Complex Event Recognition

Alexander Artikis

Elias Alevizos, Nikos Katzouris,
Vagelis Michelioudakis, George Paliouras

Institute of Informatics & Telecommunications,
NCSR Demokritos, Athens, Greece

http://cer.iit.demokritos.gr

Complex Event Recognition (Event Pattern Matching)

Input:

- Symbolic representation of time-stamped, simple, derived events (SDE) coming from (geographically distributed) sources.
- Big Data.

Complex Event Recognition (Event Pattern Matching)

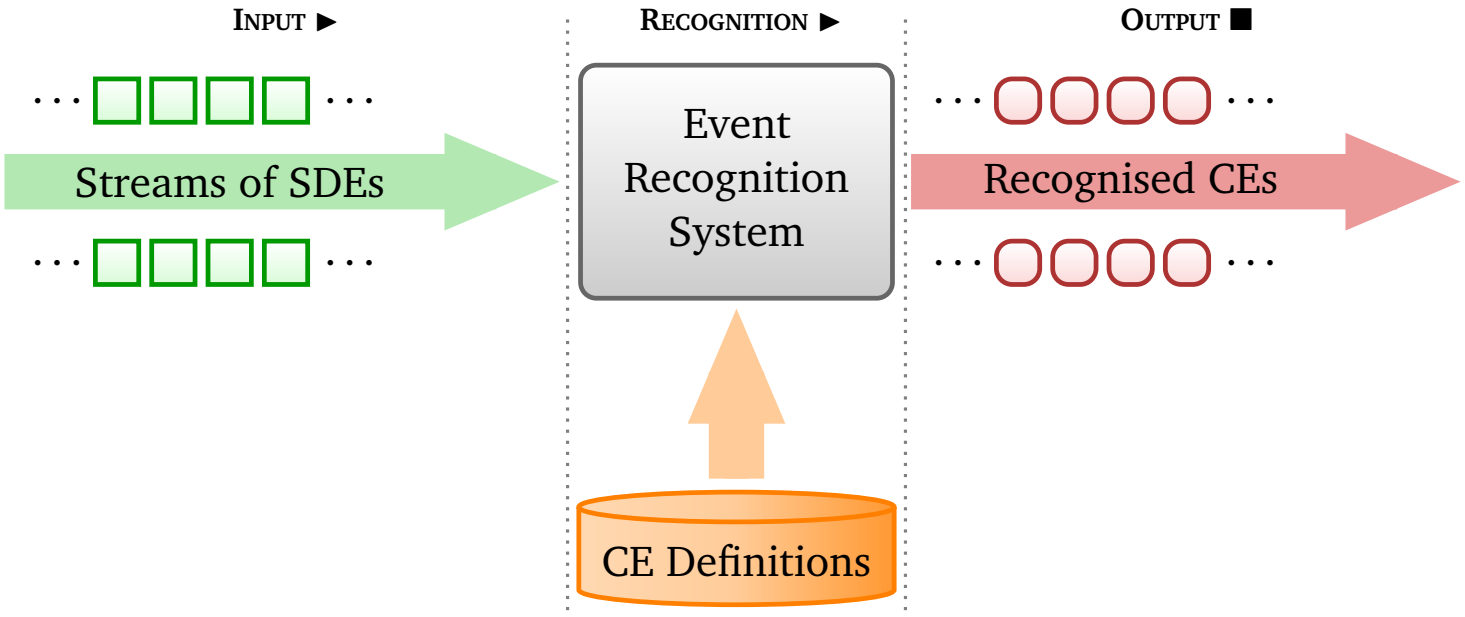
Input:

- Symbolic representation of time-stamped, simple, derived events (SDE) coming from (geographically distributed) sources.
- Big Data.

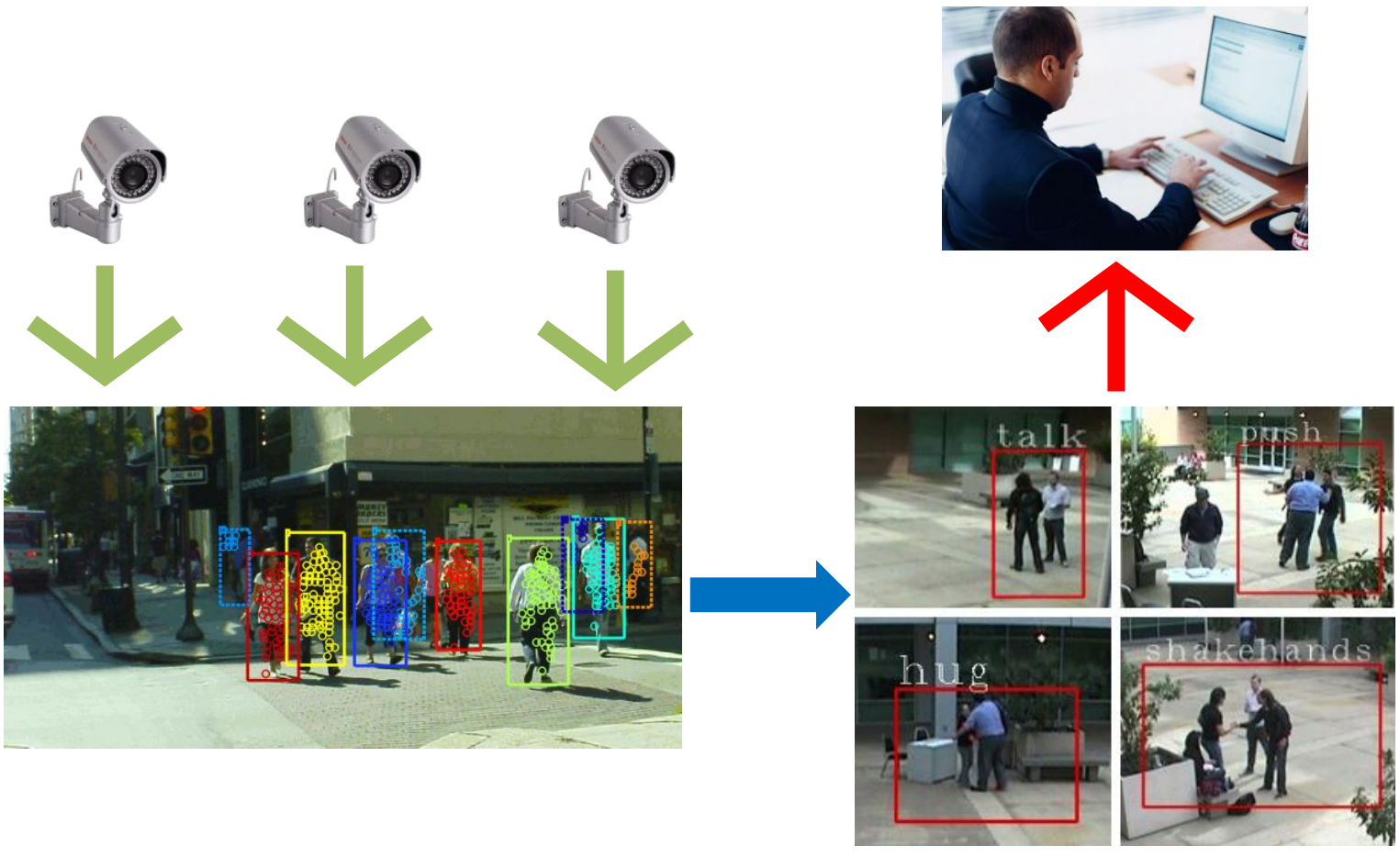
Output:

- Complex or composite events (CE) — collections of SDE and/or CE that satisfy some pattern (temporal, spatial, logical constraints).
  - Not restricted to aggregates.
- Humans understand CE easier than SDE.

Complex Event Recognition



Complex Event Recognition for Security



Complex Event Recognition for Security

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 $p(id_2) = (28.27, -9.66)$	

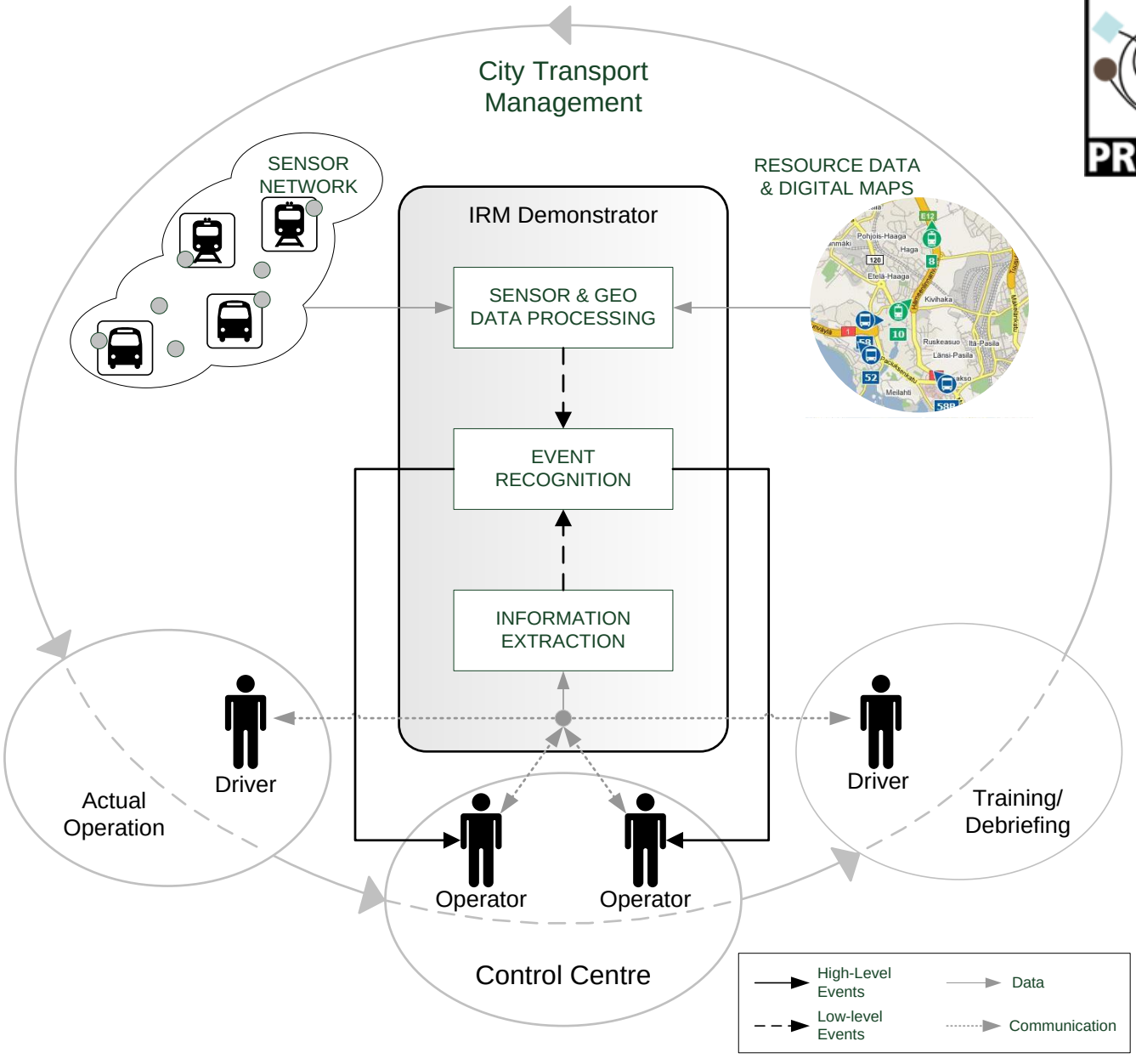
Complex Event Recognition for Security

Input		Output	
340	$inactive(id_0)$	340	$left\_object(id_1, id_0)$
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	$p(id_2) = (28.27, -9.66)$		

Complex Event Recognition for Security

Input	Output
340 $inactive(id_0)$	340 $left\_object(id_1, id_0)$
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 $p(id_2) = (28.27, -9.66)$	

City Transport & Traffic Management



City 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	

City 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

City 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		
915	passenger density		
	change to low		

City 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	[620,720]	congestion
820	scheduled stop leave	[215,820]	non-punctual
915	passenger density		
	change to low		

Credit Card Fraud Recognition

SPEEDD

SDE:

- Credit card transactions from all over the world.

CE:

- Cloned card — a credit card is being used simultaneously in different countries.
- 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.

Credit Card Fraud Recognition

SPEEDD

- Fraud must be detected within 25 milliseconds.

Credit Card Fraud Recognition

SPEEDD

- Fraud must be detected within 25 milliseconds.
- Fraudulent transactions: 0.1% of the total number of transactions.

Credit Card Fraud Recognition

SPEEDD

- Fraud must be detected within 25 milliseconds.
- Fraudulent transactions: 0.1% of the total number of transactions.
- Fraud is constantly evolving.

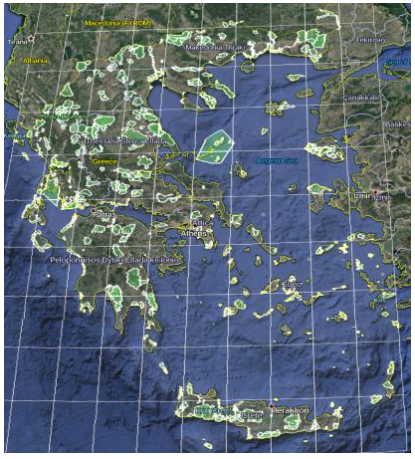
Credit Card Fraud Recognition

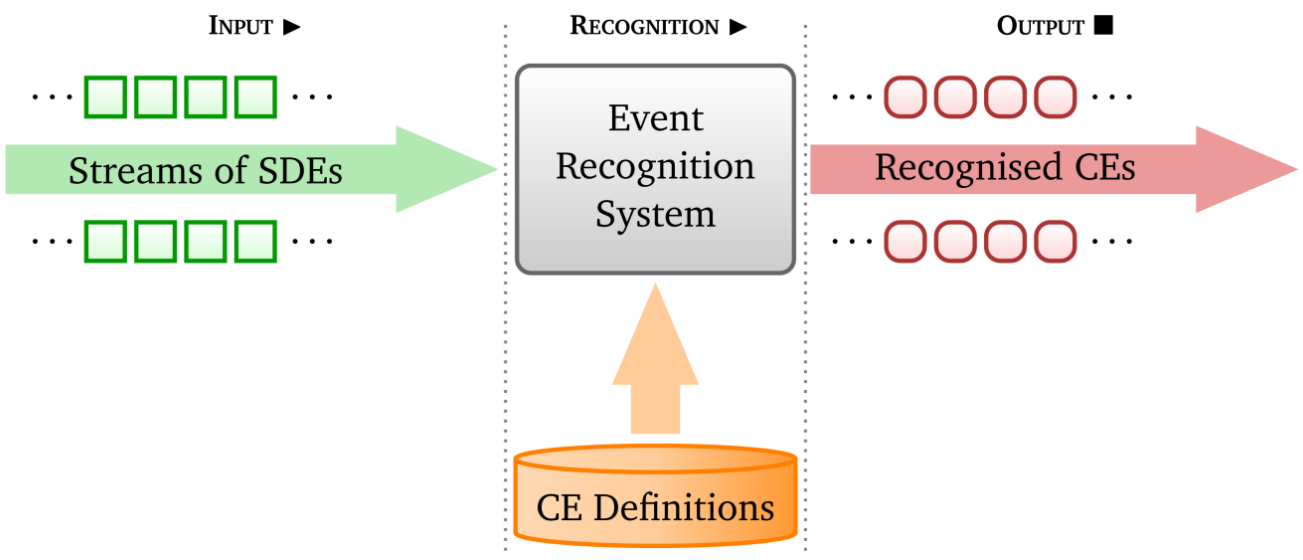
SPEEDD

- Fraud must be detected within 25 milliseconds.
- Fraudulent transactions: 0.1% of the total number of transactions.
- Fraud is constantly evolving.
- Erroneous transactions, missing fields.

Event Recognition for Maritime Surveillance

Event Recognition for Maritime Surveillance





Fast Approach

- A vessel is moving at a high speed ...
- towards other vessels.

Suspicious Delay

- A vessel fails to report position ...
- and the estimated speed during the communication gap is low.

Possible Rendezvous

- Two vessels are suspiciously delayed ...
- in the same location ...
- at the same time.

Package Picking

- A vessel stops in a location ...
- and another vessel stops in the same location ...
- in a short period of time.

Event Recognition for Maritime Surveillance

'Sea' of information:

- 400,000 vessels globally: 40,000 signals/sec to be processed online.

Event Recognition for Maritime Surveillance

'Sea' of information:

- 400,000 vessels globally: 40,000 signals/sec to be processed online.
- Position signals need to be combined with other data streams
  - Weather forecasts, sea currents, etc.

Event Recognition for Maritime Surveillance

'Sea' of information:

- 400,000 vessels globally: 40,000 signals/sec to be processed online.
- Position signals need to be combined with other data streams
  - Weather forecasts, sea currents, etc.
- ... and static information
  - NATURA areas, shallow waters, coastlines, etc.

Event Recognition for Maritime Surveillance

'Sea' of noisy information:

- GPS manipulation has risen 59% over the past two years.

Event Recognition for Maritime Surveillance

'Sea' of noisy information:

- GPS manipulation has risen 59% over the past two years.
- There is a 30% increase over the past two years of vessels reporting a false identity.

Event Recognition for Maritime Surveillance

'Sea' of noisy information:

- GPS manipulation has risen 59% over the past two years.
- There is a 30% increase over the past two years of vessels reporting a false identity.
- 27% of vessels do not report position at least 10% of the time.
  - 19% of vessels are repeat offenders.

Event Recognition

Requirements:

- Efficient reasoning
  - to support real-time decision-making in large-scale, (geographically) distributed applications.

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.

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 CE definition development.

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 CE definition development.

Composite Event Algebra

Core components of an event algebra with point-based semantics:

- Sequencing (SEQ) lists the required event types in temporal order — eg SEQ(A, B, C).

Composite Event Algebra

Core components of an event algebra with point-based semantics:

- Sequencing (SEQ) lists the required event types in temporal order — eg SEQ(A, B, C).
- Kleene closure ($+$) collects a finite yet unbound number of events of a particular type. It is used as a component of SEQ — eg SEQ($A, B+, C$).

Composite Event Algebra

Core components of an event algebra with point-based semantics:

- Sequencing (SEQ) lists the required event types in temporal order — eg SEQ(A, B, C).
- Kleene closure ($+$) collects a finite yet unbound number of events of a particular type. It is used as a component of SEQ — eg SEQ($A, B+, C$).
- Negation ($\tilde{}$ or !) verifies the absence of certain events in a sequence — eg SEQ($A, !B, C$).

Composite Event Algebra

Core components of an event algebra with point-based semantics:

- Sequencing (SEQ) lists the required event types in temporal order — eg SEQ(A, B, C).
- Kleene closure ($+$) collects a finite yet unbound number of events of a particular type. It is used as a component of SEQ — eg SEQ($A, B+, C$).
- Negation ($\tilde{}$ or !) verifies the absence of certain events in a sequence — eg SEQ($A, !B, C$).
- Value predicates specify constraints on the event attributes
  - Aggregate functions *max*, *min*, *count*, *sum*, *avg*.

Composite Event Algebra

- Composition refers to:
  - Union of constraints — eg $\mathrm{SEQ}(A, B, C) \cup \mathrm{SEQ}(A, D, E)$.
  - Negation of a sequence — eg $!\mathrm{SEQ}(A, B, C)$.
  - Kleene closure of a constraint — eg $\mathrm{SEQ}(A, B, C)+$.

Composite Event Algebra

- Composition refers to:
  - Union of constraints — eg $\mathrm{SEQ}(A, B, C) \cup \mathrm{SEQ}(A, D, E)$.
  - Negation of a sequence — eg $!\mathrm{SEQ}(A, B, C)$.
  - Kleene closure of a constraint — eg $\mathrm{SEQ}(A, B, C)+$.
- Windowing (WITHIN) restricts a CE definition to a specific time period.

Event Selection Strategies

- Strict contiguity: No intervening events allowed between two sequence events in the pattern.

Event Selection Strategies

- Strict contiguity: No intervening events allowed between two sequence events in the pattern.
- Partition contiguity: Same as above, but the stream is partitioned into substreams according to a partition attribute. Events must be contiguous within the same partition.

Event Selection Strategies

- Strict contiguity: No intervening events allowed between two sequence events in the pattern.
- Partition contiguity: Same as above, but the stream is partitioned into substreams according to a partition attribute. Events must be contiguous within the same partition.
- Skip-till-next-match: Intervening events are allowed, but only non-overlapping occurrences of SEQ are detected. E.g. for SEQ(A, B, C) and a_1, b_1, b_2, c_1, only a_1, b_1, c_1 will be detected.

Event Selection Strategies

- Strict contiguity: No intervening events allowed between two sequence events in the pattern.
- Partition contiguity: Same as above, but the stream is partitioned into substreams according to a partition attribute. Events must be contiguous within the same partition.
- Skip-till-next-match: Intervening events are allowed, but only non-overlapping occurrences of SEQ are detected. E.g. for SEQ(A, B, C) and a_1, b_1, b_2, c_1, only a_1, b_1, c_1 will be detected.
- Skip-till-any-match: Most flexible (and expensive). Detects every possible occurrence. For the previous example, a_1, b_2, c_1 will also be detected.

Example

Quickly moving away from an area of suspicious activity:

- After a communication gap, ...
- a vessel changes speed to over 30 knots.
- Partition contiguity ensures that a, b, c are contiguous and refer to the same vessel (*vesselId*).

PATTERN SEQ(*gapStart a*, *gapEnd b*, *speedChange c*)
WHERE partition-contiguity
AND *vesselId*
AND $c.velocity > 30$
WITHIN 3600

Example

Fishing pattern:

- A vessel slows down, ...
- begins a series of turns, where, for each pair of successive turns, their difference in heading is more than 90 degrees, ...
- and subsequently the vessel stops moving at a low speed.

PATTERN SEQ($lowSpeedStart\ a,\ turn+\ b,\ lowSpeedEnd\ c$)
WHERE skip-till-next-match
AND $vesselId$
AND $b[i].heading - b[i-1].heading > 90$
WITHIN 21600

CE as Chronicle

A CE can be defined as a set of events interlinked by time constraints and whose occurrence may depend on the context.

CE as Chronicle

A CE 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.

