.5})$

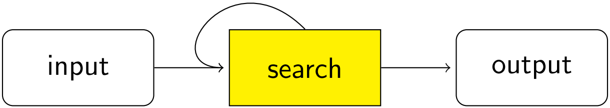
Inductive Logic Programming: Learning Chronicles

The ALEPH algorithm:

- Select an example from E^+, e.g. *punctual*(*tr1*, *tram*, *20.5*)
- Build most-specific-clause

```
[bottom clause]
punctual(Id, V, T) :-
event(abrupt_deceleration(Id, V), T1, E0, E1),
event(stop_enter(Id, V, S, early), T2, E1, E2),
event(stop_leave(Id, V, S, scheduled), T3, E2, E3).
```

Inductive Logic Programming: Learning Chronicles

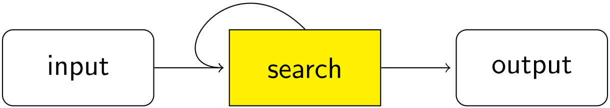


The ALEPH algorithm:

- Select an example from E^+, e.g. *punctual*(*tr1*, *tram*, *20.5*)
- Build most-specific-clause
- Search for more general clause

```
[best clause]
punctual(Id, V, T) :-
event(stop_enter(Id, V, S, early), T1, E0, E1),
event(stop_leave(Id, V, S, scheduled), T2, E1, E2).
```

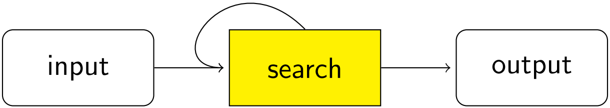
Inductive Logic Programming: Learning Chronicles



The ALEPH algorithm:

- Select an example from E^+, e.g. $punctual(tr1, tram, 20.5)$
- Build most-specific-clause
- Search for more general clause
- Add clause to H and remove covered examples

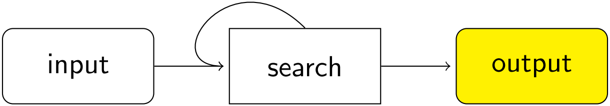
Inductive Logic Programming: Learning Chronicles



The ALEPH algorithm:

- Select an example from E^+, e.g. $\mathit{punctual}(\mathit{tr1}, \mathit{tram}, \mathit{20.5})$
- Build most-specific-clause
- Search for more general clause
- Add clause to H and remove covered examples
- Repeat until $E^+ = \varnothing$

Inductive Logic Programming: Learning Chronicles




```
[Rule 1]
punctual(Id,V,T2) :-
   event(stop_enter(Id,V,S,early),T1,E0,E1),
   event(stop_leave(Id,V,S,scheduled),T2,E1,E2).

[Rule 2]
punctual(Id,V,T2) :-
   event(stop_enter(Id,V,S,scheduled),T1,E0,E1),
   event(stop_leave(Id,V,S,scheduled),T2,E1,E2).
```

Inductive Logic Programming: Learning Chronicles

```
[Rule 2]
punctual(Id,V,T2) :-
   event(stop_enter(Id,V,S,scheduled),T1,E0,E1),
   event(stop_leave(Id,V,S,scheduled),T2,E1,E2).
```

straightforwardly translated into a chronicle model

```
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]
}
```

Inductive Logic Programming: Learning Chronicles

ILP:

+ The capacity to make use of background information and declarative bias.
- Not good at handling numbers — either expert-defined constraints or two-step relation-constraint learning.

Chronicle Recognition System: Summary

- Domain experts easily understand the CRS language.
- CRS has proven to be very efficient in different domains (eg, medical diagnosis, communication network, web-services).

But CRS does not:

- support atemporal reasoning;
- deal with uncertainty.

PART II:
Event Calculus

Event Calculus

- Formalism for representing events and their effects.
- Usually expressed as a logic (Prolog) program.

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
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
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

Event Calculus

$$
\begin{aligned}
&\text{happensAt}(\ \mathit{punctual}(\mathit{Id}, \mathit{Vehicle}),\ \ \mathit{DT}\ \ ) \leftarrow \\
&\quad \text{happensAt}(\ \mathit{stop\_enter}(\mathit{Id}, \mathit{Vehicle}, \mathit{StopCode}, \mathit{scheduled}),\ \ \mathit{AT}\ \ ), \\
&\quad \text{happensAt}(\ \mathit{stop\_leave}(\mathit{Id}, \mathit{Vehicle}, \mathit{StopCode}, \mathit{scheduled}),\ \ \mathit{DT}\ \ ), \\
&\quad \mathit{DT} > \mathit{AT}
\end{aligned}
$$