CE as Chronicle

A CE 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 Algebra

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 abnormal_vessel_movement[?id](T2) {
 event( speedChange[?id], T0 )
 event( speedChange[?id], T1 )
 event( speedChange[?id], T2 )
 T1 > T0
 T2 > T1
 T2 - T0 in [1, 20000]
 noevent( turn[?id], ( T0+1, T2 ) )
}
```

Chronicle Representation Language

- Mathematical operators in the atemporal constraints of the language are not allowed.
  - No spatial reasoning.
  - No use of background knowledge.

Chronicle Representation Language

- Mathematical operators in the atemporal constraints of the language are not allowed.
  - No spatial reasoning.
  - No use of background knowledge.
- Universal quantification is not allowed.
  - cannot express a property about all vessels in some area.

Chronicle Representation Language

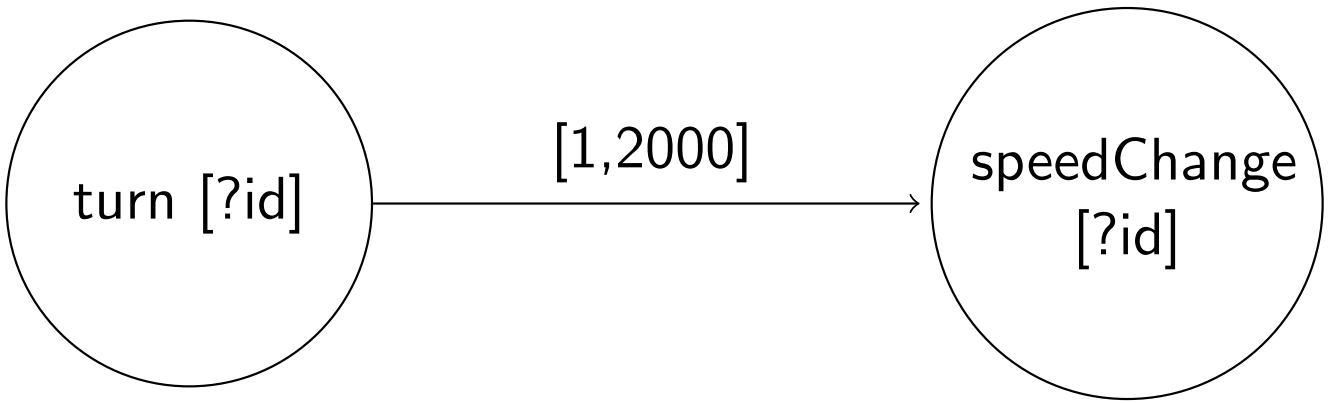
- Mathematical operators in the atemporal constraints of the language are not allowed.
  - No spatial reasoning.
  - No use of background knowledge.
- Universal quantification is not allowed.
  - cannot express a property about all vessels in some area.

CRS is a purely temporal reasoning system.

It is also a very efficient and scalable system.

Chronicle Recognition System: Semantics

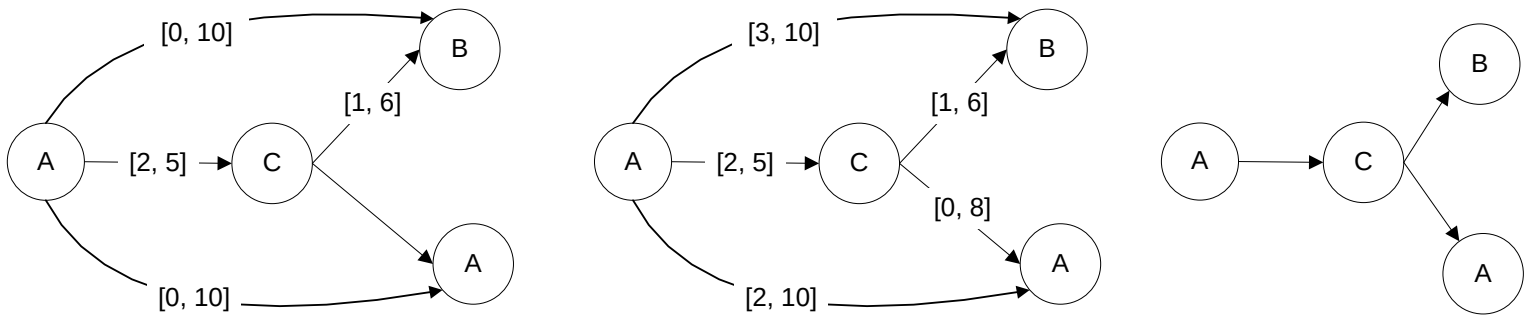
Each CE definition is represented as a Temporal Constraint Network. Eg:



Chronicle Recognition System: Consistency Checking

Compilation stage:

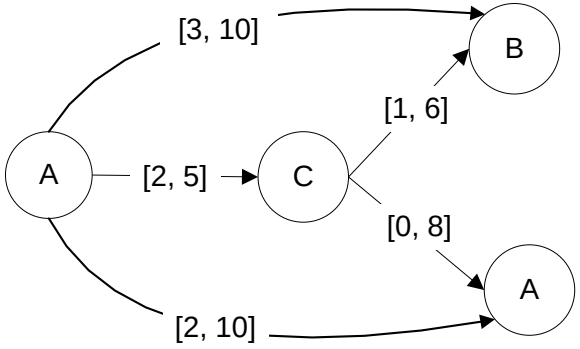
- Constraint propagation in the Temporal Constraint Network.
- Consistency checking.

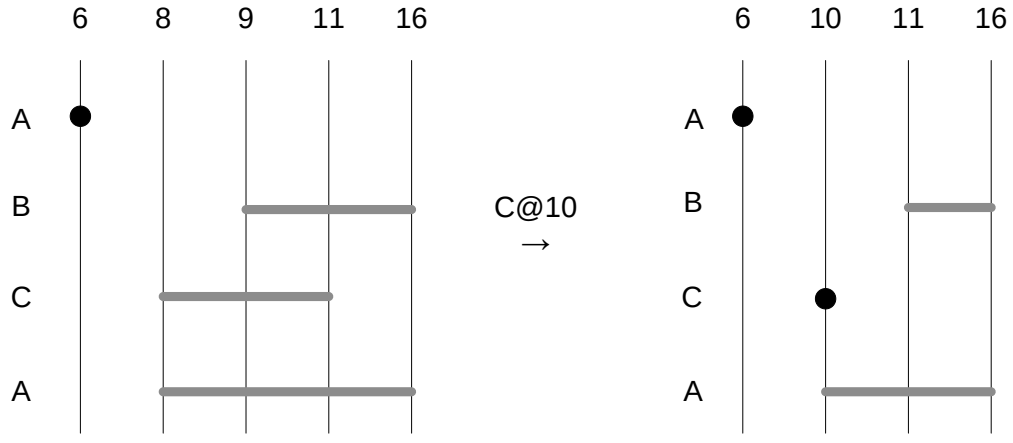


Chronicle Recognition System: Recognition

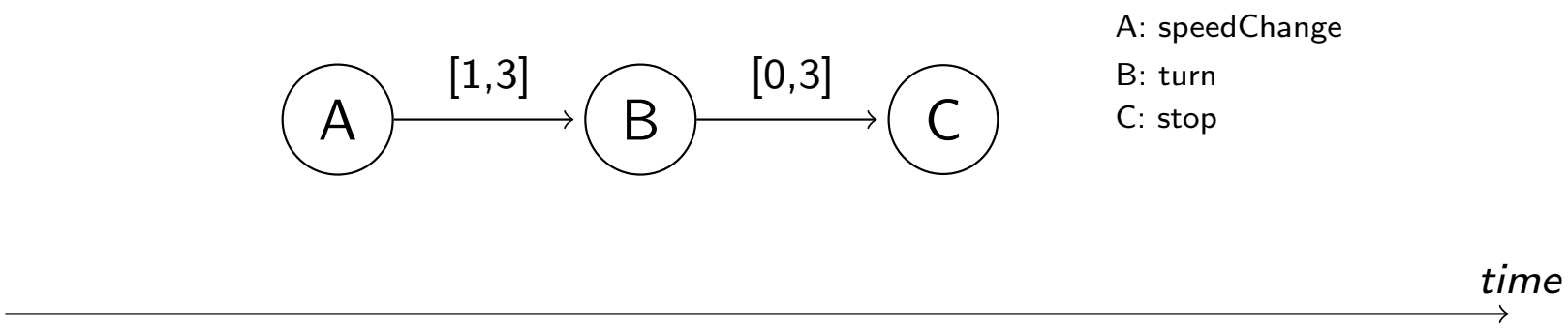
Recognition stage:

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

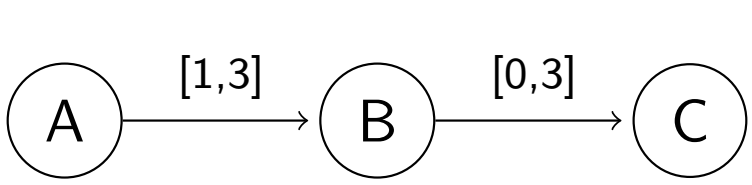




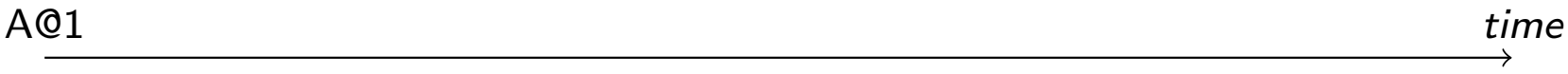
Chronicle Recognition System: Partial instances



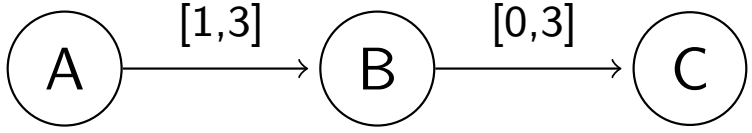
Chronicle Recognition System: Partial instances



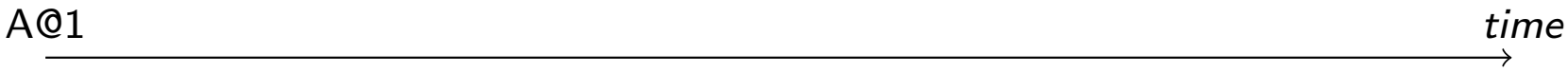
A: speedChange
B: turn
C: stop

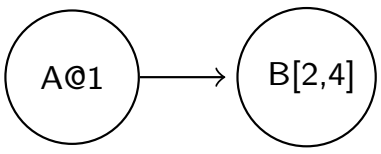


Chronicle Recognition System: Partial instances

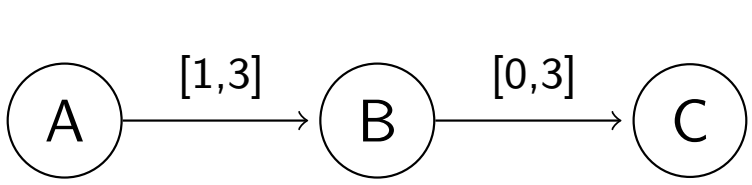


A: speedChange
B: turn
C: stop

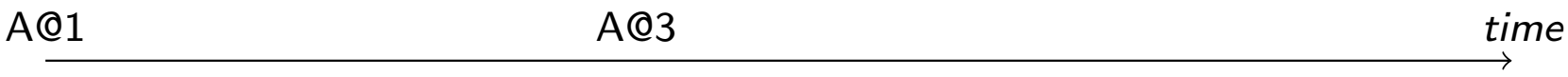


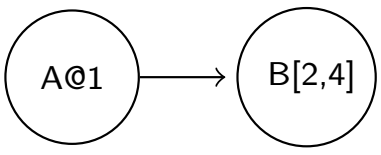


Chronicle Recognition System: Partial instances

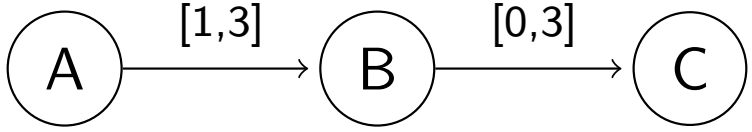


A: speedChange
B: turn
C: stop

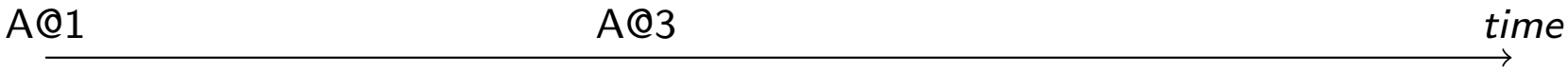


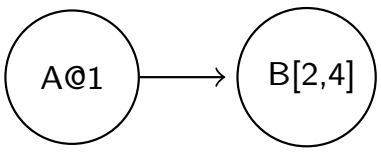


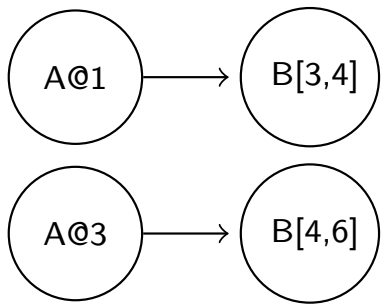
Chronicle Recognition System: Partial instances



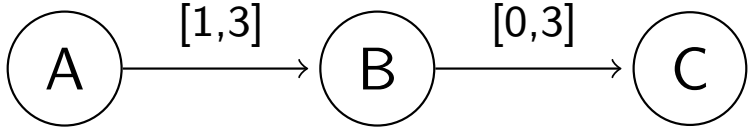
A: speedChange
B: turn
C: stop



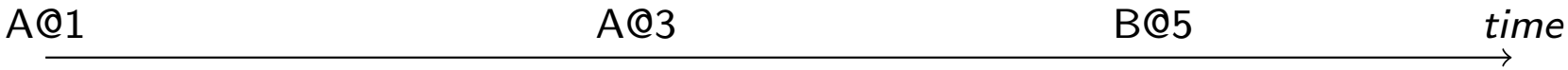


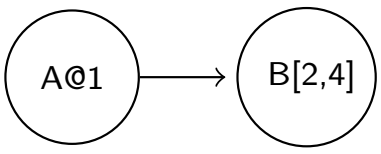


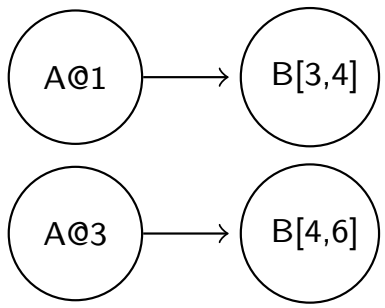
Chronicle Recognition System: Partial instances



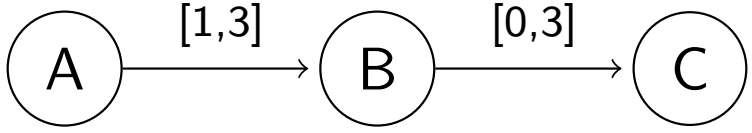
A: speedChange
B: turn
C: stop



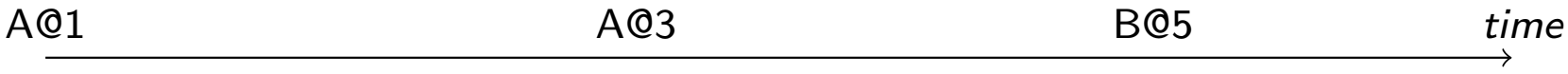


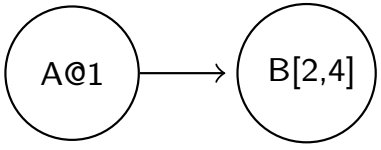


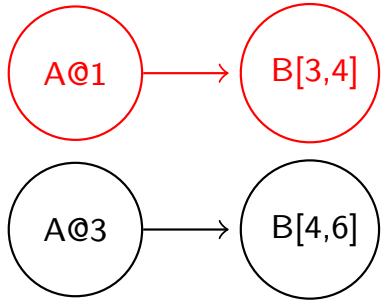
Chronicle Recognition System: Partial instances



A: speedChange
B: turn
C: stop

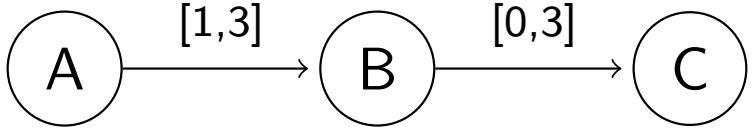




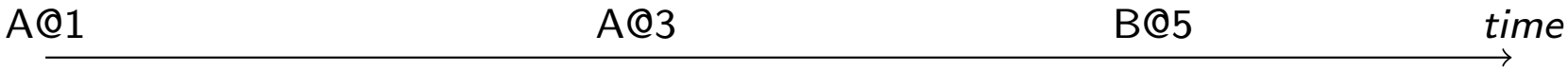


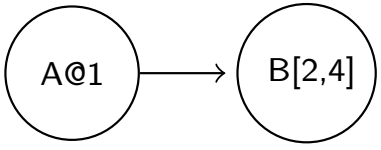
killed instance

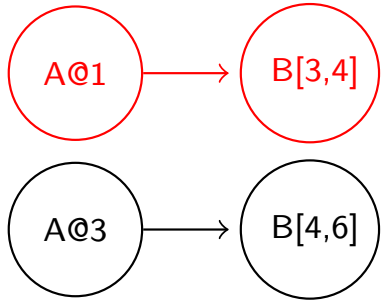
Chronicle Recognition System: Partial instances



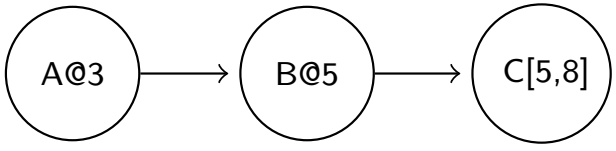
A: speedChange
B: turn
C: stop



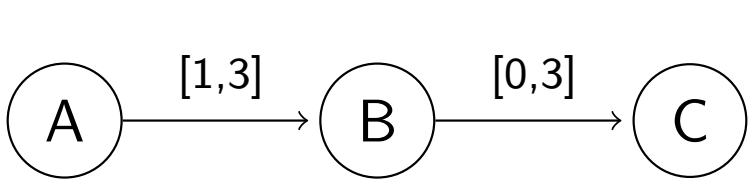




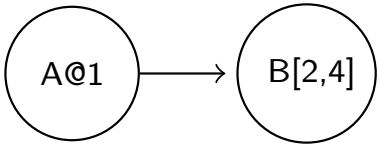
killed instance

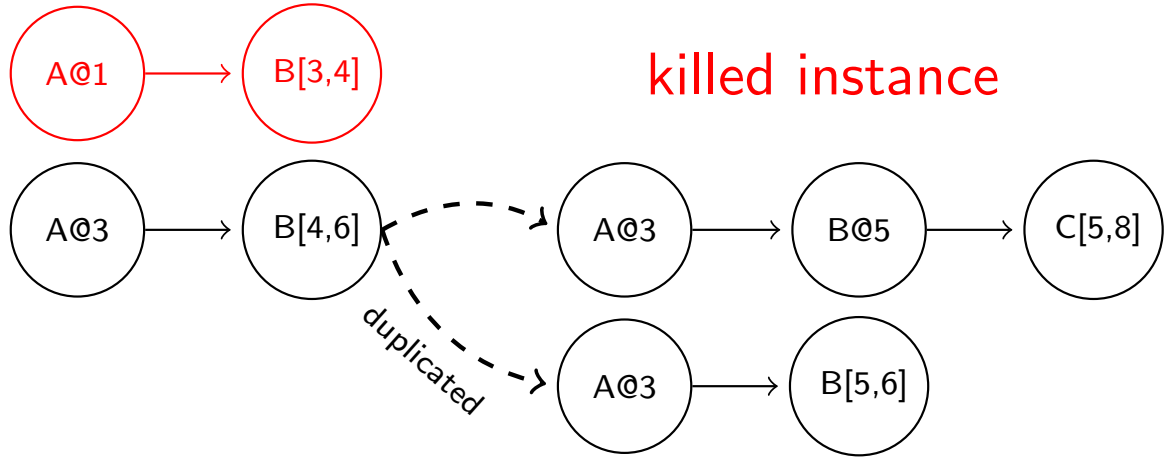


Chronicle Recognition System: Partial instances



A: speedChange
B: turn
C: stop





Chronicle Recognition System

Recognition stage — partial CE instance management:

- In order to manage all the partial CE instances, CRS stores them in trees, one for each CE definition.

Chronicle Recognition System

Recognition stage — partial CE instance management:

- In order to manage all the partial CE instances, CRS stores them in trees, one for each CE definition.
- Each event occurrence and each clock tick traverses these trees in order to kill some CE instances (tree nodes) or to develop some CE instances.

Chronicle Recognition System

Recognition stage — partial CE instance management:

- In order to manage all the partial CE instances, CRS stores them in trees, one for each CE definition.