$$
\begin{aligned}
&\text{happensAt}(\ \mathit{non\_punctual}(\mathit{Id}, \mathit{Vehicle}),\ \ \mathit{AT}\ \ ) \leftarrow \\
&\quad \text{happensAt}(\ \mathit{stop\_enter}(\mathit{Id}, \mathit{Vehicle}, \_, \mathit{late}),\ \ \mathit{AT}\ \ )
\end{aligned}
$$

$$
\begin{aligned}
&\text{happensAt}(\ \mathit{non\_punctual}(\mathit{Id}, \mathit{Vehicle}),\ \ \mathit{DT}\ \ ) \leftarrow \\
&\quad \text{happensAt}(\ \mathit{stop\_leave}(\mathit{Id}, \mathit{Vehicle}, \_, \mathit{early}),\ \ \mathit{DT}\ \ )
\end{aligned}
$$

$$
\begin{aligned}
&\text{happensAt}(\ \mathit{non\_punctual}(\mathit{Id}, \mathit{Vehicle}),\ \ \mathit{DT}\ \ ) \leftarrow \\
&\quad \text{happensAt}(\ \mathit{stop\_leave}(\mathit{Id}, \mathit{Vehicle}, \_, \mathit{late}),\ \ \mathit{DT}\ \ )
\end{aligned}
$$

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

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

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

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

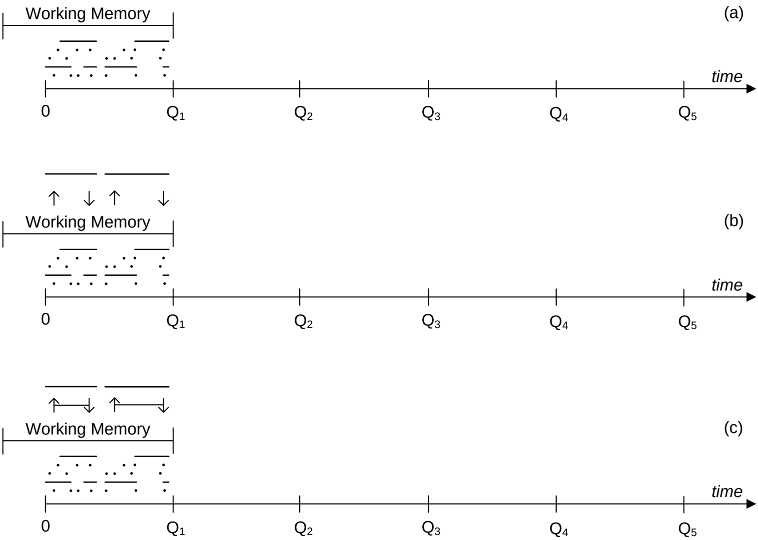
Event Calculus: Representation

- Very expressive — full power of logic programming.
- Temporal, logical and spatial representation in a single framework.

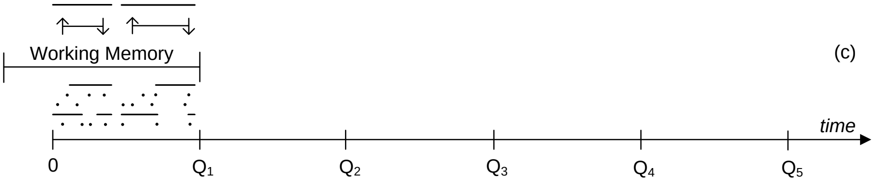
Event Calculus: Reasoning

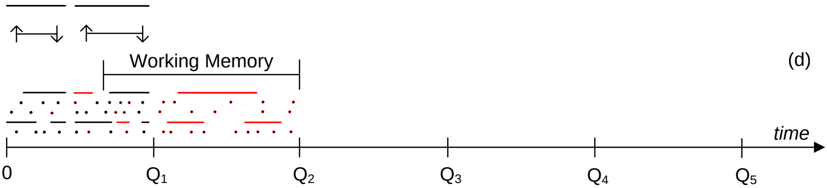
- There are various implementation routes concerning the Event Calculus, not restricted to logic programming.
- Reasoning in the Event Calculus can be performed at *query-time* or at *update-time*.