- Each event occurrence and each clock tick traverses these trees in order to kill some CE instances (tree nodes) or to develop some CE instances.
- To deal with out-of-order SDE streams, CRS keeps in memory partial CE instances longer.

Chronicle Recognition System: Optimisation

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

- Distinguish between rare events and frequent events.

Chronicle Recognition System: Optimisation

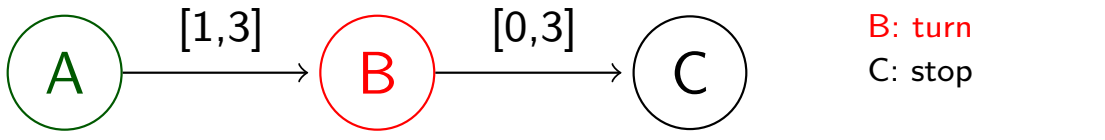
Several techniques have been developed for improving efficiency.
Eg, temporal focusing:

- Distinguish between rare events and frequent events.
- If, according to a CE definition, a rare event should take place after a frequent event, store the incoming frequent events, and start recognition only upon the arrival of the rare events.

Chronicle Recognition System: Optimisation

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

- Distinguish between rare events and frequent events.
- If, according to a CE definition, a rare event should take place after a frequent event, store the incoming frequent events, and start recognition only upon the arrival of the rare events.
- The number of partial CE instances is significantly reduced.
- Example:



Chronicle Recognition System: Summary

- One of the earliest and most successful formal event processing systems.
- Many of its features appear in modern event processing systems.
- Very efficient and scalable event recognition.

Chronicle Recognition System: Summary

- One of the earliest and most successful formal event processing systems.
- Many of its features appear in modern event processing systems.
- Very efficient and scalable event recognition.
- But:
  - It is a purely temporal reasoning system.
  - It does not handle uncertainty.

Event Calculus

- A logic programming language for representing and reasoning about events and their effects.
- Key components:
  - event (typically instantaneous).
  - fluent: a property that may have different values at different points in time.

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.

Run-Time Event Calculus (RTEC)

Predicate	Meaning
happensAt(E, T)	Event E occurs at time T
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
holdsFor$(F = V, I)$	I is the list of the maximal intervals for which $F = V$ holds continuously
holdsAt$(F = V, T)$	The value of fluent F is V at time T
union_all$([J_1, \ldots, J_n],\ I)$	$I = (J_1 \cup \ldots \cup J_n)$
intersect_all$([J_1, \ldots, J_n],\ I)$	$I = (J_1 \cap \ldots \cap J_n)$
relative_complement_all $(I',\ [J_1, \ldots, J_n],\ I)$	$I = I' \setminus (J_1 \cup \ldots \cup J_n)$

Run-Time Event Calculus (RTEC)

Predicate	Meaning
happensAt(E, T)	Event E occurs at time T
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
holdsFor$(F = V, I)$	I is the list of the maximal intervals for which $F = V$ holds continuously
holdsAt$(F = V, T)$	The value of fluent F is V at time T
union_all$([J_1, \ldots, J_n],\ I)$	$I = (J_1 \cup \ldots \cup J_n)$
intersect_all$([J_1, \ldots, J_n],\ I)$	$I = (J_1 \cap \ldots \cap J_n)$
relative_complement_all $(I',\ [J_1, \ldots, J_n],\ I)$	$I = I' \setminus (J_1 \cup \ldots \cup J_n)$

CE Definitions in the Run-Time Event Calculus: Simple Fluents

CE definition:

$$\textbf{initiatedAt}(CE,\ T) \leftarrow \textbf{happensAt}(E_{In_1},\ T), [\text{conditions}]$$
$$\ldots$$
$$\textbf{initiatedAt}(CE,\ T) \leftarrow \textbf{happensAt}(E_{In_i},\ T), [\text{conditions}]$$

$$\textbf{terminatedAt}(CE,\ T) \leftarrow \textbf{happensAt}(E_{T_1},\ T), [\text{conditions}]$$
$$\ldots$$
$$\textbf{terminatedAt}(CE,\ T) \leftarrow \textbf{happensAt}(E_{T_j},\ T), [\text{conditions}]$$

where

$$\text{conditions:} \quad {}^{0-K}\textbf{happensAt}(E_k,\ T), \ {}^{0-M}\textbf{holdsAt}(F_m,\ T), \ {}^{0-N}\text{atemporal-constraint}_n$$

CE Definitions in the Run-Time Event Calculus: Simple Fluents

CE definition:

initiatedAt$(CE,\ T) \leftarrow$
happensAt$(E_{In_1},\ T)$,
[conditions]

...

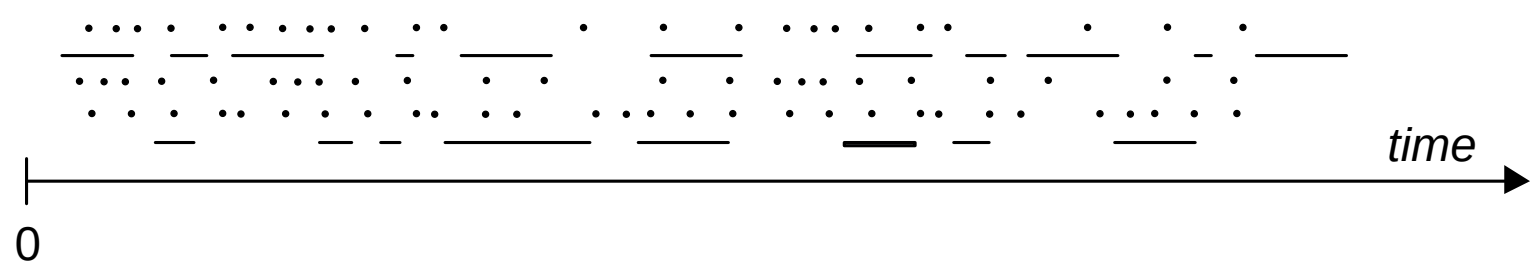
initiatedAt$(CE,\ T) \leftarrow$
happensAt$(E_{In_i},\ T)$,
[conditions]

terminatedAt$(CE,\ T) \leftarrow$
happensAt$(E_{T_1},\ T)$,
[conditions]

...

terminatedAt$(CE,\ T) \leftarrow$
happensAt$(E_{T_j},\ T)$,
[conditions]

CE recognition:



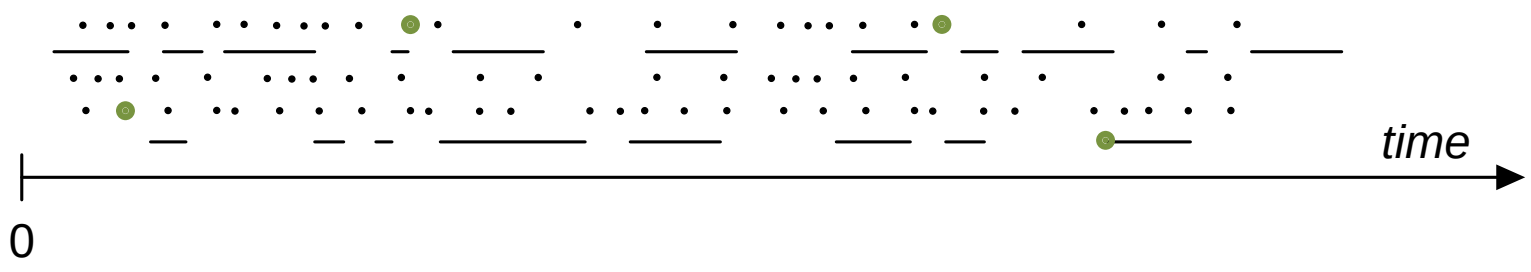
CE Definitions in the Run-Time Event Calculus: Simple Fluents