Event Calculus: Query-Time Reasoning

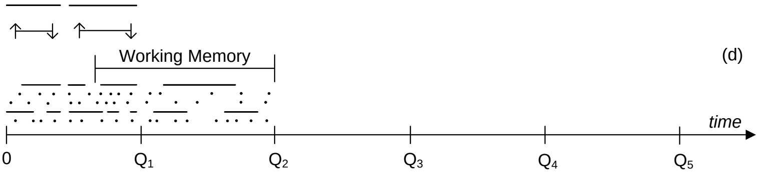


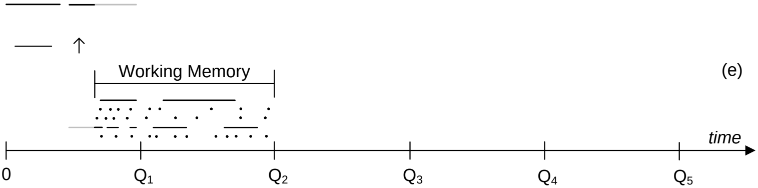
Event Calculus: Query-Time Reasoning

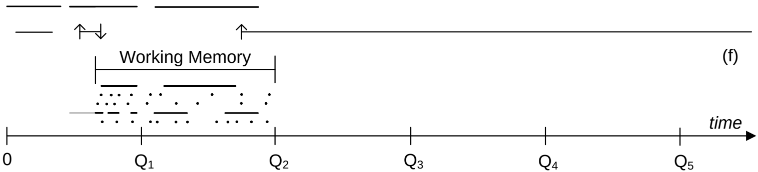




Event Calculus: Query-Time Reasoning







Event Calculus: Update-Time Reasoning

- *Update-time* reasoning: the recognition system infers and *stores* all consequences of each LLE when the LLE is entered into the recognition system. Query processing, therefore, amounts to retrieving the appropriate HLE intervals from the memory.
- Typical example: the 'Cached Event Calculus'.

Note:

- Caching does not necessarily imply update-time reasoning.

Event Calculus: Update-Time Reasoning

T	Input LLE	Output HLE
5	$stop\_enter(b_5, bus, 55, scheduled)$	
12	$stop\_leave(b_5, bus, 55, scheduled)$	$since(12)$: $punctuality(b_5, bus) = punctual$
18	$stop\_enter(b_5, bus, 56, early)$	
23	$stop\_leave(b_5, bus, 56, late)$	$[12, 23)$: $punctuality(b_5, bus) = punctual$ $since(23)$: $punctuality(b_5, bus) = non\_punctual$
30	$stop\_enter(b_5, bus, 57, scheduled)$	
	...	

Event Calculus: Reasoning

- It has been shown that the Event Calculus meets the user requirements in various application domains, including City Transport Management.
- Optimising the Event Calculus is an open research problem.

Event Calculus: Machine Learning

Event Calculus is a logic program, thus Inductive Logic Programming (ILP) can be applied.

Input:

- HLE annotation/examples.
- Background knowledge:
  - Event Calculus axioms.
  - Narrative of LLE.
  - Other domain specific knowledge (optionally).

Output: HLE definitions in terms of happensAt, initiatedAt and terminatedAt.

Learning Event Calculus HLE Definitions

Try to learn the definition of *punctual*:

happensAt($punctual(Id, Vehicle)$, T)

Positive examples:

...

happensAt($punctual(b_5, bus)$, 43)

...

Negative examples:

...

happensAt($punctual(b_5, bus)$, 87)

...

Learning Event Calculus HLE Definitions

Try to learn the definition of 'reducing passenger satisfaction':
initiatedAt($passenger\_satisfaction(Id, V) = reducing$, T)

Examples:

...

not holdsAt($passenger\_satisfaction(b_1, bus) = reducing$, 6)

holdsAt($passenger\_satisfaction(b_1, bus) = reducing$, 8)

...

Background Knowledge:

...

happensAt($passenger\_density\_change(b_1, bus, low)$, 6)

happensAt($passenger\_density\_change(b_1, bus, high)$, 8)

...

holdsAt($temperature(b_1, bus) = very\_warm$, 8)

holdsAt($noise\_level(b_1, bus) = high$, 8)

...

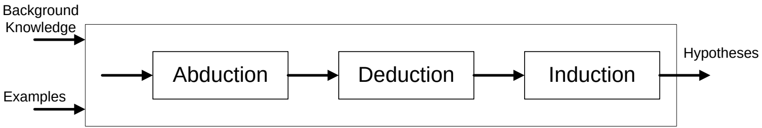
Learning Event Calculus HLE Definitions

Combination of Abductive Logic Programming (ALP) with ILP:

- Enables non-observation predicate learning.
- Exploits background knowledge.
- Creates ‘explanations’ that fill incomplete knowledge
  - In our case, produce ground initiatedAt predicates.