CE definition:

$$\textbf{initiatedAt}(CE,\ T) \leftarrow \textbf{happensAt}(E_{In_1},\ T), [\text{conditions}]$$
$$\ldots$$
$$\textbf{initiatedAt}(CE,\ T) \leftarrow \textbf{happensAt}(E_{In_i},\ T), [\text{conditions}]$$

$$\textbf{terminatedAt}(CE,\ T) \leftarrow \textbf{happensAt}(E_{T_1},\ T), [\text{conditions}]$$
$$\ldots$$
$$\textbf{terminatedAt}(CE,\ T) \leftarrow \textbf{happensAt}(E_{T_j},\ T), [\text{conditions}]$$

CE recognition:



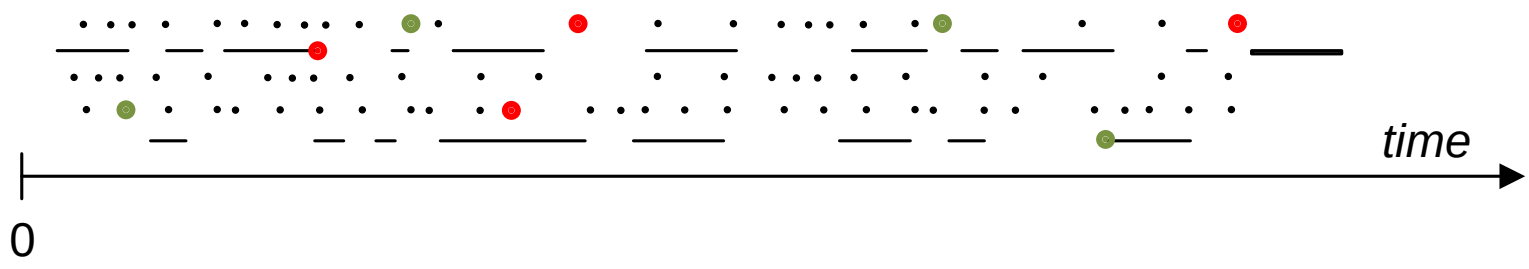
CE Definitions in the Run-Time Event Calculus: Simple Fluents

CE definition:

$$\textbf{initiatedAt}(CE,\ T) \leftarrow \textbf{happensAt}(E_{In_1},\ T), [\text{conditions}]$$
$$\ldots$$
$$\textbf{initiatedAt}(CE,\ T) \leftarrow \textbf{happensAt}(E_{In_i},\ T), [\text{conditions}]$$

$$\textbf{terminatedAt}(CE,\ T) \leftarrow \textbf{happensAt}(E_{T_1},\ T), [\text{conditions}]$$
$$\ldots$$
$$\textbf{terminatedAt}(CE,\ T) \leftarrow \textbf{happensAt}(E_{T_j},\ T), [\text{conditions}]$$

CE recognition:



CE Definitions in the Run-Time Event Calculus: Simple Fluents

CE definition:

initiatedAt$(CE,\ T) \leftarrow$
happensAt$(E_{In_1},\ T)$,
[conditions]

...

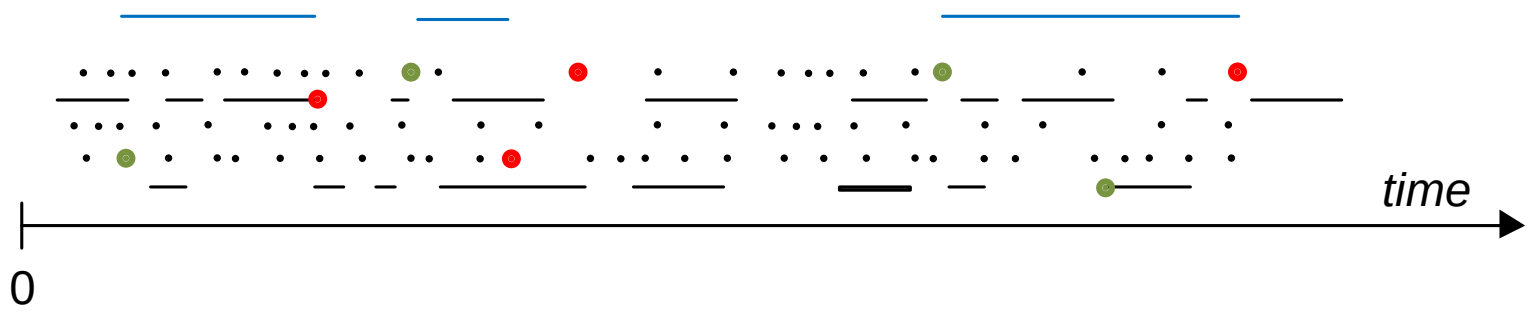
initiatedAt$(CE,\ T) \leftarrow$
happensAt$(E_{In_i},\ T)$,
[conditions]

terminatedAt$(CE,\ T) \leftarrow$
happensAt$(E_{T_1},\ T)$,
[conditions]

...

terminatedAt$(CE,\ T) \leftarrow$
happensAt$(E_{T_j},\ T)$,
[conditions]

CE recognition: **holdsFor**$(CE,\ I)$



CE Definitions in the Run-Time Event Calculus: Simple Fluents

CE definition:

$$\textbf{initiatedAt}(\mathit{leaving\_object}(P, Obj) = \text{true},\ T) \leftarrow$$
$$\quad \textbf{happensAt}(\mathit{appear}(Obj),\ T),$$
$$\quad \textbf{holdsAt}(\mathit{inactive}(Obj) = \text{true},\ T),$$
$$\quad \textbf{holdsAt}(\mathit{close}(P, Obj) = \text{true},\ T),$$
$$\quad \textbf{holdsAt}(\mathit{person}(P) = \text{true},\ T)$$

$$\textbf{terminatedAt}(\mathit{leaving\_object}(P, Obj) = \text{true},\ T) \leftarrow$$
$$\quad \textbf{happensAt}(\mathit{disappear}(Obj),\ T)$$

CE recognition: $\textbf{holdsFor}(\mathit{leaving\_object}(P, Obj) = \text{true},\ I)$

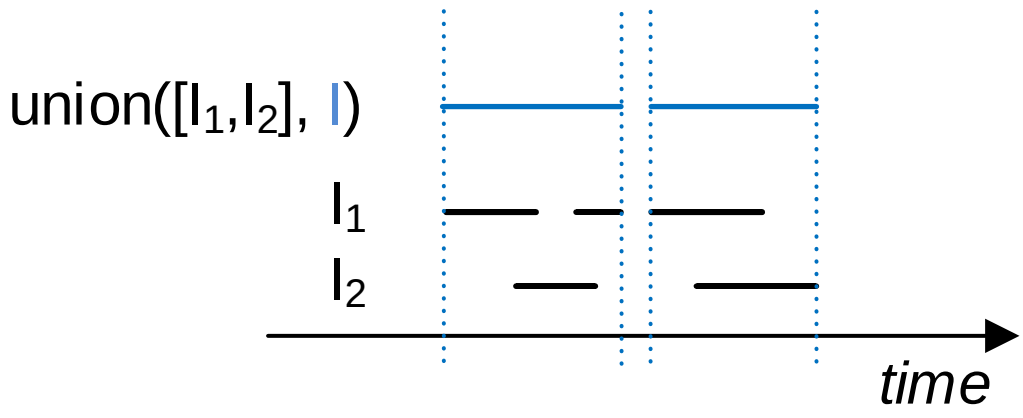
CE Definitions in the Run-Time Event Calculus: Statically Determined Fluents

$$\begin{aligned}
&\textbf{holdsFor}(CE,\ I) \leftarrow \\
&\quad \textbf{holdsFor}(F_1,\ I_{F_1}), \\
&\quad \ldots, \\
&\quad \textbf{holdsFor}(F_f,\ I_{F_f}), \\
&\quad \mathit{interval_manipulation}_1(I_\alpha, \ldots, I_\omega), \\
&\quad \ldots, \\
&\quad \mathit{interval_manipulation}_k(I_A, \ldots, I_\Omega)
\end{aligned}$$

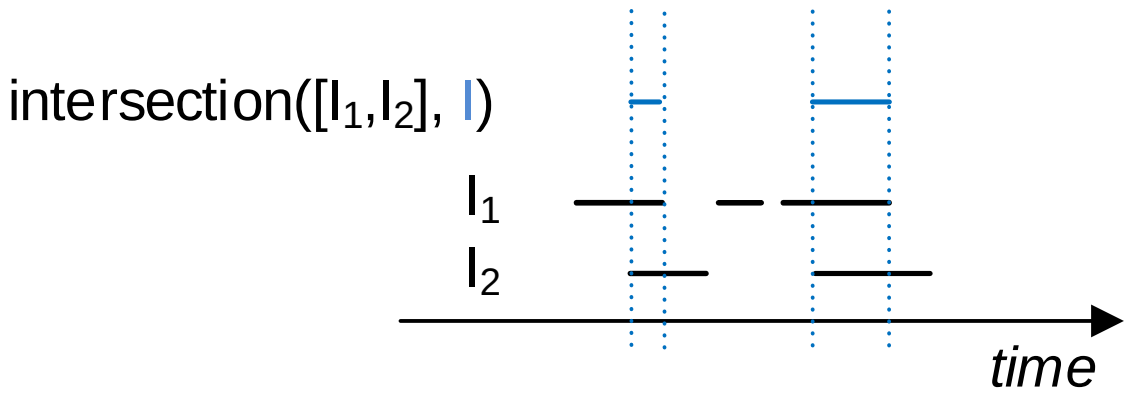
where

$$\begin{aligned}
&\mathit{interval_manipulation}(I_1, \ldots, I_n,\ I) : \\
&\quad \text{union}([I_1, \ldots, I_n],\ I) \\
&\quad \text{intersection}([I_1, \ldots, I_n],\ I) \\
&\quad \text{relative_complement}(I_1,\ [I_2, \ldots, I_n],\ I)
\end{aligned}$$

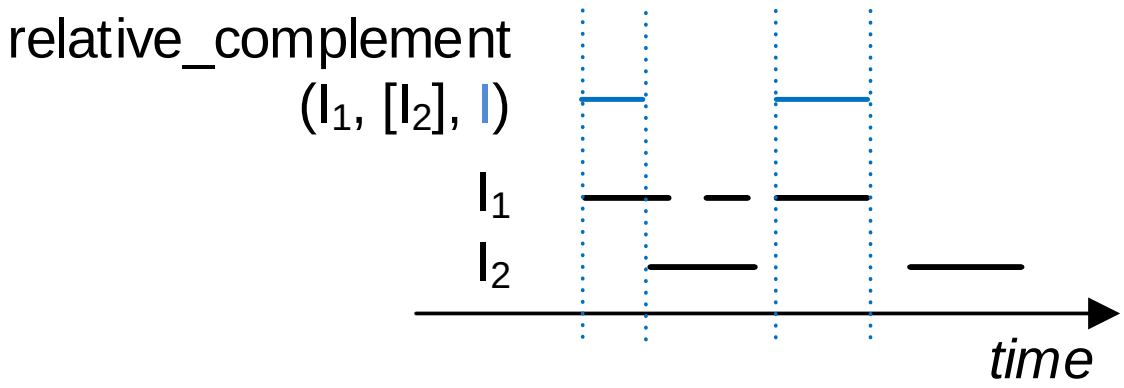
Interval Manipulation: Union



Interval Manipulation: Intersection



Interval Manipulation: Relative Complement



CE Definitions in the Run-Time Event Calculus: Statically Determined Fluents

CE definition:

$$\textbf{holdsFor}(\mathit{abnormal}(\mathit{Vessel}) = \text{true},\ I) \leftarrow$$
$$\textbf{holdsFor}(\mathit{slowMotion}(\mathit{Vessel}) = \text{true},\ I_1),$$
$$\textbf{holdsFor}(\mathit{gap}(\mathit{Vessel}) = \text{true},\ I_2),$$
$$\textbf{holdsFor}(\mathit{stop}(\mathit{Vessel}) = \text{true},\ I_3),$$
$$\text{union}([I_1, I_2, I_3],\ I)$$

CE Definitions in the Run-Time Event Calculus: Statically Determined Fluents

CE definition:

$$\textbf{holdsFor}(abnormal(Vessel) = \text{true},\ I) \leftarrow$$
$$\textbf{holdsFor}(slowMotion(Vessel) = \text{true},\ I_1),$$
$$\textbf{holdsFor}(gap(Vessel) = \text{true},\ I_2),$$
$$\textbf{holdsFor}(stop(Vessel) = \text{true},\ I_3),$$
$$\text{union}([I_1, I_2, I_3],\ I)$$

Shorthand:

$$abnormal(Vessel) \text{ iff}$$
$$slowMotion(Vessel) \text{ or}$$
$$gap(Vessel) \text{ or}$$
$$idle(Vessel)$$

CE Definitions in the Run-Time Event Calculus: Statically Determined Fluents

Shorthand:

$$
\begin{aligned}
&\mathit{suspicious}(\mathit{Vessel}_1, \mathit{Vessel}_2) \text{ iff} \\
&\quad \mathit{abnormal}(\mathit{Vessel}_1), \\
&\quad \mathit{abnormal}(\mathit{Vessel}_2), \\
&\quad \mathit{close}(\mathit{Vessel}_1, \mathit{Vessel}_2), \\