$$
\begin{aligned}
&\text{holdsAt}(\ F = V,\ \ T\ ) \leftarrow \\
&\qquad \text{initiatedAt}(\ F = V,\ \ T'\ ), \\
&\qquad T^{'} < T, \\
&\qquad \text{not broken}(\ F = V,\ \ T^{'},\ \ T\ )
\end{aligned}
$$

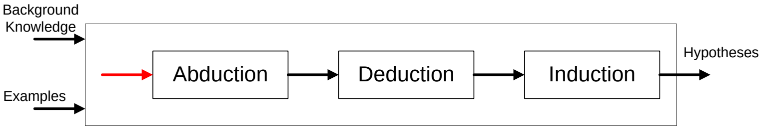
Learning Event Calculus HLE Definitions



The XHAIL system:

- Abduction: produce ground initiatedAt predicates.
- Deduction: produce preliminary ground hypotheses.
- Induction: perform generalisation.

Learning Event Calculus HLE Definitions



Examples:

...

not holdsAt($passenger\_satisfaction(b_1, bus) = reducing$, 6)

holdsAt($passenger\_satisfaction(b_1, bus) = reducing$, 8)

...

Narrative:

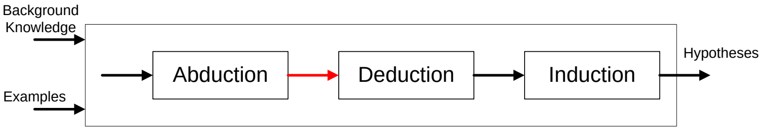
...

happensAt($passenger\_density\_change(b_1, bus, low)$, 6)

happensAt($passenger\_density\_change(b_1, bus, high)$, 8)

...

Learning Event Calculus HLE Definitions



Delta set:

…

$\textcolor{red}{\mathrm{initiatedAt}}(\ passenger\_satisfaction(b_1, bus) = reducing,\ \ 8\ )$

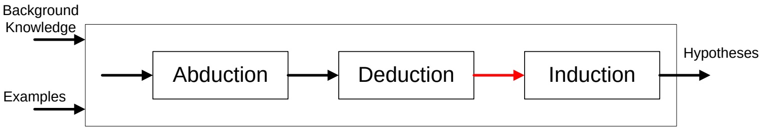
$\textcolor{red}{\mathrm{initiatedAt}}(\ passenger\_satisfaction(b_1, bus) = reducing,\ \ 9\ )$

…

$\textcolor{red}{\mathrm{initiatedAt}}(\ passenger\_satisfaction(b_5, bus) = reducing,\ \ 200\ )$

…

Learning Event Calculus HLE Definitions



Kernel set:

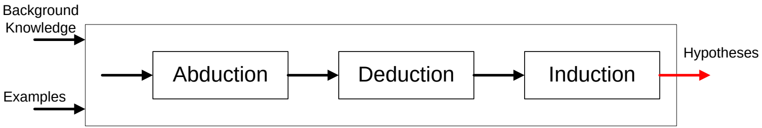
$\text{initiatedAt}(\ \textit{passenger\_satisfaction}(b_1, bus) = \textit{reducing},\ 8\ ) \leftarrow$
$\quad \text{happensAt}(\ \textit{passenger\_density\_change}(b_1, bus, high),\ 8\ ),$
$\quad \text{holdsAt}(\ \textit{temperature}(b_1, bus) = \textit{very\_warm},\ 8\ ),$
$\quad \text{holdsAt}(\ \textit{noise\_level}(b_1, bus) = \textit{high},\ 8\ )$

...

$\text{initiatedAt}(\ \textit{passenger\_satisfaction}(b_5, bus) = \textit{reducing},\ 200\ ) \leftarrow$
$\quad \text{happensAt}(\ \textit{passenger\_density\_change}(b_5, bus, high),\ 200\ ),$
$\quad \text{holdsAt}(\ \textit{temperature}(b_5, bus) = \textit{very\_warm},\ 200\ ),$
$\quad \text{holdsAt}(\ \textit{noise\_level}(b_5, bus) = \textit{low},\ 200\ )$

...

Learning Event Calculus HLE Definitions



Hypotheses:

$$
\begin{aligned}
&\text{initiatedAt}(\ \mathit{passenger\_satisfaction}(\mathit{Id}, V) = \mathit{reducing},\ \ T\ ) \leftarrow \\
&\qquad \text{happensAt}(\ \mathit{passenger\_density\_change}(\mathit{Id}, V, \mathit{high}),\ \ T\ ), \\
&\qquad \text{holdsAt}(\ \mathit{temperature}(\mathit{Id}, V) = \mathit{very\_warm},\ \ T\ )
\end{aligned}
$$

...

Event Calculus: Machine Learning

Depending on the available examples (HLE annotation) and desired HLE formalisation, we may use:

- ILP only
  - see Part I for the advantages and disadvantages of ILP.
- ALP & ILP
  - state-of-the-art systems do not scale well in large datasets.

Event Calculus: Summary

- Domain experts easily understand the Event Calculus.
- The Event Calculus is a very expressive formalism, supporting both temporal and atemporal representation and reasoning.
- The Event Calculus has proven efficient enough for various application domains.

But:

- The Event Calculus does not deal with uncertainty.

PART III:
Markov Logic

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 & Graphical 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

One such approach is Markov Logic Networks

First order logic

- Constants, variables, functions and predicates
  e.g. tram, T, $punctual(tr_0, tram)$, happensAt(E, T)
- Grounding: replace all variables with constants
  e.g. happensAt($punctual(tr_0, tram)$, 10)
- **World**: Assignment of truth values to all ground predicates
  e.g.
  ...
  happensAt($punctual(tr_0, tram)$, 10) $= True$
  happensAt($punctual(tr_0, tram)$, 15) $= False$
  ...
- A KB in first-order logic: a set of hard constraints on a set of possible worlds

Markov Logic

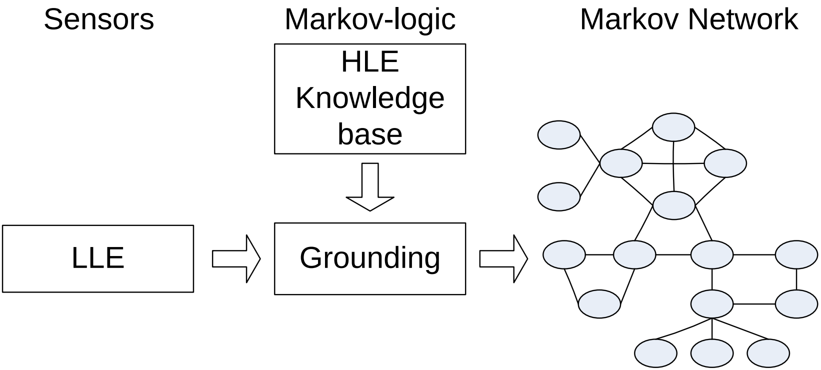
Markov Logic or Markov Logic Networks (MLN):

- Unifies first order logic with graphical models
  - Compactly represent complex event relations
  - Handle uncertainty
- Syntactically: weighted first-order logic formulas (F_i, w_i)
- Semantically: (F_i, w_i) represents a probability distribution over possible worlds

$$P(world) \propto exp\left(\sum(weights\ of\ \ formulas\ it\ satisfies)\right)$$

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

Markov Logic: Knowledge-base model construction



Markov Logic: Representation

Example definition of HLE 'uncomfortable_driving' :

$$
\begin{array}{ll}
w_1 & \begin{array}{l} abrupt\_movement(Id, V, T) \leftarrow \\ \quad abrupt\_acceleration(Id, V, T) \lor \\ \quad abrupt\_deceleration(Id, V, T) \lor \\ \quad sharp\_turn(Id, V, T) \end{array}
\end{array}
$$

$$
\begin{array}{ll}
w_2 & \begin{array}{l} uncomfortable\_driving(Id, V, T_2) \leftarrow \\ \quad approach\_intersection(Id, V, T_1) \land \\ \quad abrupt\_movement(Id, V, T_2) \land \\ \quad before(T_1, T_2) \end{array}
\end{array}
$$

Markov Logic: Representation

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

Markov Logic: LLE uncertainty propagation

Sensors detect LLE

- Certainty
- Degree of confidence

Example:

- $sharp\_turn(tr_0, tram, 20)$ detected with probability 0.7
- it can be represented with a weight value — log-odds of the detection probability

Markov Logic: Network Construction

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

$\frac{1}{3}w_1 \quad \neg abrupt\_acceleration(Id, V, T) \lor abrupt\_movement(Id, V, T)$

$\frac{1}{3}w_1 \quad \neg abrupt\_deceleration(Id, V, T) \lor abrupt\_movement(Id, V, T)$

$\frac{1}{3}w_1 \quad \neg sharp\_turn(Id, V, T) \lor abrupt\_movement(Id, V, T)$

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

Markov Logic: Network Construction

Template that produces ground Markov network:

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

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

$$Z = \sum_{x \in \mathcal{X}} exp(P(X = x))$$

Markov Logic: Network Construction

$\frac{1}{3}w_1 \quad \neg abrupt\_acceleration(Id, V, T) \lor abrupt\_movement(Id, V, T)$