&\quad \text{not } (\mathit{in}(\mathit{Vessel}_1, \mathit{Area}) \text{ or } \mathit{in}(\mathit{Vessel}_2, \mathit{Area}))
\end{aligned}
$$

CE Definitions in the Run-Time Event Calculus: Statically Determined Fluents

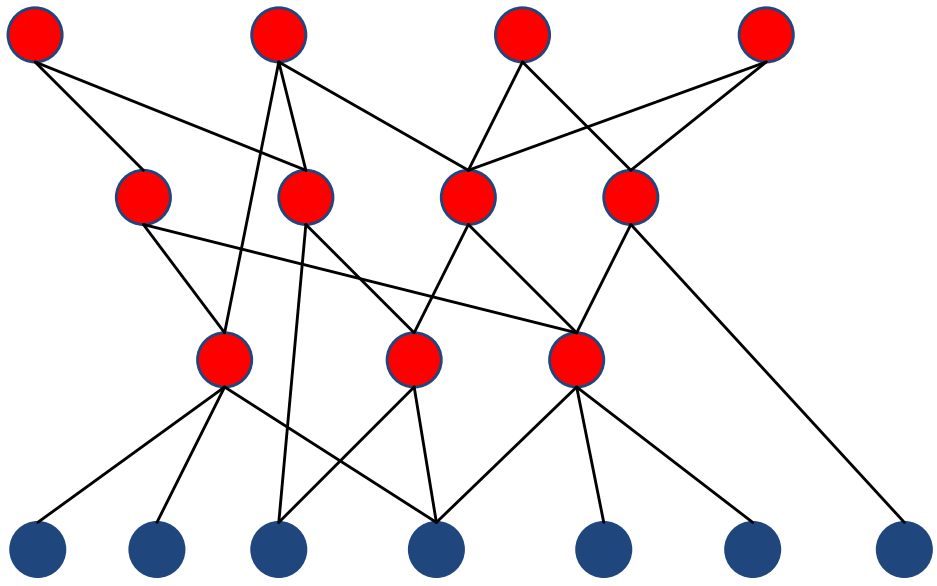
Shorthand:

$$\begin{aligned} &suspicious(Vessel_1, Vessel_2) \text{ iff} \\ &\quad abnormal(Vessel_1), \\ &\quad abnormal(Vessel_2), \\ &\quad close(Vessel_1, Vessel_2), \\ &\quad \text{not } (in(Vessel_1, Area) \text{ or } in(Vessel_2, Area)) \end{aligned}$$

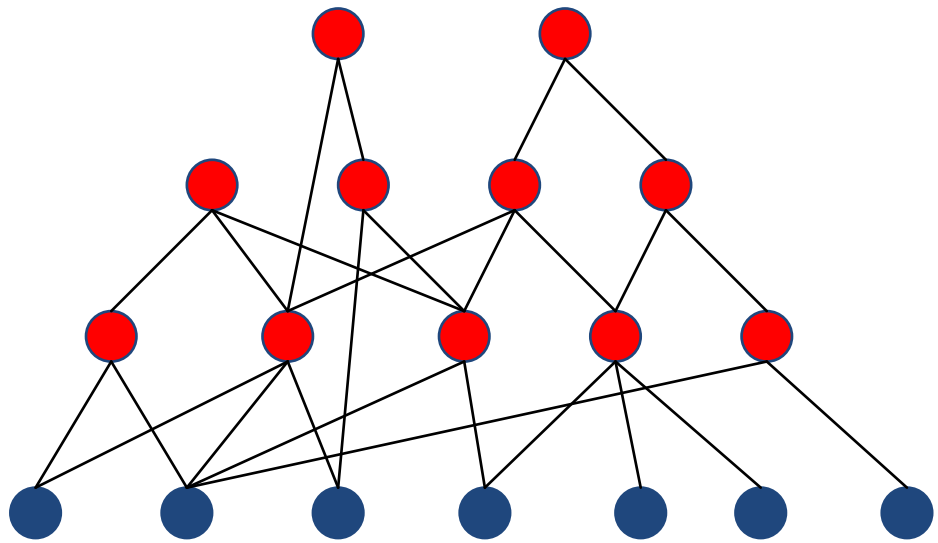
CE definition:

$$\begin{aligned} &\textbf{holdsFor}(suspicious(Vessel_1, Vessel_2) = \text{true},\ I) \leftarrow \\ &\quad \textbf{holdsFor}(abnormal(Vessel_1) = \text{true},\ I_1), \\ &\quad \textbf{holdsFor}(abnormal(Vessel_2) = \text{true},\ I_2), \\ &\quad \textbf{holdsFor}(close(Vessel_1, Vessel_2) = \text{true},\ I_3), \\ &\quad \textbf{holdsFor}(in(Vessel_1, Area) = \text{true},\ I_4), \\ &\quad \textbf{holdsFor}(in(Vessel_2, Area) = \text{true},\ I_5), \\ &\quad \text{intersection}([I_1, I_2, I_3],\ I_6), \\ &\quad \text{relative\_complement}(I_6,\ [I_4, I_5],\ I) \end{aligned}$$

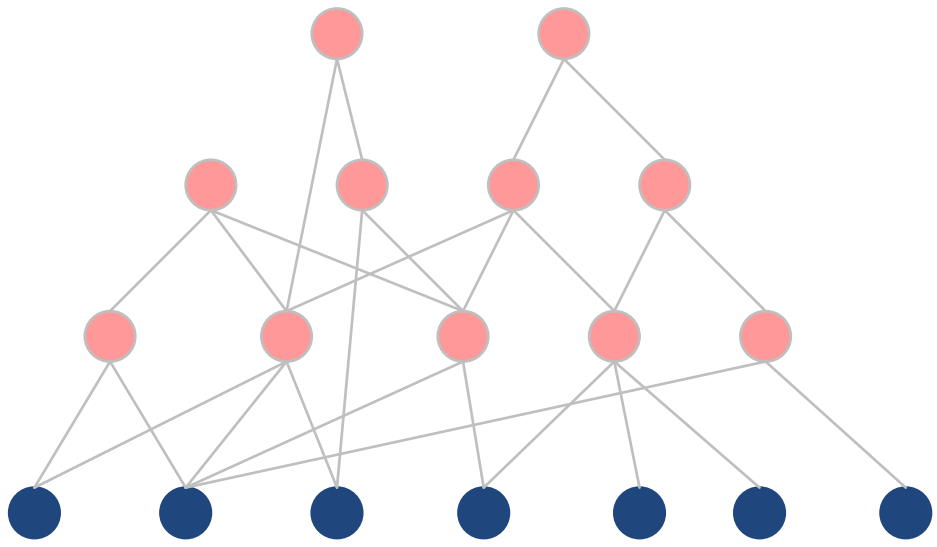
CE Hierarchies



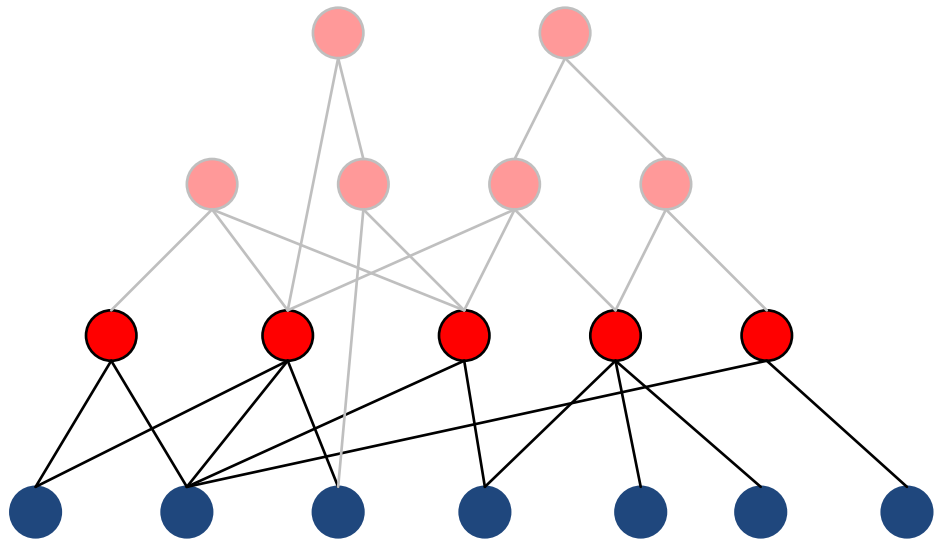
CE Hierarchies



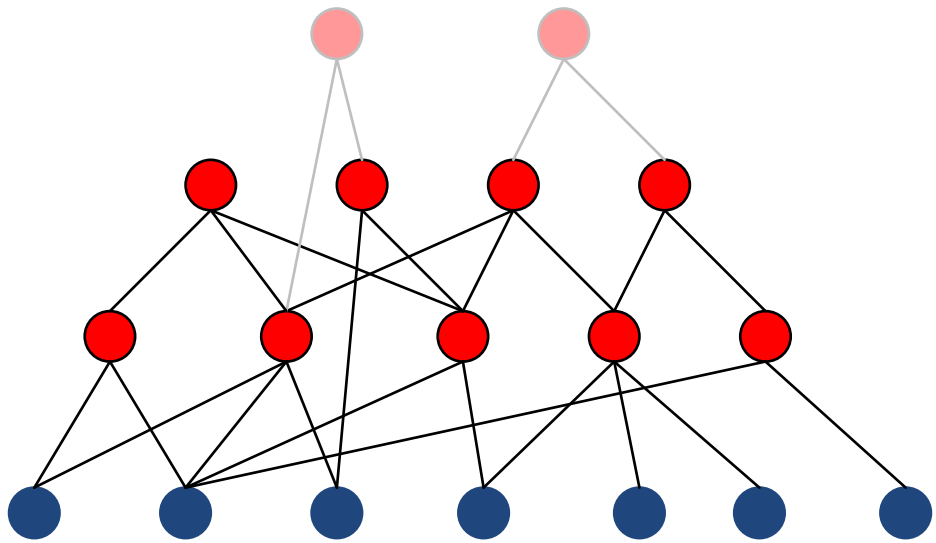
CE Hierarchies: Caching



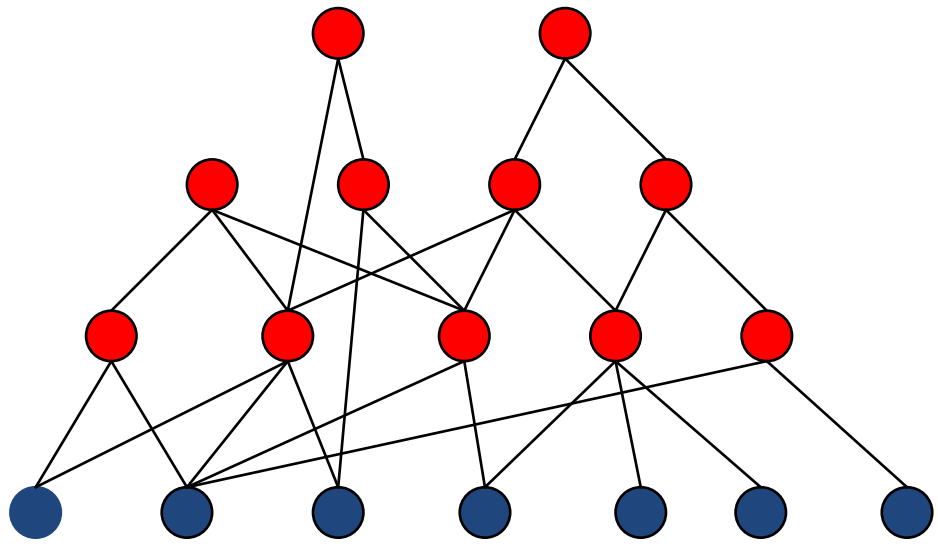
CE Hierarchies: Caching



CE Hierarchies: Caching



CE Hierarchies: Caching

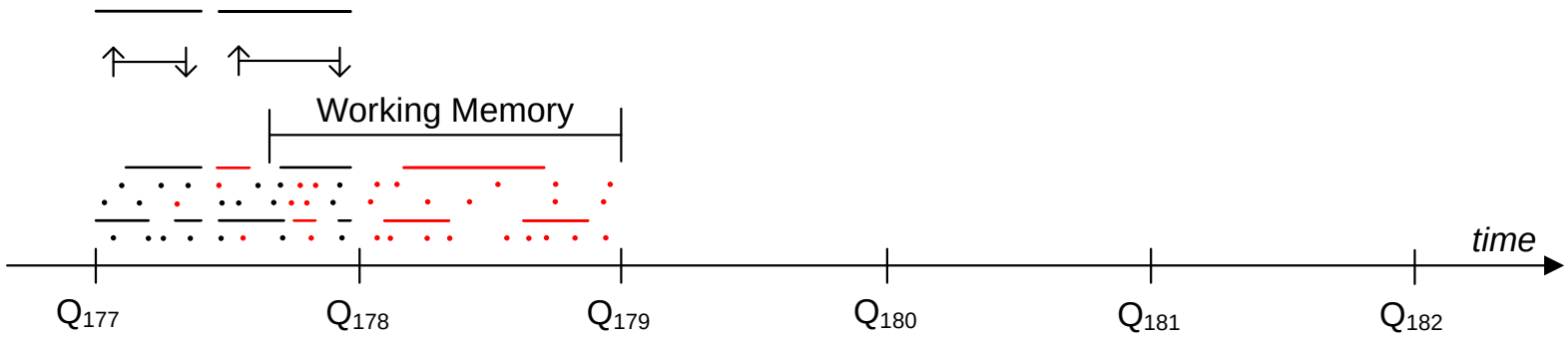


Run-Time Event Recognition

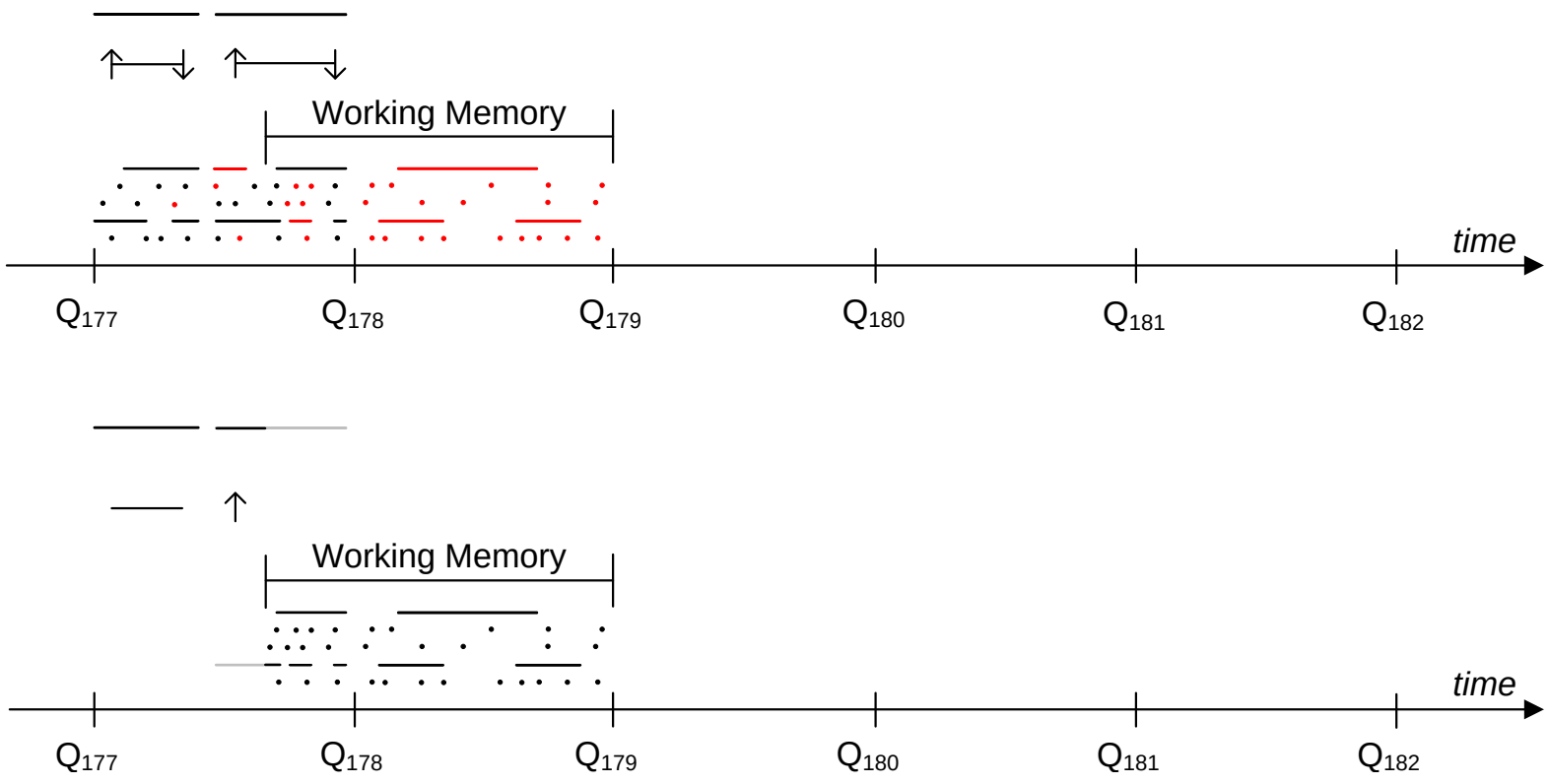
Real-time decision-support in the presence of:

- Very large SDE streams.
- Non-sorted SDE streams.
- SDE revision.
- Very large CE numbers.

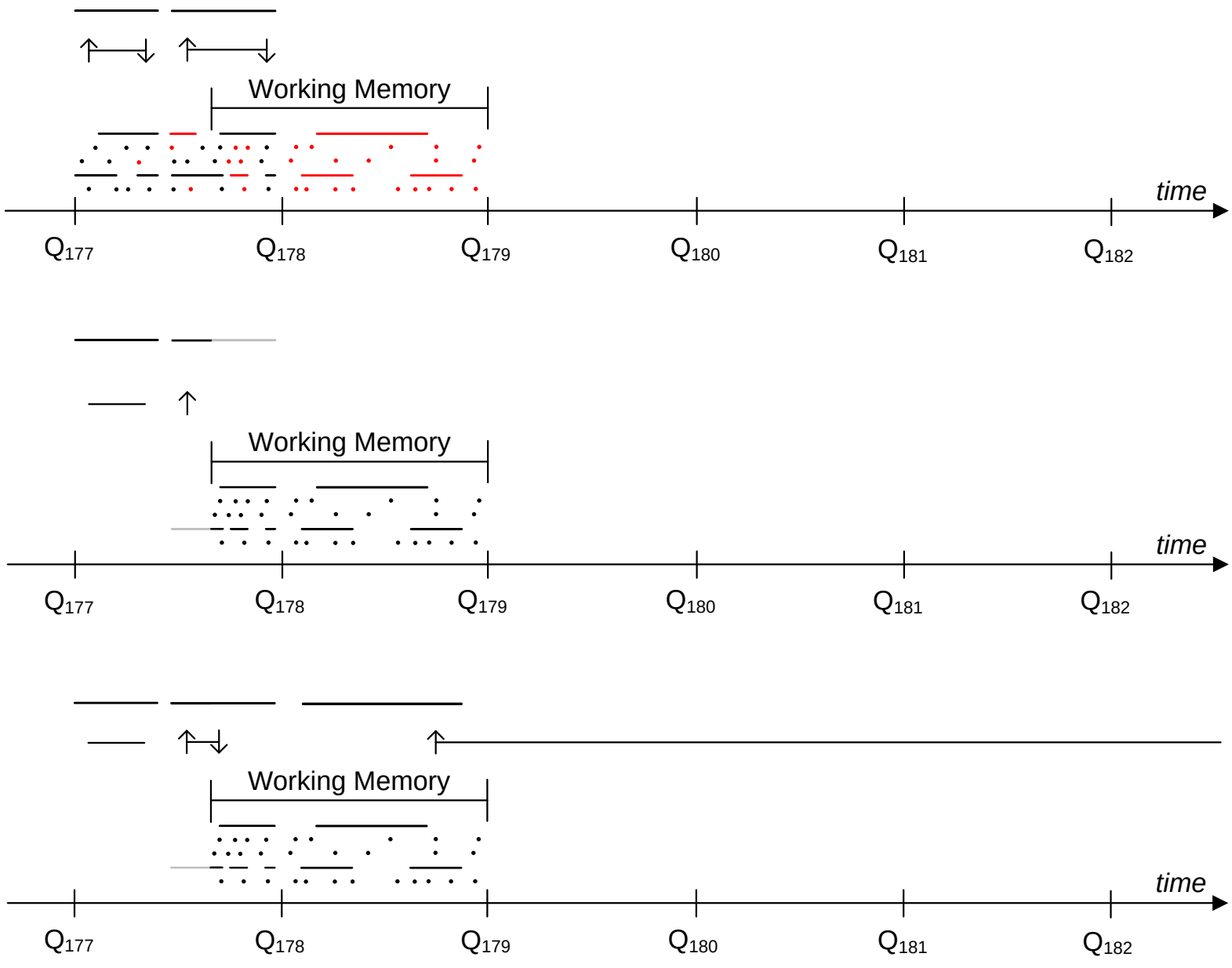
Run-Time Event Calculus: Windowing



Run-Time Event Calculus: Windowing



Run-Time Event Calculus: Windowing



Run-Time Event Calculus: Summary

- Representation of complex temporal phenomena.
  - Succinct representation $\rightarrow$ code maintenance.
  - Intuitive representation $\rightarrow$ domain experts unfamiliar with programming into the loop.

Run-Time Event Calculus: Summary

- Representation of complex temporal phenomena.
  - Succinct representation $\rightarrow$ code maintenance.
  - Intuitive representation $\rightarrow$ domain experts unfamiliar with programming into the loop.
- The full power of logic programming is available.
  - Complex atemporal computations.
  - Combination of events streams with static knowledge.

Run-Time Event Calculus: Summary

- Representation of complex temporal phenomena.
  - Succinct representation $\rightarrow$ code maintenance.
  - Intuitive representation $\rightarrow$ domain experts unfamiliar with programming into the loop.
- The full power of logic programming is available.
  - Complex atemporal computations.
  - Combination of events streams with static knowledge.
- Very efficient reasoning.
  - Even when event streams arrive with a delay.
  - Even in the presence of large specifications.

Run-Time Event Calculus: Summary

- Representation of complex temporal phenomena.
  - Succinct representation $\rightarrow$ code maintenance.
  - Intuitive representation $\rightarrow$ domain experts unfamiliar with programming into the loop.
- The full power of logic programming is available.
  - Complex atemporal computations.
  - Combination of events streams with static knowledge.
- Very efficient reasoning.
  - Even when event streams arrive with a delay.
  - Even in the presence of large specifications.
- But:
  - The Event Calculus does not deal with uncertainty.

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 CE definition development.

Common Problems of Event Recognition

- Limited dictionary of SDE and context variables.
  - No explicit representation of oil spillage.

Common Problems of Event Recognition

- Limited dictionary of SDE and context variables.
  - No explicit representation of oil spillage.
- Incomplete SDE stream.
  - Sharp turn was not detected.

Common Problems of Event Recognition

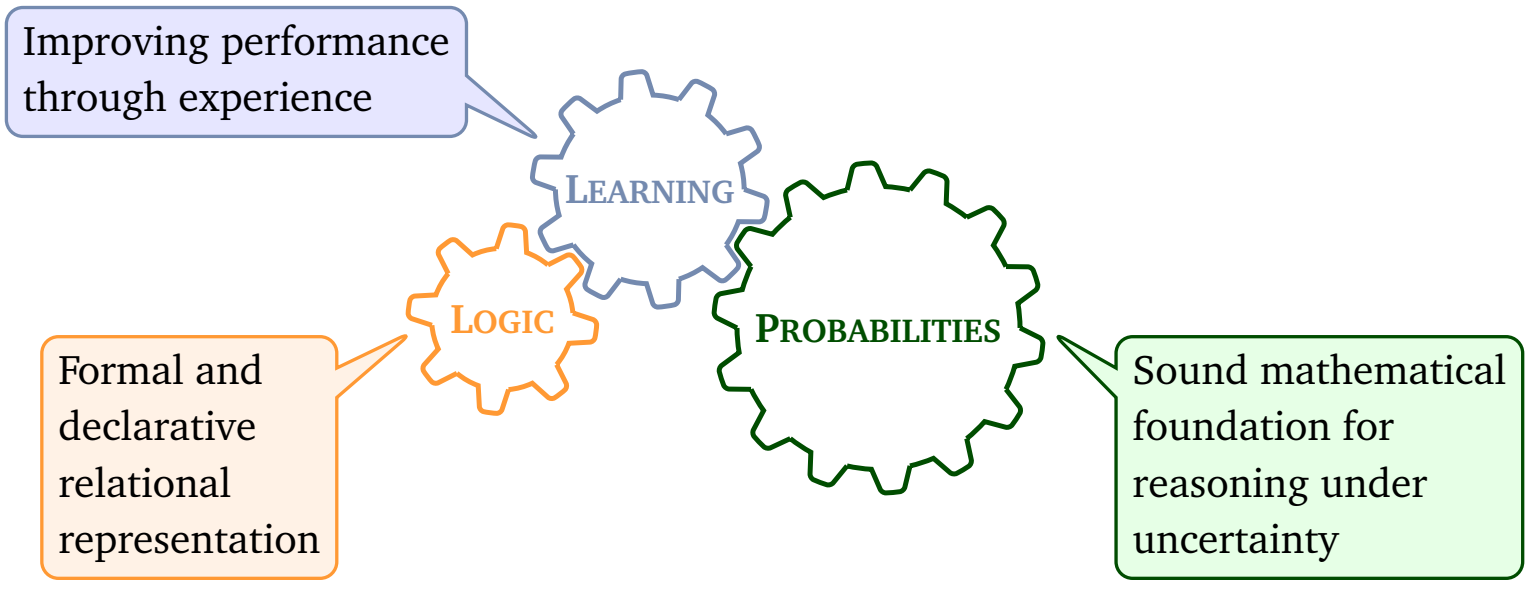
- Limited dictionary of SDE and context variables.
  - No explicit representation of oil spillage.
- Incomplete SDE stream.
  - Sharp turn was not detected.
- Erroneous SDE detection.
  - Slow motion was classified as stop.

Common Problems of Event Recognition

- Limited dictionary of SDE and context variables.
  - No explicit representation of oil spillage.