$\frac{1}{3}w_1 \quad \neg abrupt\_deceleration(Id, V, T) \lor abrupt\_movement(Id, V, T)$

$\frac{1}{3}w_1 \quad \neg sharp\_turn(Id, V, T) \lor abrupt\_movement(Id, V, T)$

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

LLE:
$abrupt\_acceleration(tr_0, tram, 101)$
$approach\_intersection(tr_0, tram, 100)$
$before(100, 101)$

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

Markov Logic: Network Construction

For example, the clause:

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

produces the following groundings:

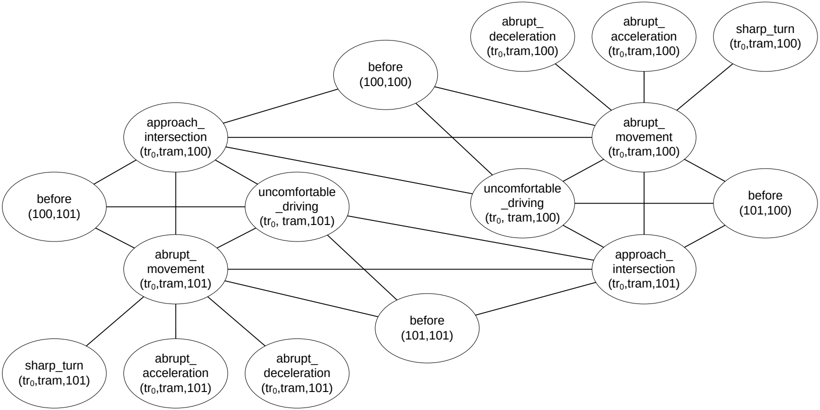
$w_2 \quad \neg approach\_intersection(tr_0, tram, 100) \ \vee \neg abrupt\_movement(tr_0, tram, 100) \vee$
$\neg before(100, 100) \vee uncomfortable\_driving(tr_0, tram, 100)$

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

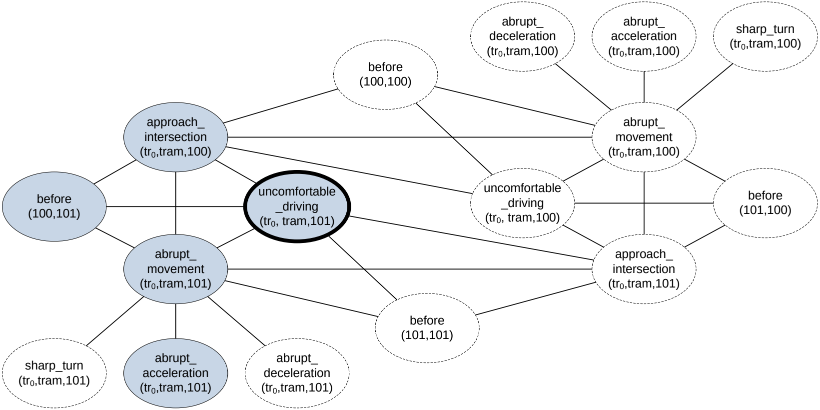
$w_2 \quad \neg approach\_intersection(tr_0, tram, 101) \ \vee \neg abrupt\_movement(tr_0, tram, 100) \vee$
$\neg before(101, 100) \vee uncomfortable\_driving(tr_0, tram, 100)$