- Incomplete SDE stream.
  - Sharp turn was not detected.
- Erroneous SDE detection.
  - Slow motion was classified as stop.
- Inconsistent ground truth (CE & SDE annotation).
  - Disagreement between (human) annotators.

Therefore, an adequate treatment of uncertainty is required.

Statistical Relational Learning



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.

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.

ProbLog

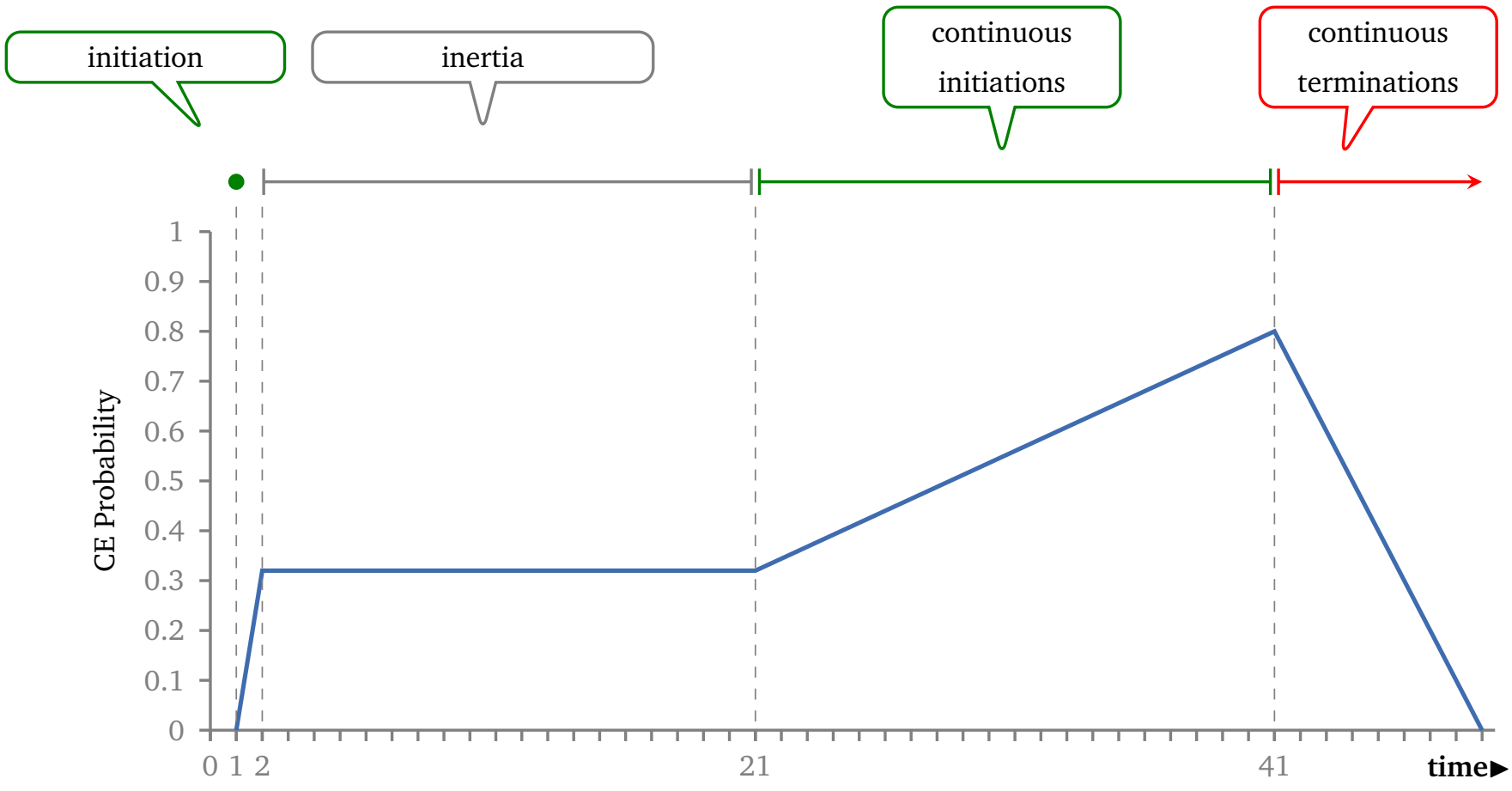
- A probabilistic logic programming language.
- Allows for independent probabilistic facts prob::fact.
  - prob indicates the probability that fact is part of a possible world.
- Rules are written as in classic Prolog.
- The probability of a query q imposed on a ProbLog database *(success probability)*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

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

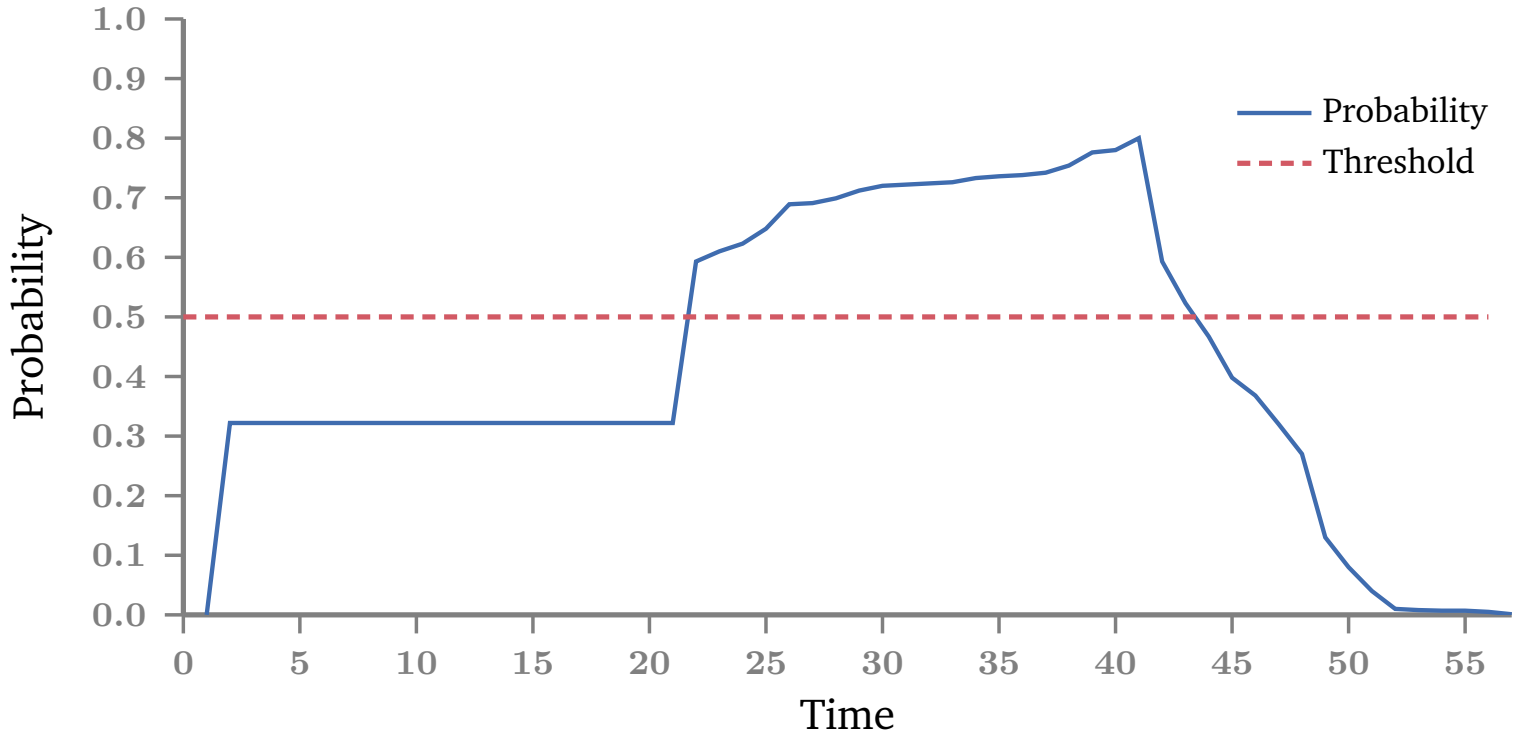
Event Recognition using ProbLog

Input	Output
340 0.45 :: $inactive(id_0)$	340 0.41 :: $left\_object(id_1, id_0)$
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



Markov Logic Networks (MLN)

SYNTAX: weighted first-order logic formulas (w_i, F_i)

When input events SDE_A **and** SDE_B occur at T,
then the output event CE is initiated:

3.18 $happensAt(SDE_A,\ T) \wedge happensAt(SDE_B,\ T) \Rightarrow initiatedAt(CE,\ T)$

Markov Logic Networks (MLN)

Syntax: weighted first-order logic formulas (w_i, F_i)

When input events SDE_A **and** SDE_B occur at T,
then the output event CE is initiated:

3.18 $happensAt(SDE_A,\ T) \wedge happensAt(SDE_B,\ T) \Rightarrow initiatedAt(CE,\ T)$

Semantics: (w_i, F_i) represents a probability distribution over possible worlds

$$P(Y=\mathbf{y} \mid X=\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp\left(\sum_i w_i\, n_i(\mathbf{x}, \mathbf{y})\right)$$

SDEs

weight of the i-th formula

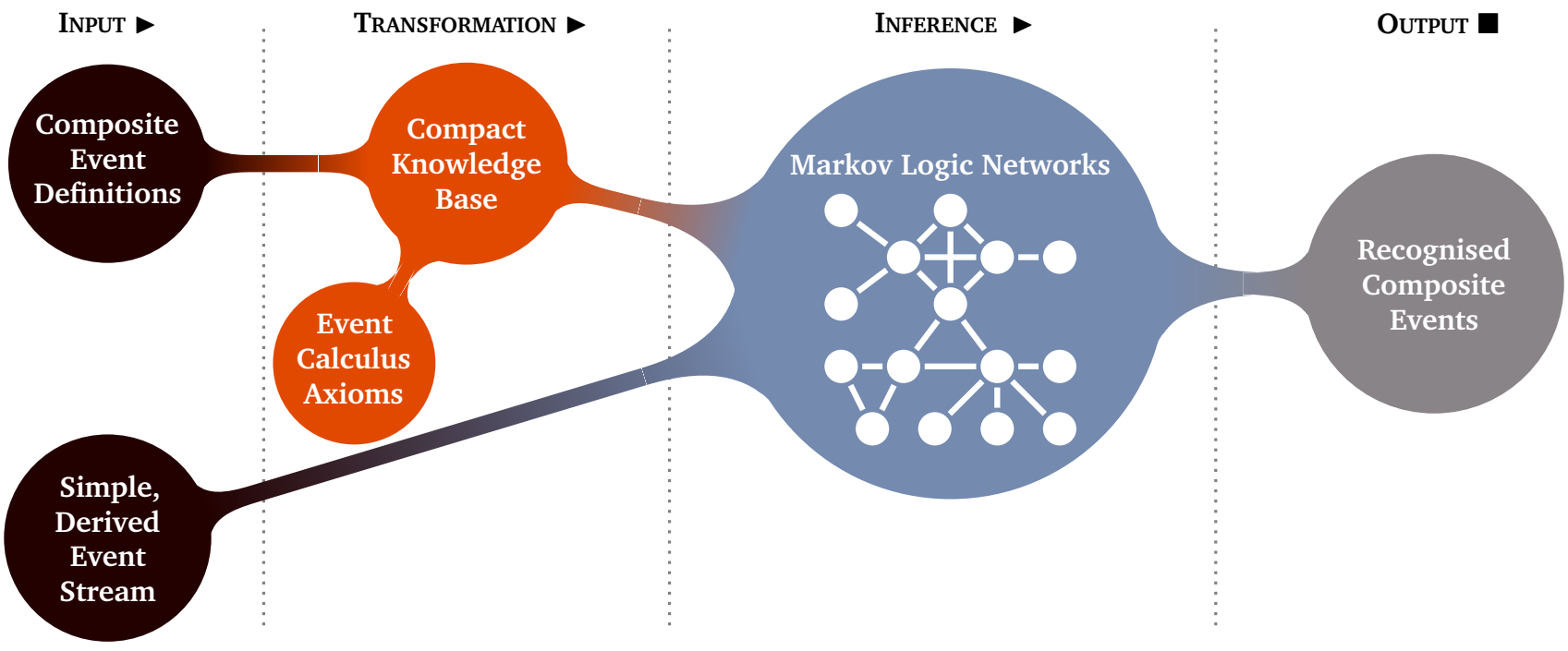
Possible world: CEs

Partition function

number of satisfied groundings

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

Event Calculus in Markov Logic Networks (MLN-EC)



MLN-EC: Probabilistic Inference

Marginal Inference:

- For all time points T, calculate the probability of each **CE** being true (recognised), given all **input SDEs** (evidence)

$$P(\texttt{holdsAt}(CE,\ T){=}true|SDEs)$$

- Marginal inference is #P-complete $\rightarrow$ approximate inference
- MC-SAT algorithm (Markov Chain Monte Carlo techniques with SAT solver)

MLN-EC: Probabilistic Inference

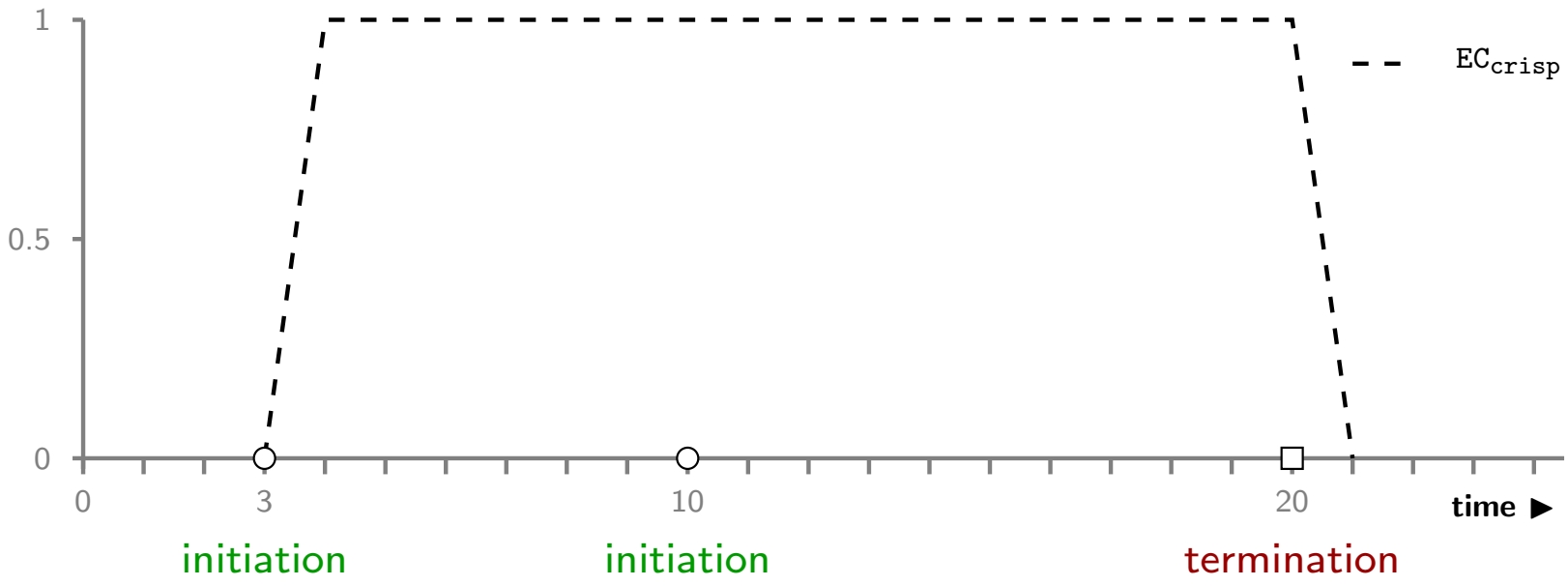
Maximum a Posteriori (MAP) Inference:

- Find the world with the highest probability
- **Input:** truth values for all **input SDEs** (evidence)
- **Output:** truth values of the **output CEs** that maximise the probability (recognition)

$$\underset{\texttt{holdsAt}(CE,T)}{\text{argmax}} \Big(P(\texttt{holdsAt}(CE,\ T)|SDEs) \Big)$$

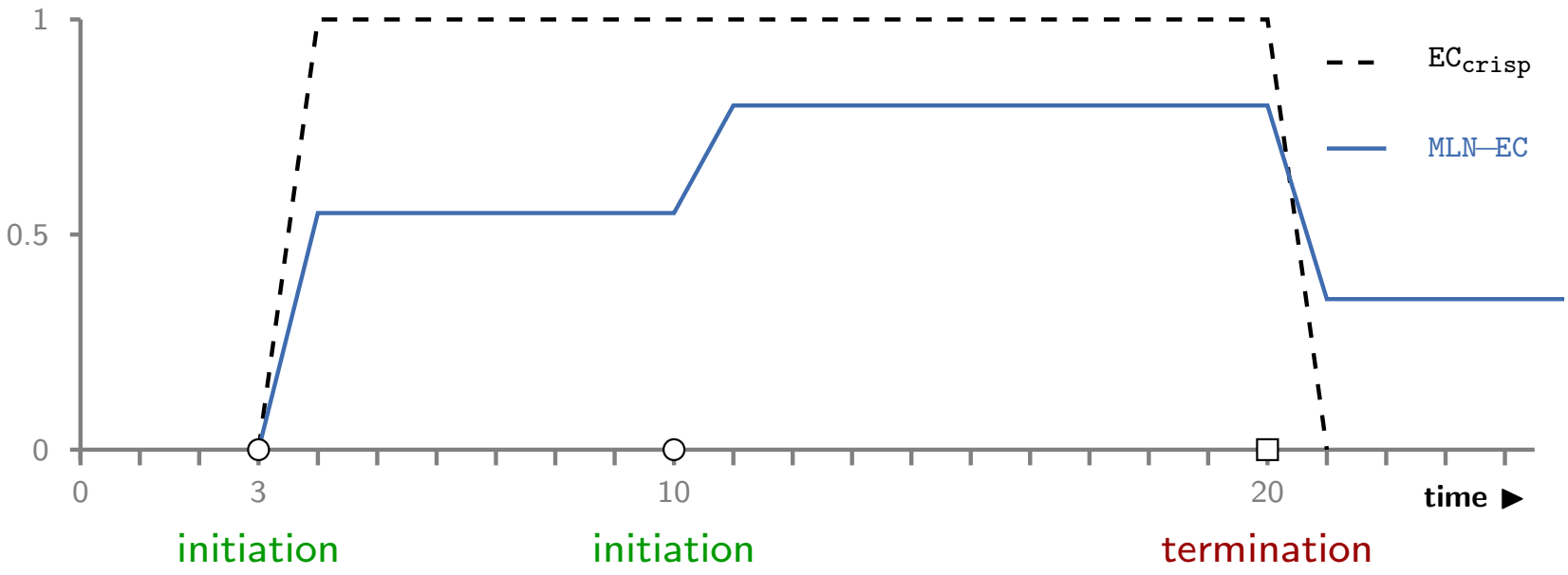
- MAP Inference is NP-hard $\rightarrow$ approximate inference
- Various methods: local search, linear programming, etc.

MLN-EC: Inertia



- ∞ holdsAt(CE, T+1)$\Leftarrow$ [Initiation Conditions]
- ∞ $\neg$holdsAt(CE, T+1)$\Leftarrow$ [Termination Conditions]
- ∞ $\neg$holdsAt(CE, T+1)$\Leftarrow$ $\neg$holdsAt(CE), T) $\wedge$ $\neg$[Initiation Conditions]
- ∞ holdsAt(CE, T+1)$\Leftarrow$ holdsAt(CE, T) $\wedge$ $\neg$[Termination Conditions]

MLN-EC: Inertia



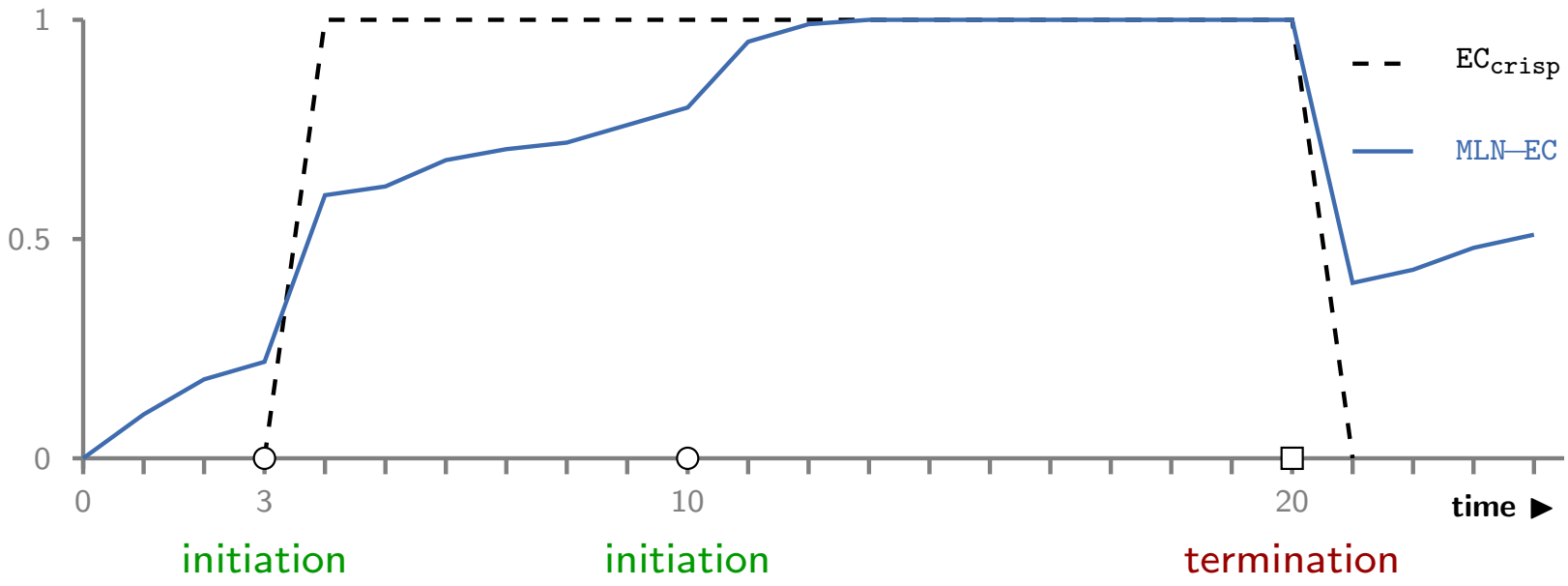
1.2 $\text{holdsAt}(CE,\ T+1) \Leftarrow$
$[\texttt{Initiation Conditions}]$

∞ $\neg\text{holdsAt}(CE,\ T+1) \Leftarrow$
$\neg\text{holdsAt}(CE),\ T) \wedge$
$\neg[\texttt{Initiation Conditions}]$

0.7 $\neg\text{holdsAt}(CE,\ T+1) \Leftarrow$
$[\texttt{Termination Conditions}]$

∞ $\text{holdsAt}(CE,\ T+1) \Leftarrow$
$\text{holdsAt}(CE,\ T) \wedge$
$\neg[\texttt{Termination Conditions}]$

MLN-EC: Inertia



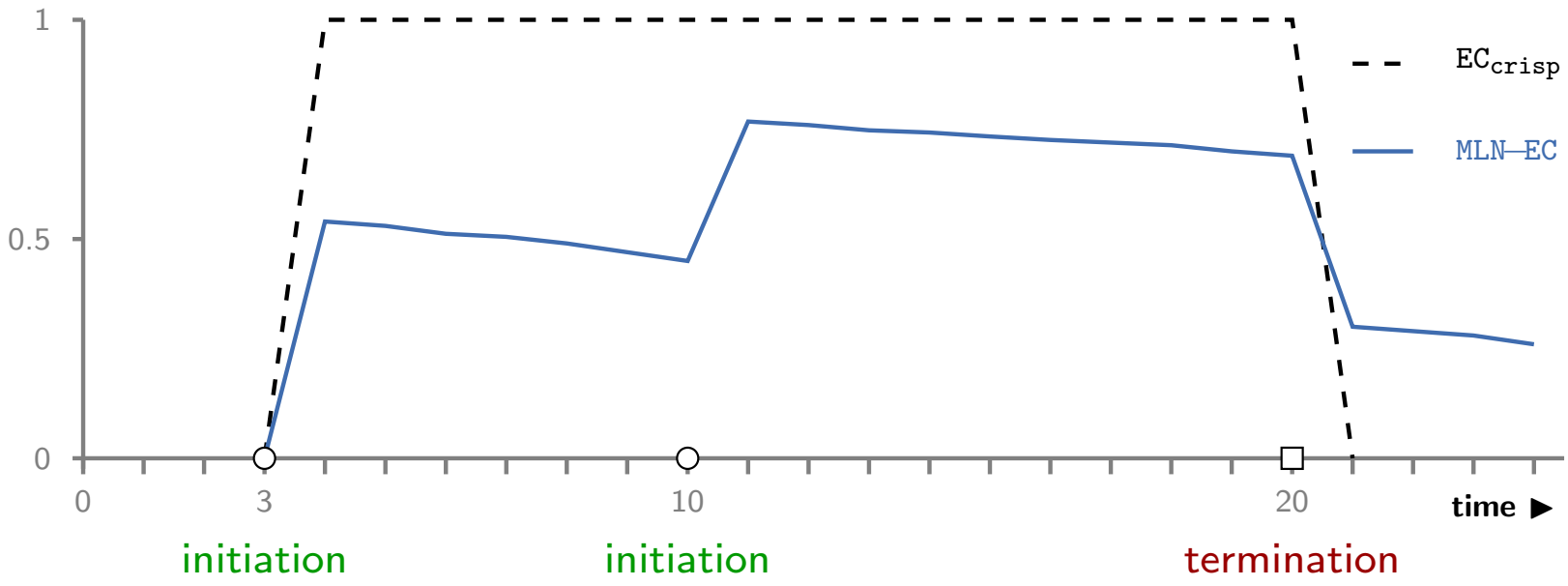
1.2 holdsAt(CE, T+1)⇐
[Initiation Conditions]

2.3 ¬holdsAt(CE, T+1)⇐
¬holdsAt(CE), T) ∧
¬[Initiation Conditions]

0.7 ¬holdsAt(CE, T+1)⇐
[Termination Conditions]

∞ holdsAt(CE, T+1)⇐
holdsAt(CE, T) ∧
¬[Termination Conditions]

MLN-EC: Inertia



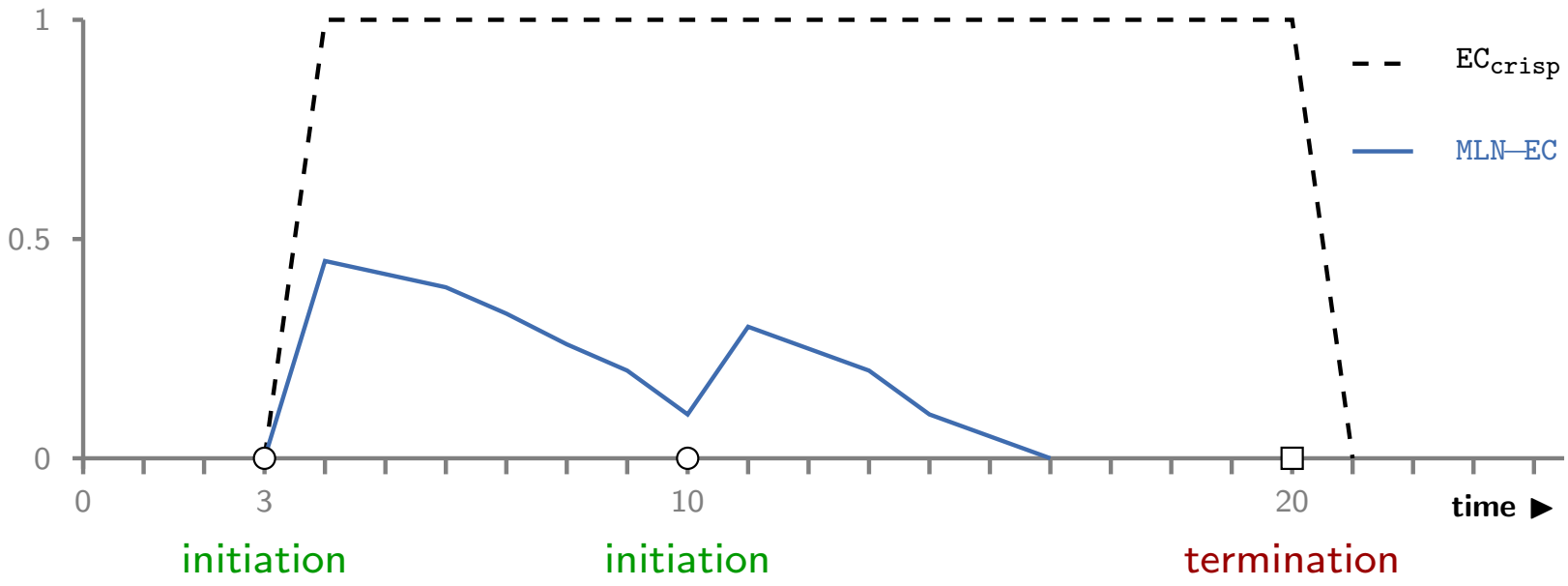
1.2 holdsAt(CE, T+1)⇐
[Initiation Conditions]

∞ ¬holdsAt(CE, T+1)⇐
¬holdsAt(CE), T) ∧
¬[Initiation Conditions]

0.7 ¬holdsAt(CE, T+1)⇐
[Termination Conditions]

2.3 holdsAt(CE, T+1)⇐
holdsAt(CE, T) ∧
¬[Termination Conditions]