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

Markov Logic: Network Construction

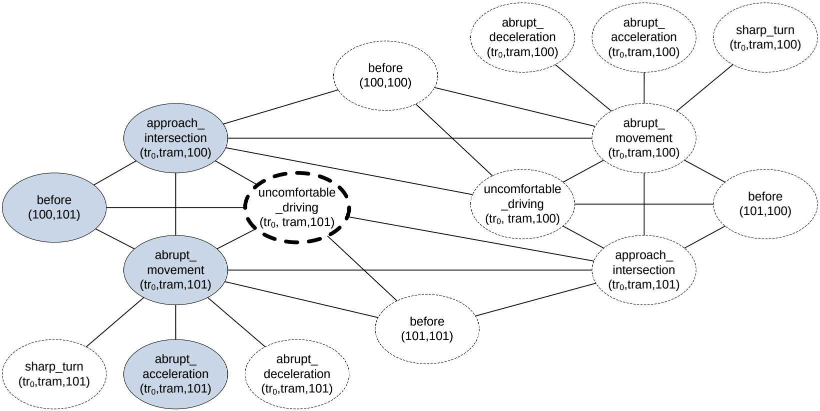


Markov Logic: World state discrimination



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

Markov Logic: World state discrimination



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

Markov Logic: Inference

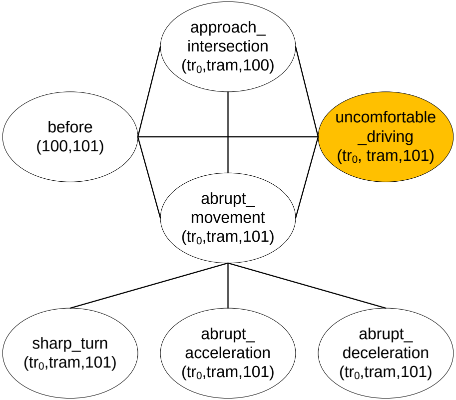
- Event recognition involves querying about HLE
- Having a ground Markov network
- Apply standard probabilistic inference methods
- Large network with complex structure
- Infeasible inference
- MLN combine logical and probabilistic inference methods

Markov Logic: Conditional inference

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

- Query variables Q: HLE

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

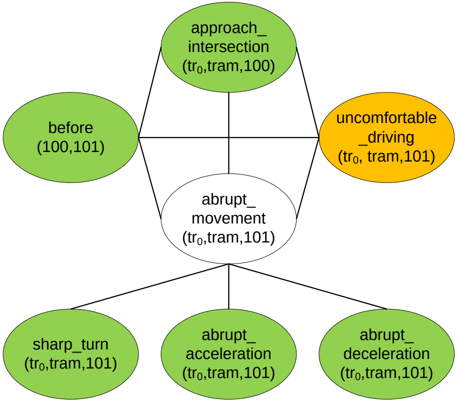


Markov Logic: Conditional inference

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

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

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

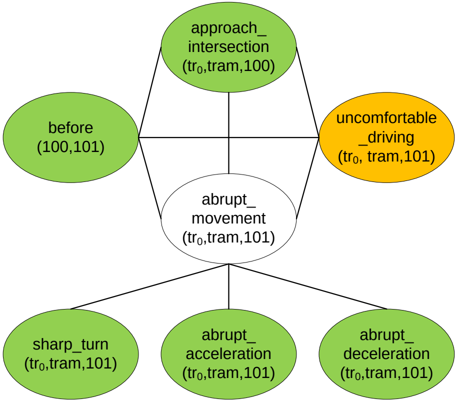


Markov Logic: Conditional inference

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

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

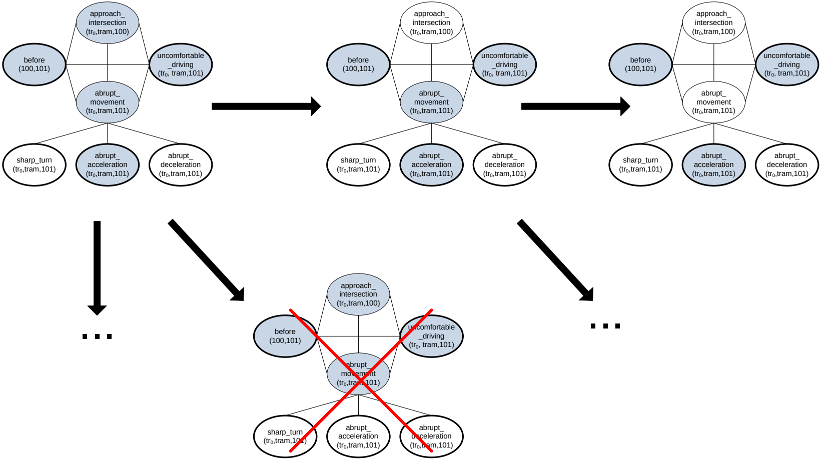
$$P(Q \mid E = e) = \frac{P(Q, E = e, H)}{P(E = e, H)}$$



Markov Logic: Conditional inference

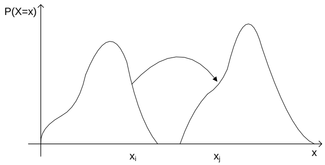
- Efficiently approximated with sampling
- Markov Chain Monte Carlo: e.g Gibbs sampling
- Random walks in state space
- Reject all states where $E = e$ does not hold

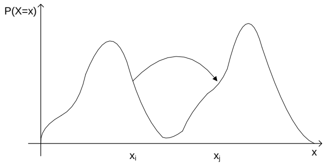
Markov Logic: MCMC



Markov Logic: Deterministic dependencies

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





Markov Logic: Machine Learning

- Traning data: LLE annotated with HLE

  ...
  $abrupt\_acceleration(tr_0, tram, 101)$
  $approach\_intersection(tr_0, tram, 100)$
  $uncomfortable\_driving(tr_0, tram, 101)$
  ...
  $\neg abrupt\_acceleration(tr_8, tram, 150)$
  $approach\_intersection(tr_8, tram, 149)$
  $\neg uncomfortable\_driving(tr_0, tram, 150)$
  ...