MLN-EC: Inertia



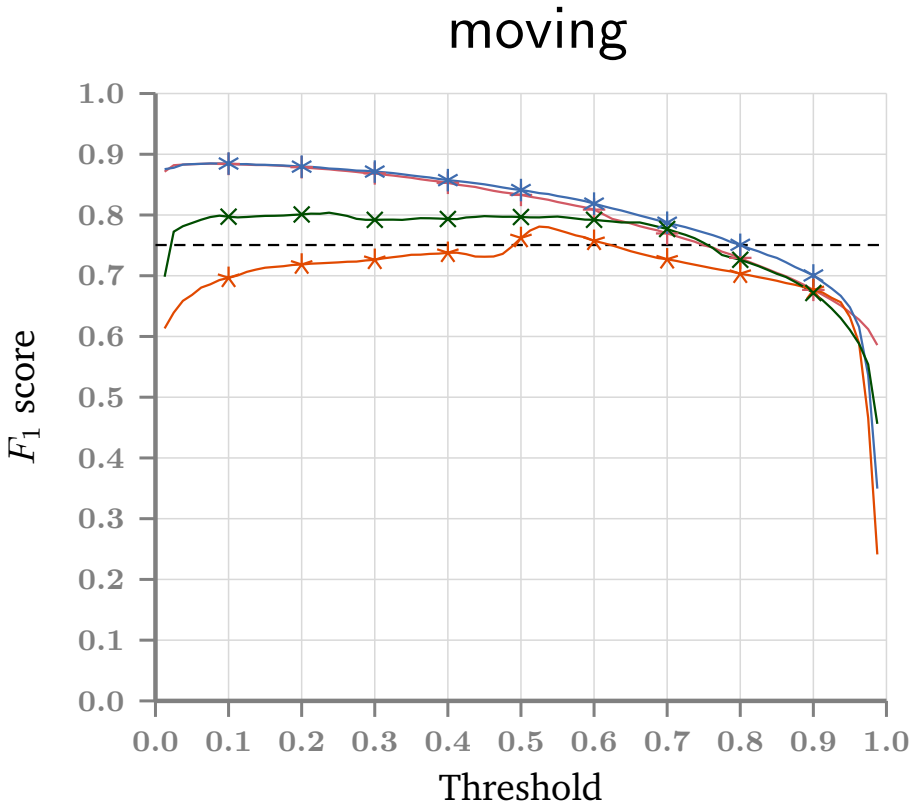
1.2 holdsAt(CE, T+1)⇐
[Initiation Conditions]

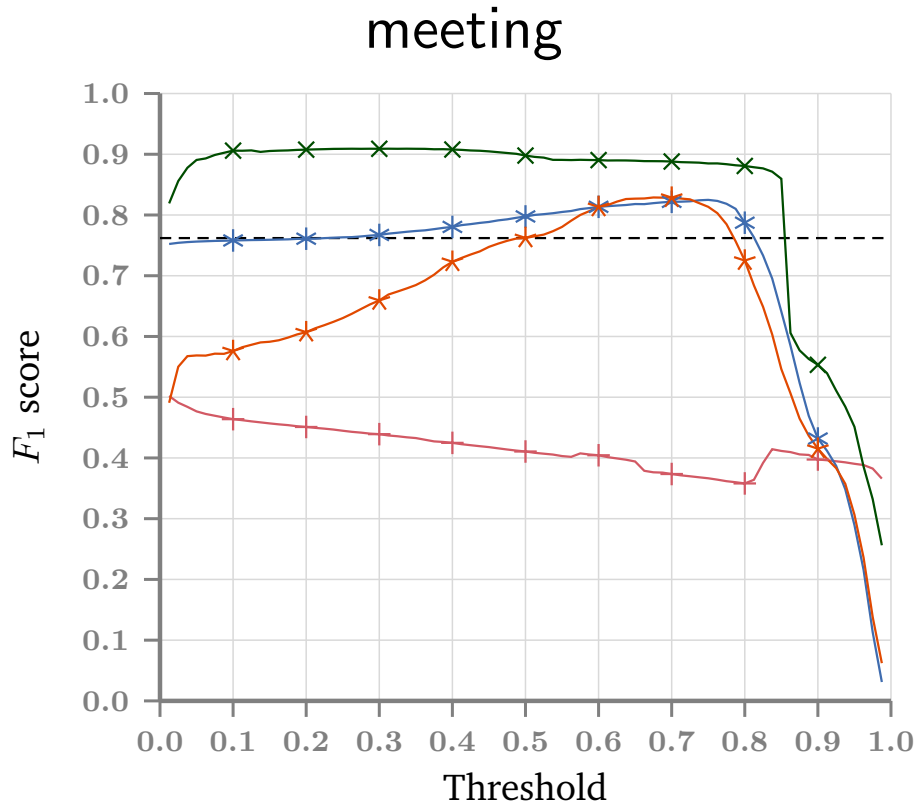
∞ ¬holdsAt(CE, T+1)⇐
¬holdsAt(CE), T) ∧
¬[Initiation Conditions]

0.7 ¬holdsAt(CE, T+1)⇐
[Termination Conditions]

0.6 holdsAt(CE, T+1)⇐
holdsAt(CE, T) ∧
¬[Termination Conditions]

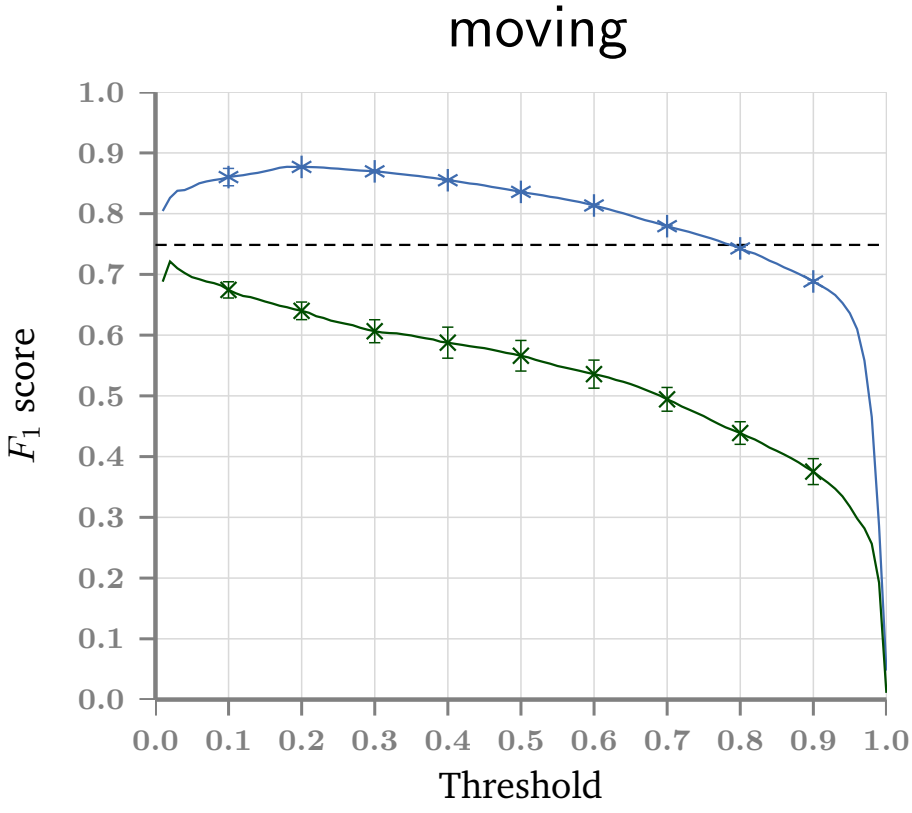
MLN-EC: Complete Data

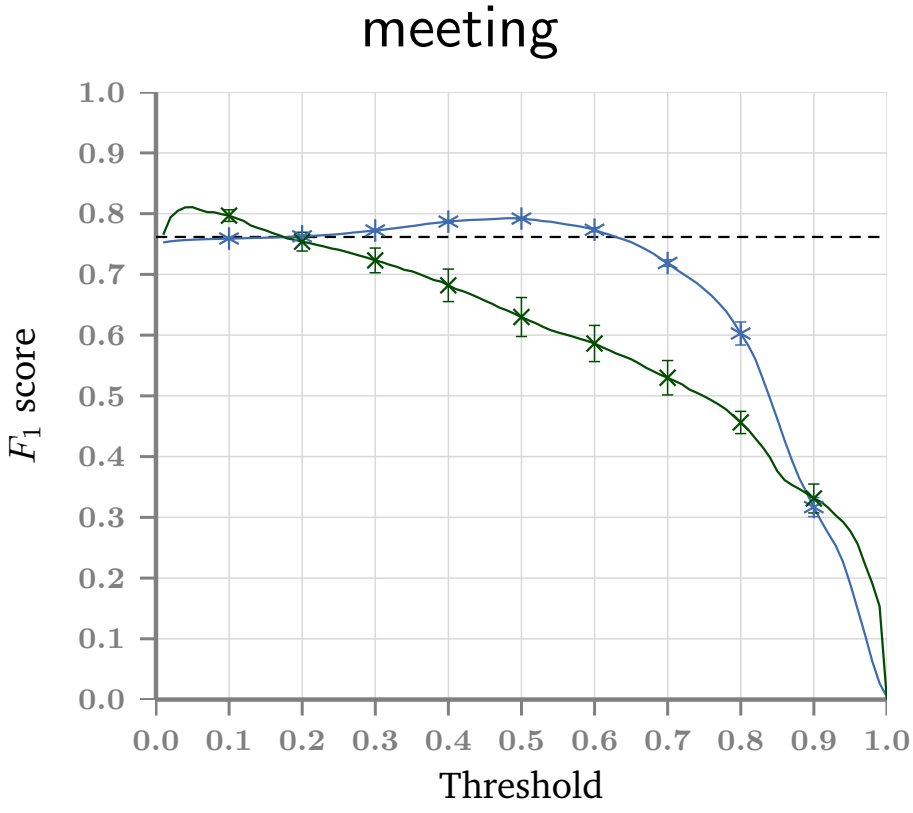




----	$\texttt{EC}_{\texttt{crisp}}$	:	Crisp Event Calculus
—×—	`l−CRF`	:	Linear-chain Conditional Random Field
—+—	$MLN-EC_{HI}$	:	Hard-constrained inertia
—✳—	$MLN-EC_{SIT}$	:	Soft-constrained termination inertial rules
—✳—	$MLN-EC_{SI}$	:	Soft-constrained inertial rules

MLN-EC: Incomplete Data





----	$\texttt{EC}_{\texttt{crisp}}$	:	Crisp Event Calculus
—×—	$\texttt{l}-\texttt{CRF}$	:	Linear-chain Conditional Random Field
—✳—	$MLN-EC_{SIT}$	:	Soft-constrained termination inertial rules

MLN-EC: Summary

- We can deal with both:
  - Uncertainty in the CE definitions, and
  - uncertainty in the input data streams.
- Customisable inertia behaviour to meet the requirements of different applications.

MLN-EC: Summary

- We can deal with both:
  - Uncertainty in the CE definitions, and
  - uncertainty in the input data streams.
- Customisable inertia behaviour to meet the requirements of different applications.

But:

- There is room for improvement with respect to efficiency.

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 CE definition development.

Machine Learning for Event Recognition

Manual development of CE definitions:

- Time consuming.
- Error-prone.

Machine Learning for Event Recognition

Manual development of CE definitions:

- Time consuming.
- Error-prone.

Automated construction for CE definitions:

- Learn complex definitions
  - Structure learning.

Machine Learning for Event Recognition

Manual development of CE definitions:

- Time consuming.
- Error-prone.

Automated construction for CE definitions:

- Learn complex definitions
  - Structure learning.
- Learn from noisy data
  - Weight learning.

Machine Learning for Event Recognition

Manual development of CE definitions:

- Time consuming.
- Error-prone.

Automated construction for CE definitions:

- Learn complex definitions
  - Structure learning.
- Learn from noisy data
  - Weight learning.
- Learn with incomplete or missing annotation
  - Semi-supervised, unsupervised learning.

Machine Learning for Event Recognition

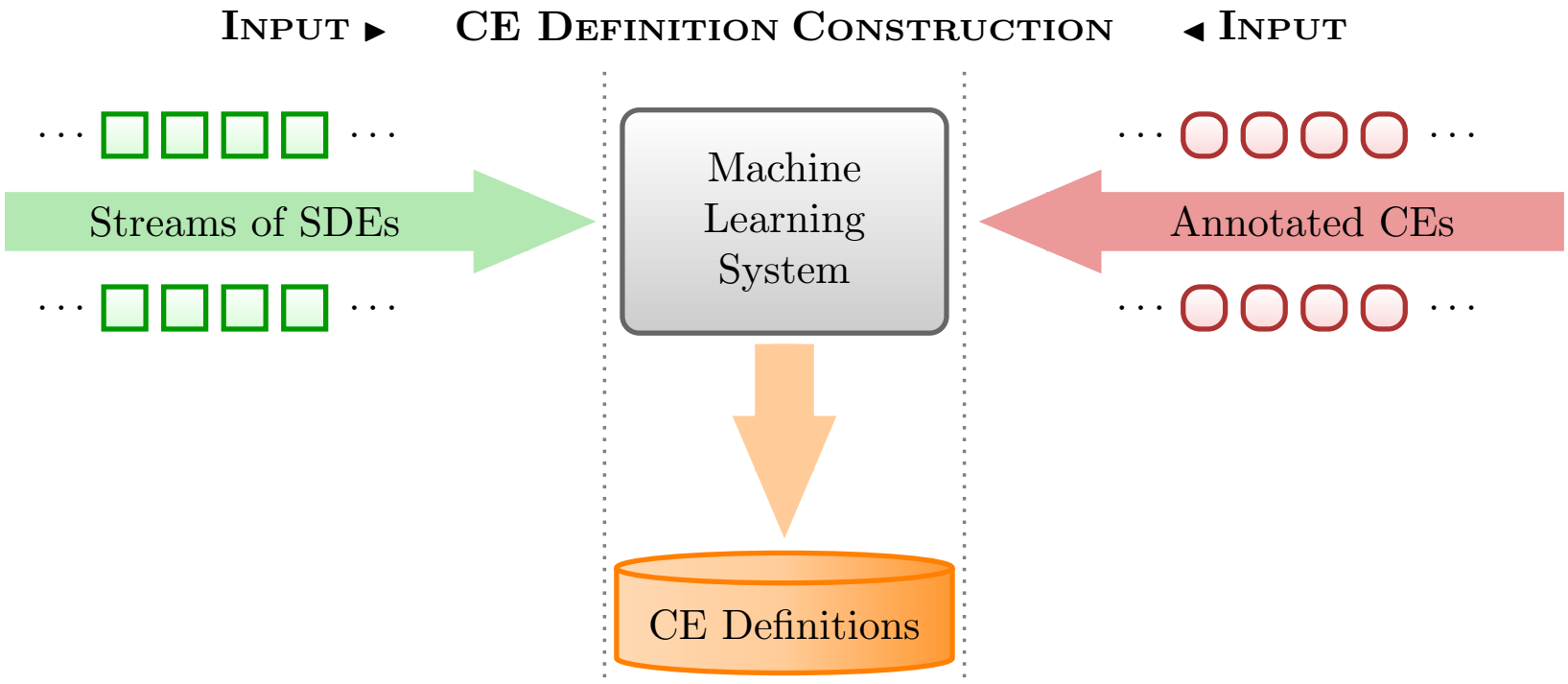
Manual development of CE definitions:

- Time consuming.
- Error-prone.

Automated construction for CE definitions:

- Learn complex definitions
  - Structure learning.
- Learn from noisy data
  - Weight learning.
- Learn with incomplete or missing annotation
  - Semi-supervised, unsupervised learning.
- Learn from large amounts of (streaming) data
  - Incremental learning, online learning.

Machine Learning for Event Recognition



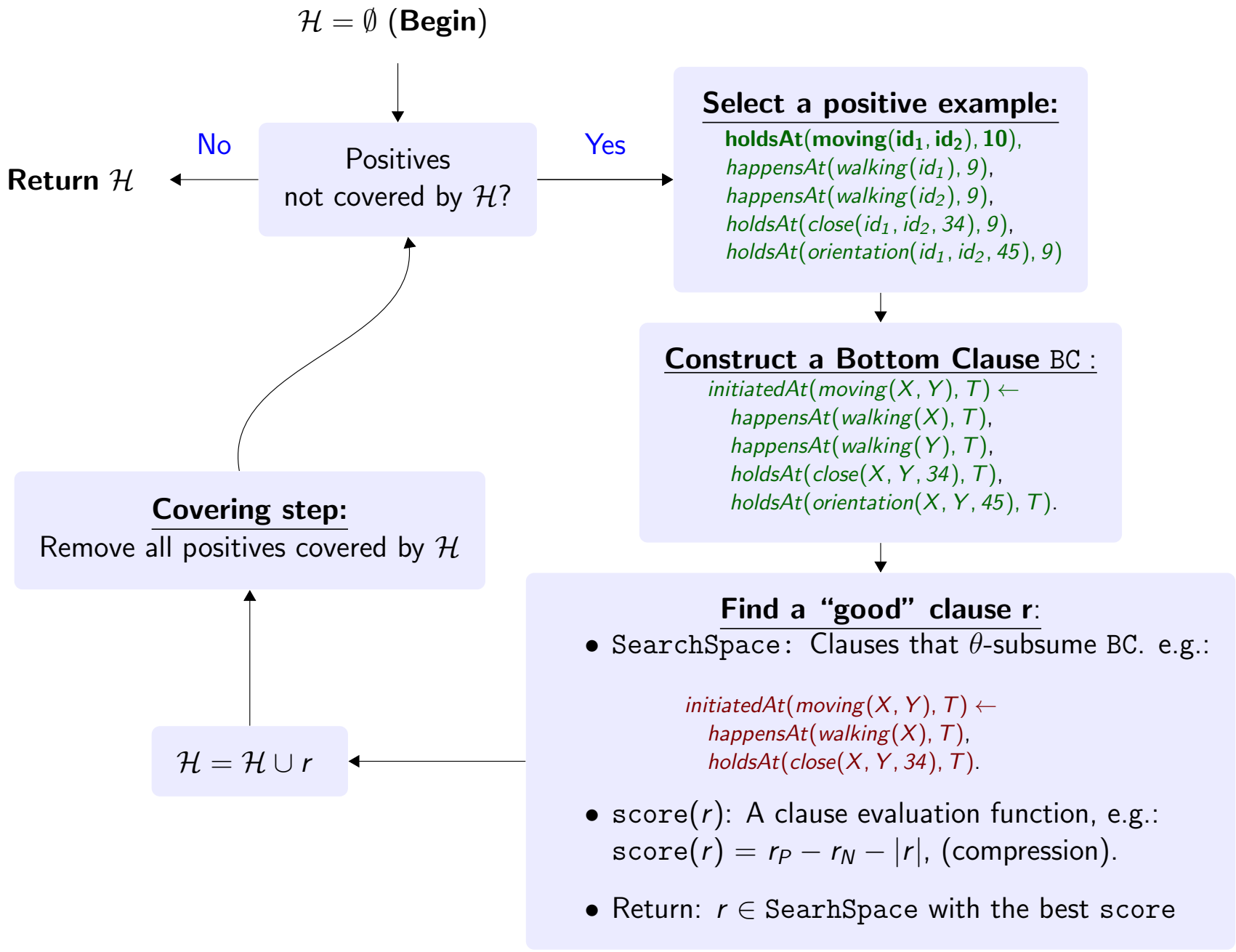
Structure Learning with ILP

Inductive Logic Programming (ILP):

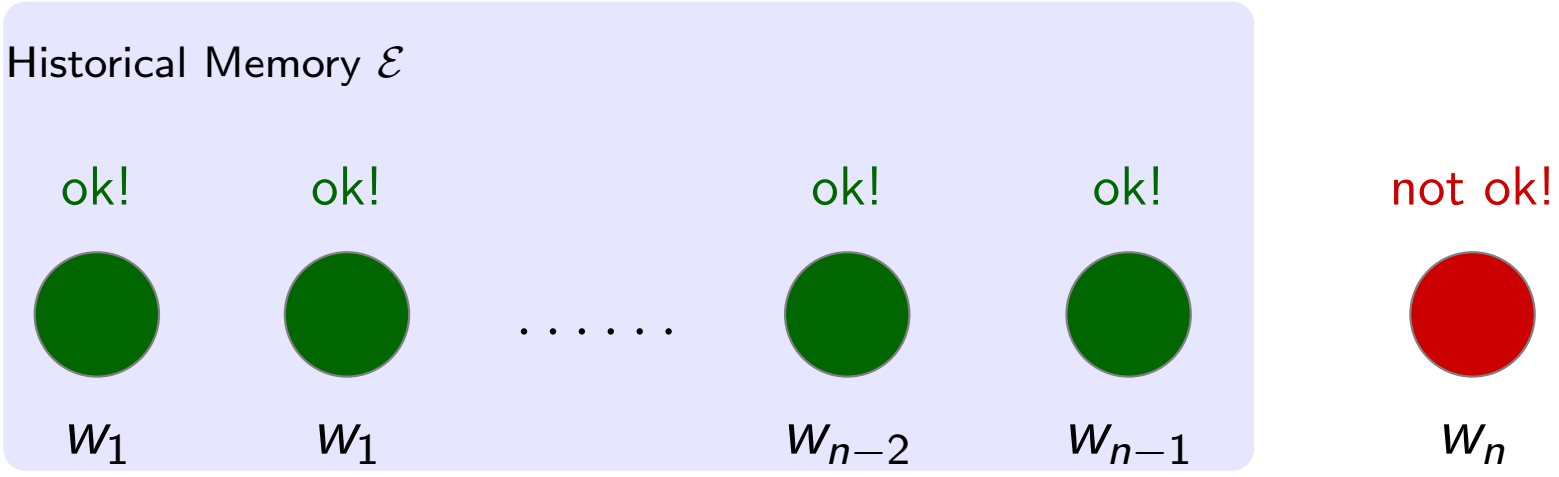
- Input: SDE streams annotated with CE
  - Positive (E^+) and negative (E^-) examples.
- Event recognition engine
  - Background knowledge B.
- Syntax of event recognition language
  - Language bias M.
- Output: A set of CE definitions
  - A hypothesis $\mathcal{H}$ in the language of M, such that:

$$\underset{e^+ \in E^+}{\text{maximize}}\ B \cup \mathcal{H} \models e^+ \ \textbf{and}\ \underset{e^- \in E^-}{\text{minimize}}\ B \cup \mathcal{H} \models e^-$$

A Generic ILP Algorithm



Online Learning



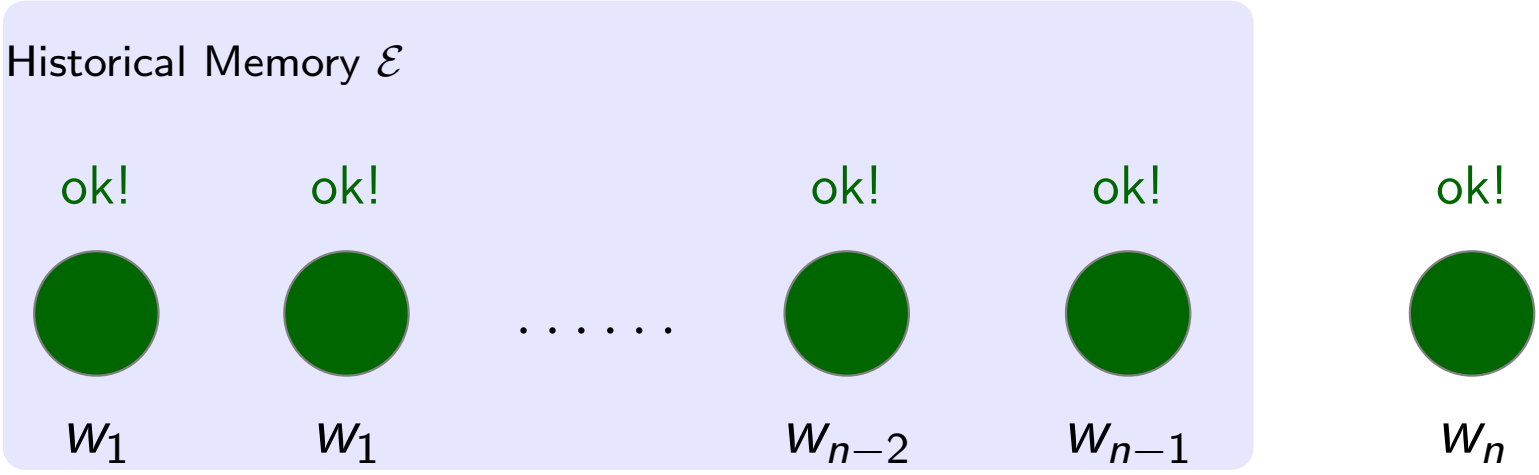
H :

$$initiatedAt(moving(X, Y), T) \leftarrow$$
$$happensAt(walking(X), T),$$
$$happensAt(walking(Y), T),$$
$$holdsAt(close(X, Y, 34), T).$$

Online Learning

Goal:

- Revise H to an H' that accounts for (all) the examples.

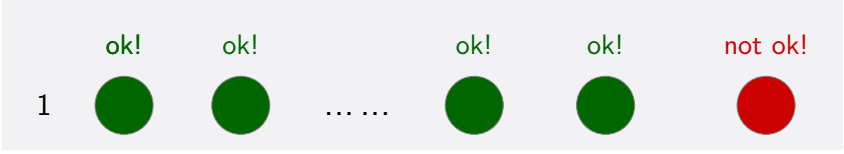


H' :

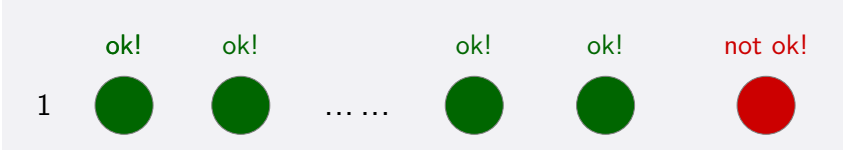
$$initiatedAt(moving(X, Y), T) \leftarrow$$
$$\quad happensAt(walking(X), T),$$
$$\quad happensAt(walking(Y), T),$$
$$\quad holdsAt(close(X, Y, 34), T),$$
$$\quad holdsAt(orientation(X, Y, 45), T).$$

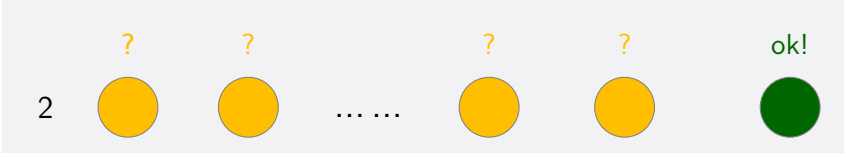
$$terminatedAt(moving(X, Y), T) \leftarrow$$
$$\quad happensAt(inactive(X), T),$$
$$\quad not\ holdsAt(close(X, Y, 34), T).$$

Online Learning



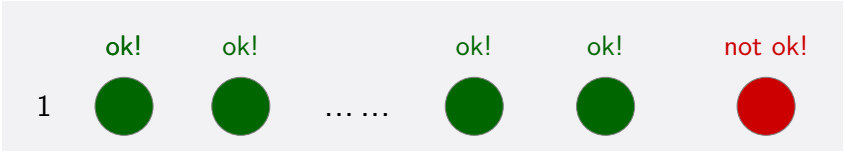