- Weight estimation
  - Structure is known
  - Find weight values that maximize the likelihood function
  - Likelihood function: how well our model fits to the data
  - Generative learning
  - Discriminative learning
- Structure learning: first-order logic formulas

Markov Logic: Generative Weight Learning

Log-likelihood:

$$\log P_w(X = x) = \sum_i w_i n_i(x) - \log Z$$

- Use iterative methods: e.g. gradient ascent
- Optimize over the weight space
- Good news: Converge to the global optimum
- Bad news: Each iteration requires inference on the network

Markov Logic: Generative Weight Learning

Pseudo-log-likelihood function:

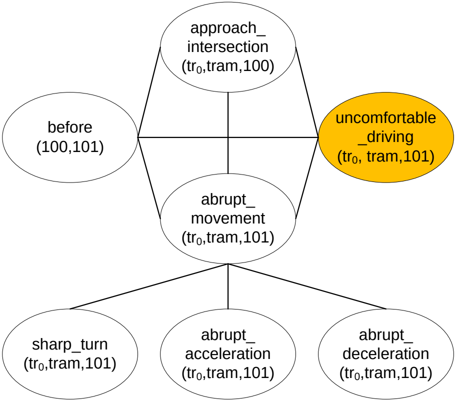
$$log\ P_w^*(X=x) = \sum_{l=1}^{n} log\ P_w(X_l = x_l \mid MB_x(X_l))$$

- Each ground predicate is conditioned on its Markov blanked
- Very efficient
- Does not require inference
- Markov-blanket must be fully observed

Markov Logic: Generative Weight Learning

Training data:

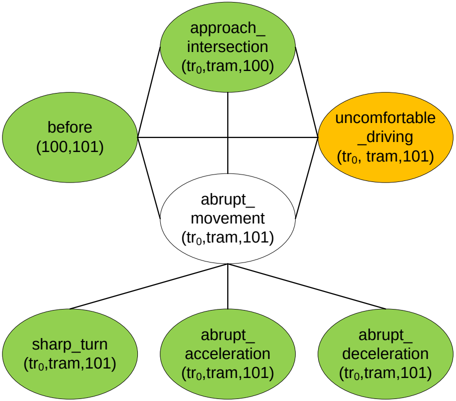
- 'uncomfortable_driving' is annotated



Markov Logic: Generative Weight Learning

Training data:

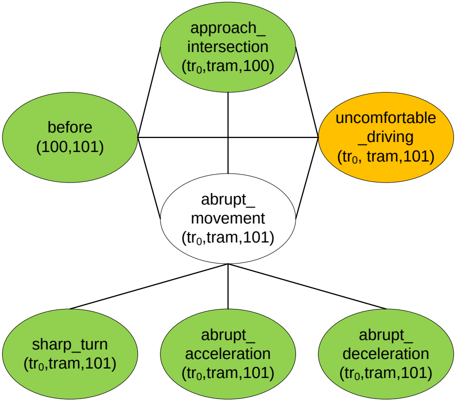
- 'uncomfortable_driving' is annotated
- The truth value of LLE is known



Markov Logic: Generative Weight Learning

Training data:

- 'uncomfortable_driving' is annotated
- The truth value of LLE is known
- But the truth value of 'abrupt_movement' is unknown



Markov Logic: Discriminative Weight Learning

In event recognition we know *a priori*:

- Evidence variables: LLE
- Query variables: HLE
- Recognize HLE given LLE

Conditional log-likelihood function:

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

- Conditioning on evidence reduces the likely states
- Inference takes place on a simpler model
- Can exploit information from long-range dependencies

Markov Logic: Summary

- Unify first order logic with graphical models
  - First order logic: represent complex events
  - Graphical models: deal with uncertainty
- Can exploit background knowledge
- Combine logical and probabilistic inference methods
- Provide weight and structure learning algorithms

But:

- Hard to incorporate numerical constraints in inference
- Need for simultaneous learning of weights, numerical (temporal) constraints and logical structure

OPEN ISSUES

Open Issues

- Extension of the Chronicle Recognition System with atemporal reasoning.
- Improvement of the reasoning efficiency of the Event Calculus.
- Learning HLE definitions given large, partially supervised datasets.
- Use of numerical constraints in the inference algorithms of Markov Logic Networks.
- Simultaneous learning of weights, numerical (temporal) constraints and logical structure of HLE definitions.