Online Learning

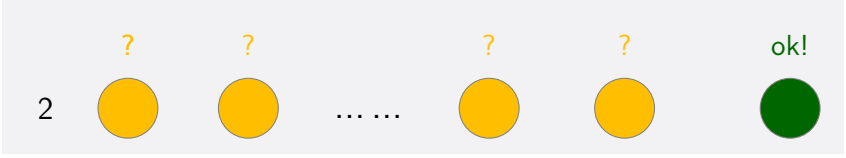


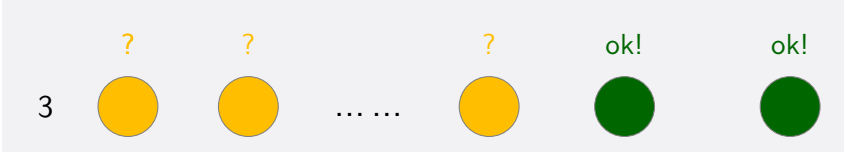


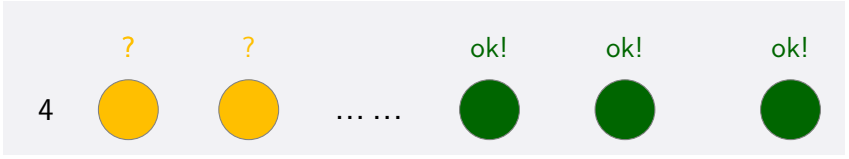
Online Learning

Online Learning

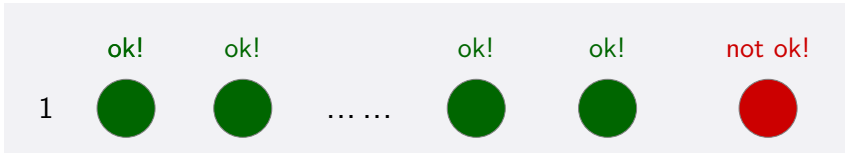


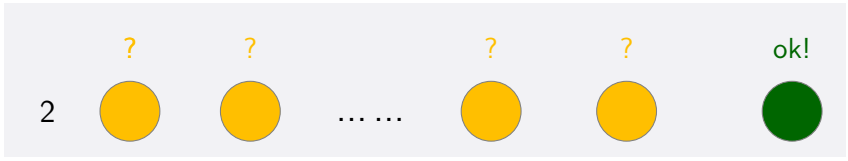


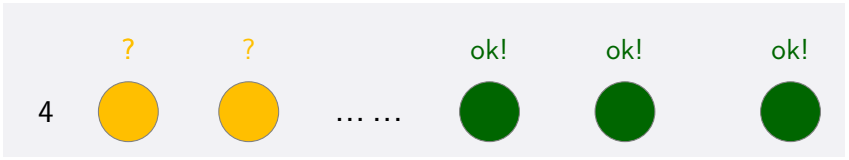


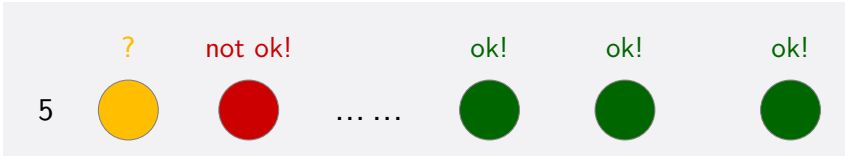


Online Learning

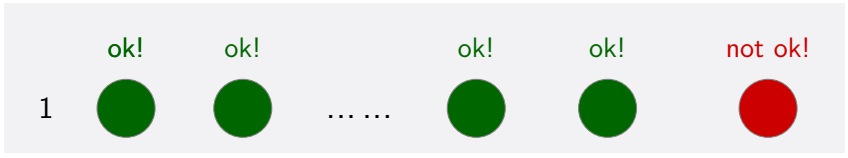


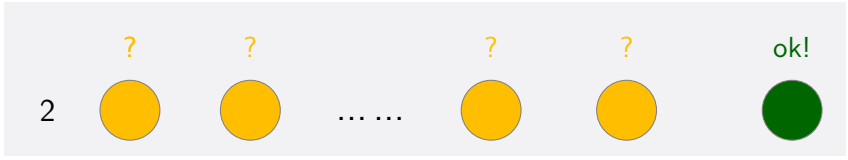


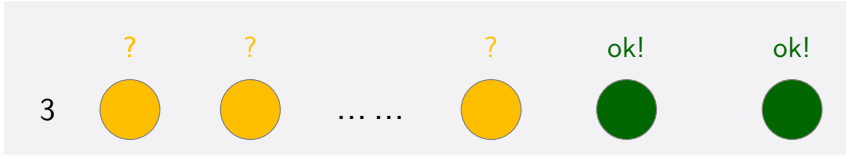


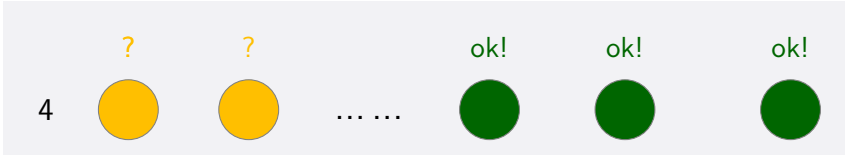


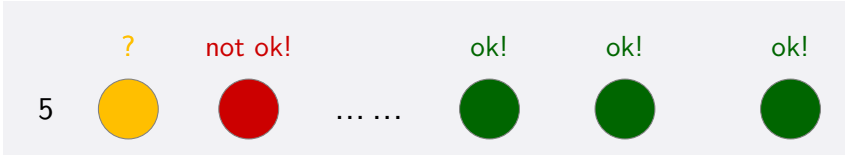
Online Learning

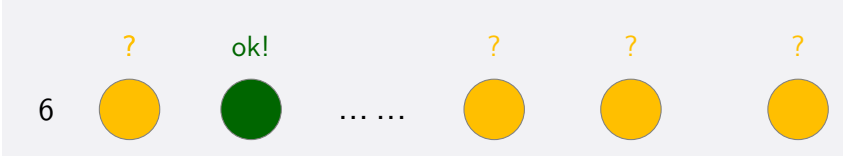




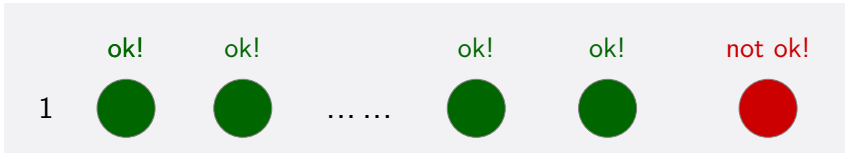


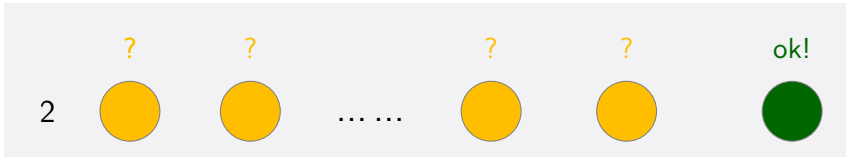


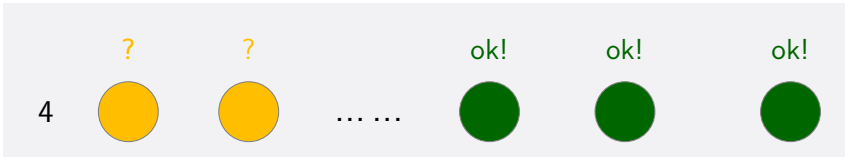


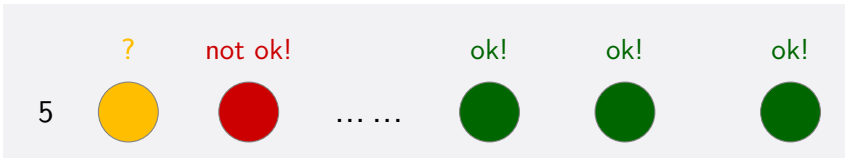


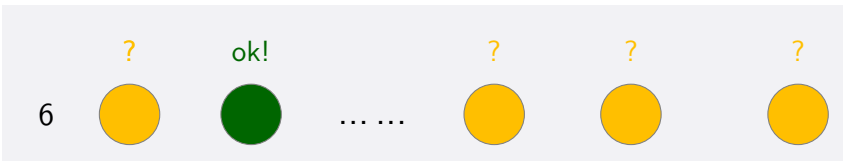
Online Learning





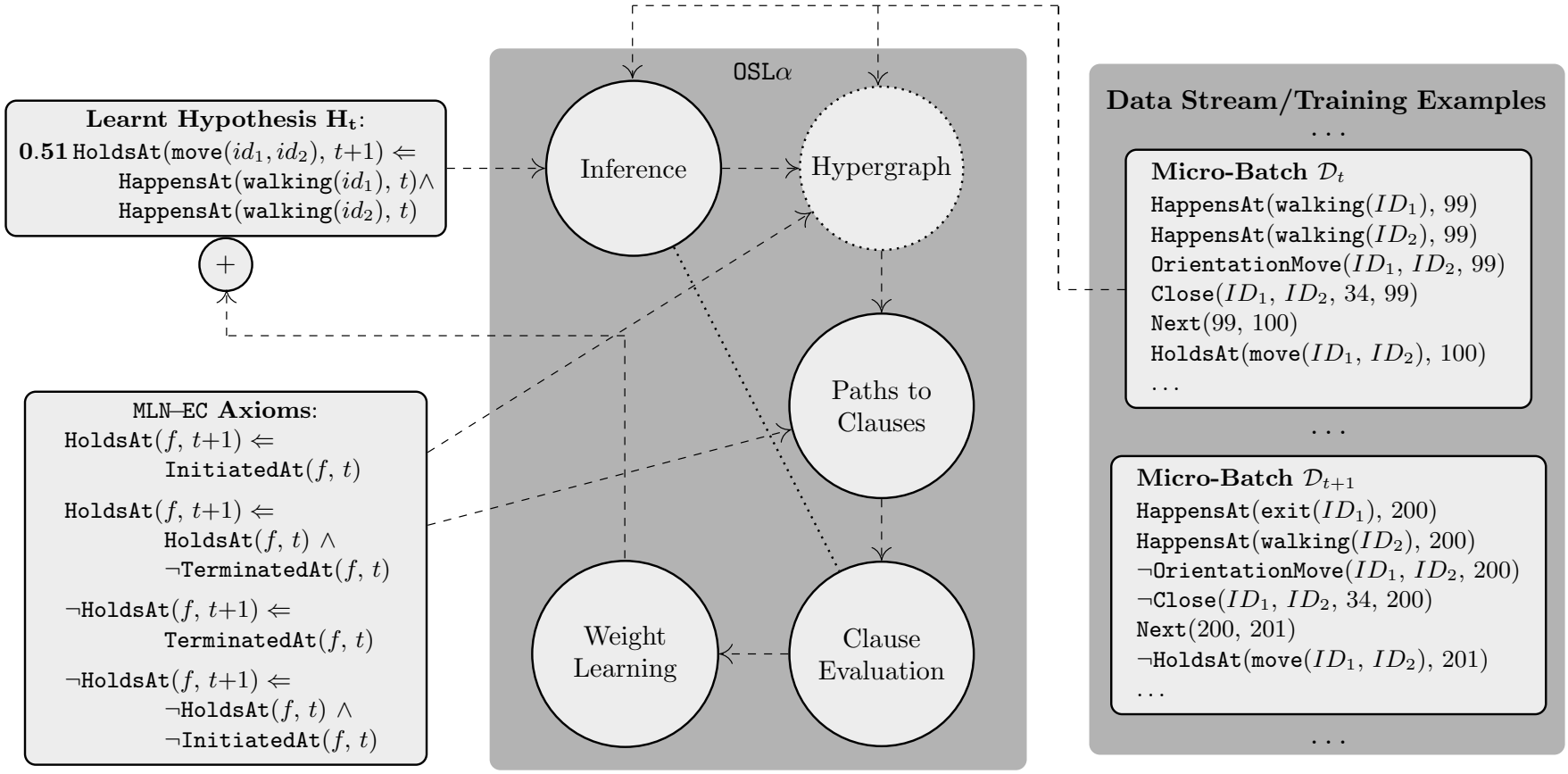






We must start all over again

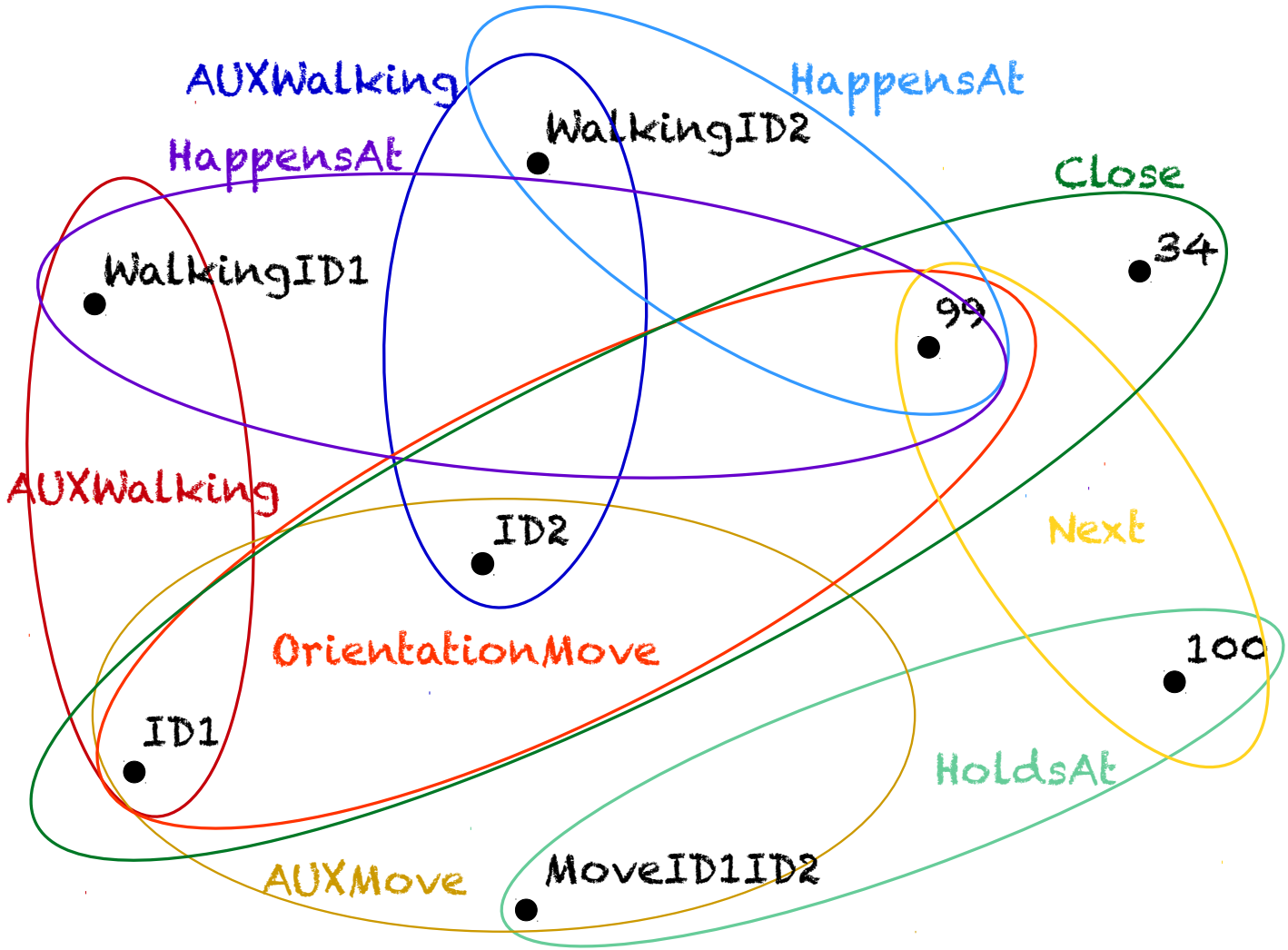
Markov Logic Networks: Online Structure Learning using background knowledge Axiomatization (OSLα)



Training Example (Micro-Batch)

Simple Derived Events	Complex Event Annotation
...	...
HappensAt(walking(ID_1), 99)	
HappensAt(walking(ID_2), 99)	
OrientationMove(ID_1, ID_2, 99)	HoldsAt(move(ID_1, ID_2), 100)
Close(ID_1, ID_2, 34, 99)	
Next(99, 100)	
...	...

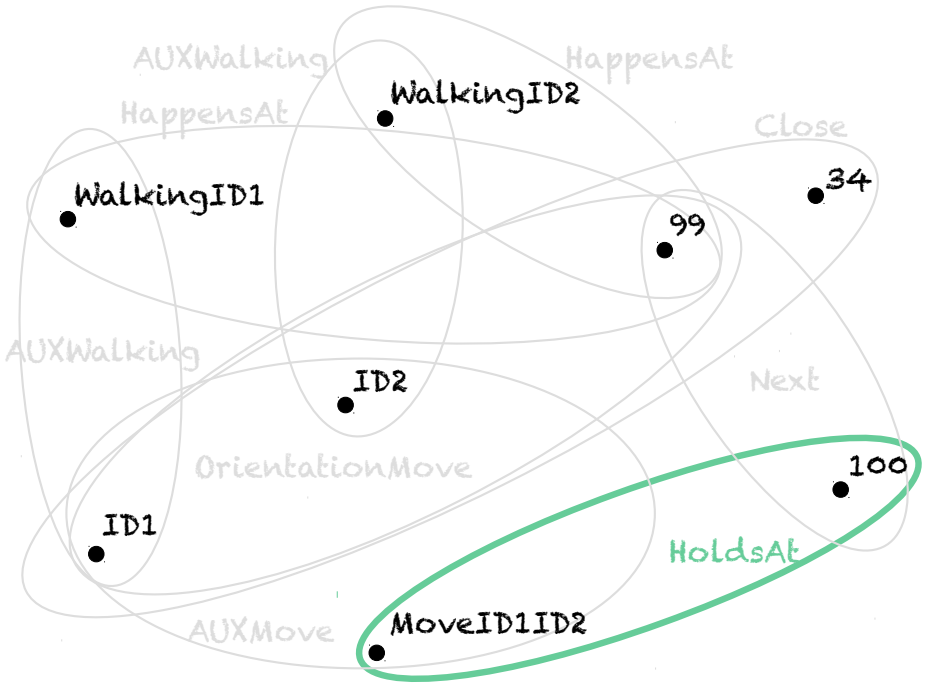
Hypergraph


AUXWalking
HappensAt
WalkingID2
HappensAt
Close
WalkingID1
34
99
AUXWalking
ID2
Next
OrientationMove
100
ID1
HoldsAt
AUXMove
MoveID1ID2

Hypergraph and Relational Pathfinding

$$\mathbf{y}_t^P = \Big(\neg \texttt{HoldsAt}(\texttt{MoveID}_1\texttt{ID}_2,\ 100)\Big)$$

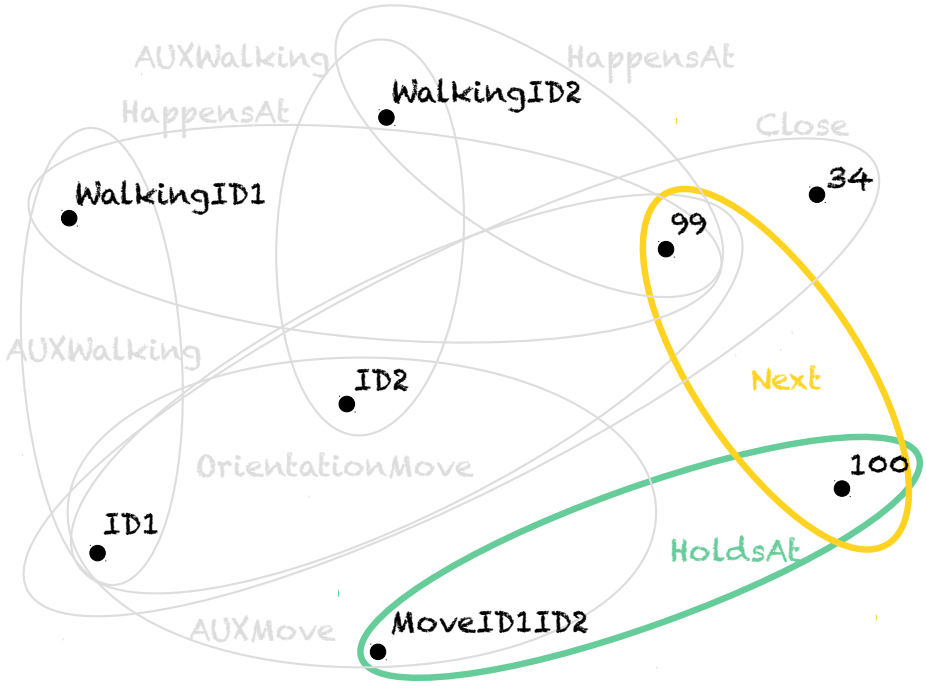
$$\{\texttt{HoldsAt}(\texttt{MoveID}_1\texttt{ID}_2,\ 100),$$



Relational Pathfinding

$$\mathbf{y}_t^P = \Big(\neg \texttt{HoldsAt}(\texttt{MoveID}_1\texttt{ID}_2,\ 100)\Big)$$

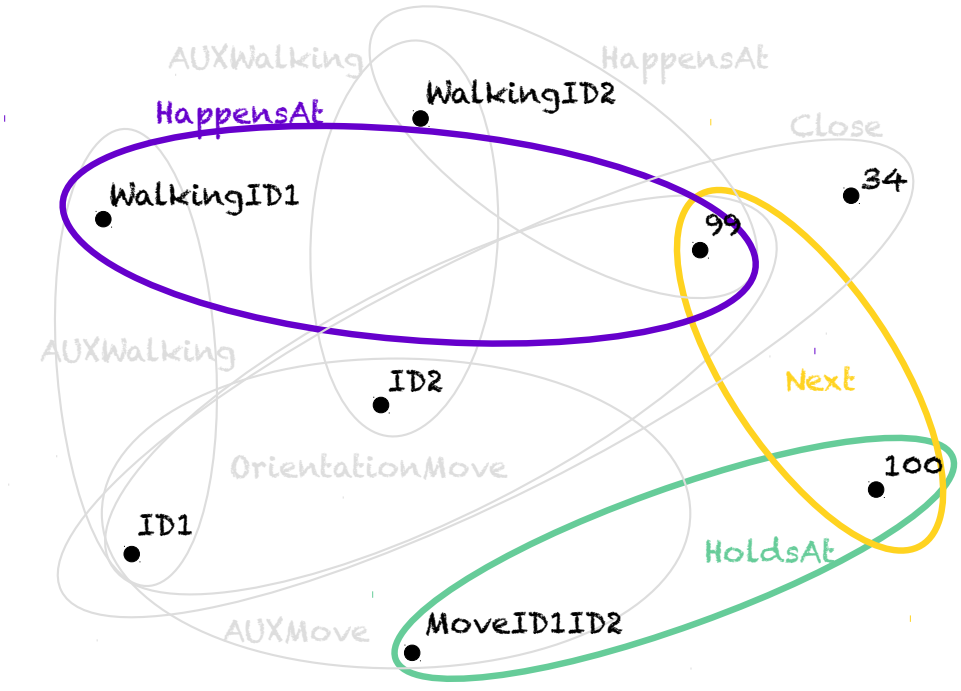
$$\{\texttt{HoldsAt}(\texttt{MoveID}_1\texttt{ID}_2,\ 100),\ \texttt{Next}(99,\ 100),$$



Relational Pathfinding

$$\mathbf{y}_t^P = \Big(\neg \texttt{HoldsAt}(\texttt{MoveID}_1\texttt{ID}_2,\ 100)\Big)$$

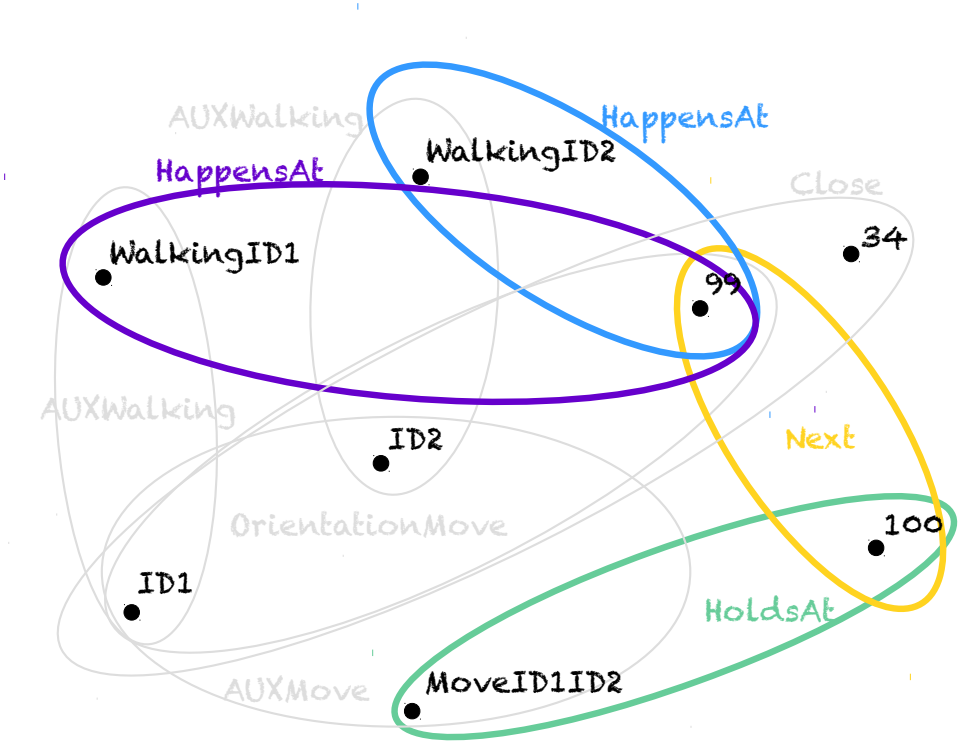
$\{\texttt{HoldsAt}(\texttt{MoveID}_1\texttt{ID}_2,\ 100),\ \texttt{Next}(99,\ 100),$
$\texttt{HappensAt}(\texttt{WalkingID}_1,\ 99),$



Relational Pathfinding

$$\mathbf{y}_t^P = \Big(\neg \texttt{HoldsAt}(\texttt{MoveID}_1\texttt{ID}_2,\ 100)\Big)$$

$\{\texttt{HoldsAt}(\texttt{MoveID}_1\texttt{ID}_2,\ 100),\ \texttt{Next}(99,\ 100),$
$\texttt{HappensAt}(\texttt{WalkingID}_1,\ 99),$
$\texttt{HappensAt}(\texttt{WalkingID}_2,\ 99)\}$



Template-Guided Search

Wrongly predicted atom:
$\neg \texttt{HoldsAt}(\texttt{MoveID}_1\texttt{ID}_2, 100)$

$\texttt{HoldsAt}(f, t_1) \Leftarrow$
$\texttt{InitiatedAt}(f, t_0) \wedge$
$\texttt{Next}(t_0, t_1)$

Template-Guided Search

Wrongly predicted atom:
$\neg \texttt{HoldsAt}(\texttt{MoveID}_1\texttt{ID}_2, 100)$

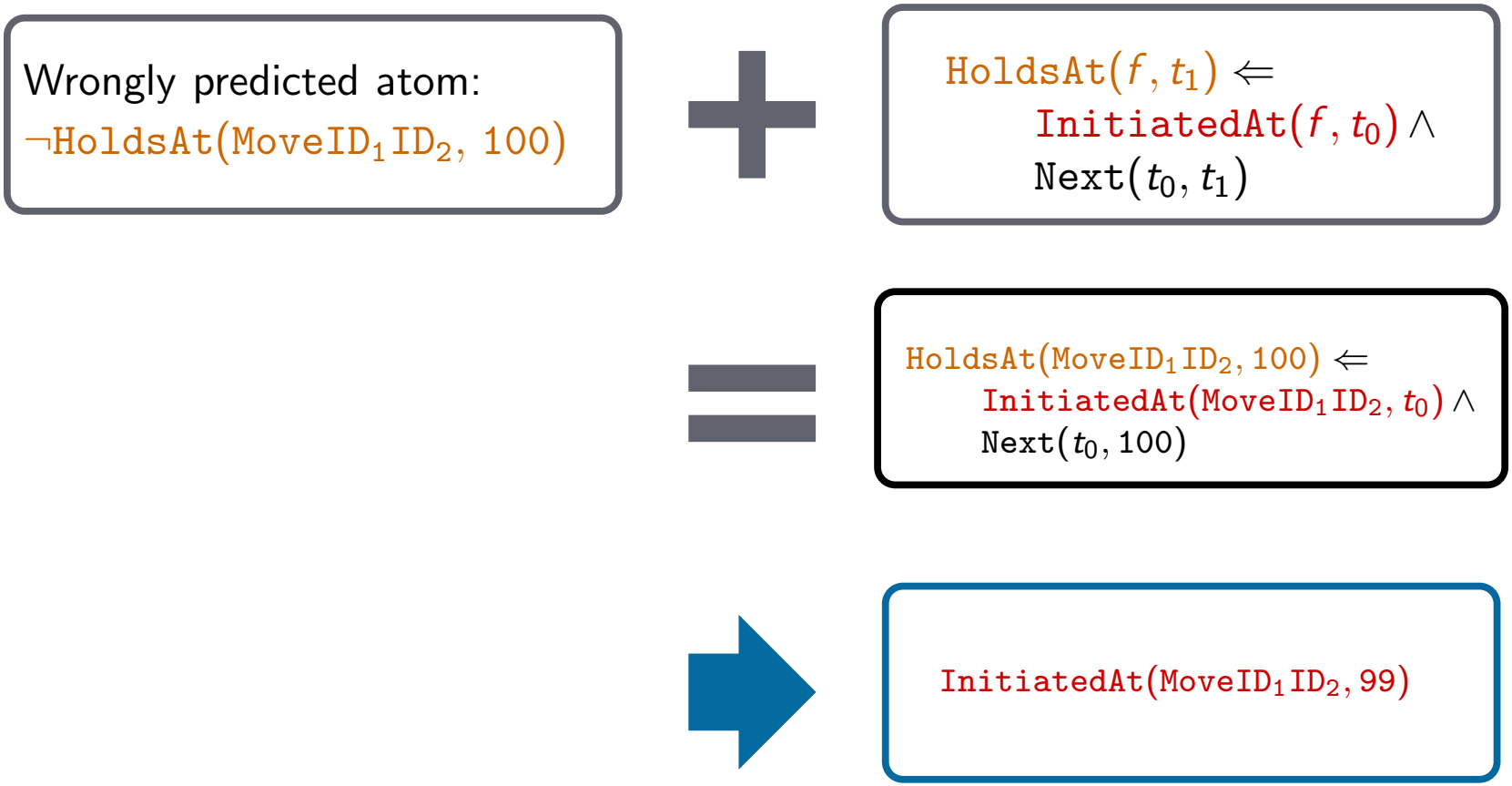
+

$\texttt{HoldsAt}(f, t_1) \Leftarrow$
$\texttt{InitiatedAt}(f, t_0) \wedge$
$\texttt{Next}(t_0, t_1)$

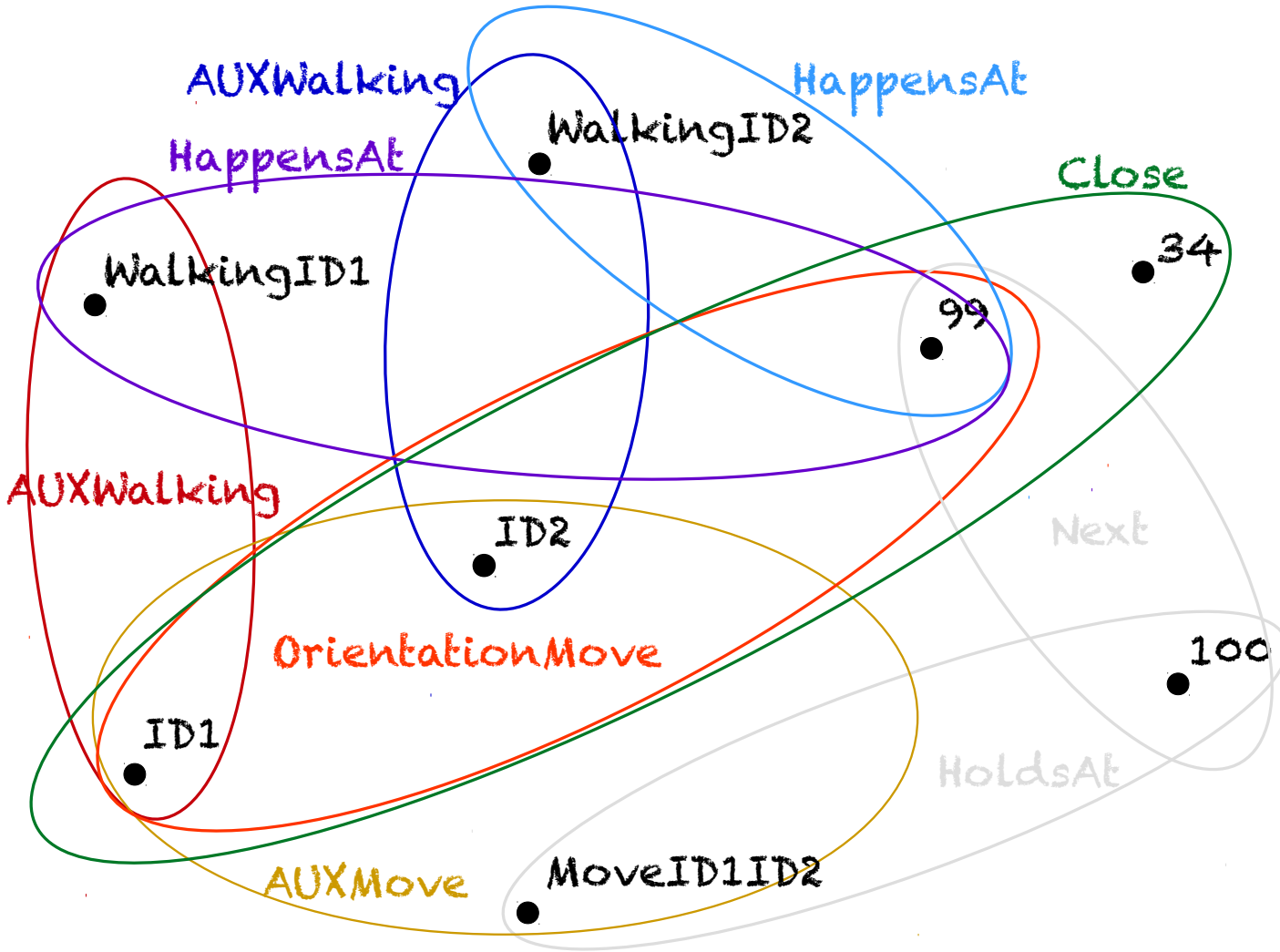
=

$\texttt{HoldsAt}(\texttt{MoveID}_1\texttt{ID}_2, 100) \Leftarrow$
$\texttt{InitiatedAt}(\texttt{MoveID}_1\texttt{ID}_2, t_0) \wedge$
$\texttt{Next}(t_0, 100)$

Template-Guided Search



Reduced Hypergraph



Clause Creation, Clause Evaluation and Weight Learning

Clause creation:

- Generalize each path into a definite clause

$$\texttt{InitiatedAt}(\texttt{move}(id_1, id_2), t) \Leftarrow \\ \texttt{HappensAt}(\texttt{walking}(id_1), t) \wedge \\ \texttt{HappensAt}(\texttt{walking}(id_2), t)$$

Clause Creation, Clause Evaluation and Weight Learning

Clause creation:

- Generalize each path into a definite clause

$$\texttt{InitiatedAt}(\texttt{move}(id_1, id_2), t) \Leftarrow \\ \texttt{HappensAt}(\texttt{walking}(id_1), t) \wedge \\ \texttt{HappensAt}(\texttt{walking}(id_2), t)$$

Clause evaluation:

- Keep clauses whose coverage of the annotation is significantly greater than that of the clauses already learnt.

Clause Creation, Clause Evaluation and Weight Learning

Clause creation:

- Generalize each path into a definite clause

$$\texttt{InitiatedAt}(\texttt{move}(id_1, id_2), t) \Leftarrow \\ \texttt{HappensAt}(\texttt{walking}(id_1), t) \wedge \\ \texttt{HappensAt}(\texttt{walking}(id_2), t)$$

Clause evaluation:

- Keep clauses whose coverage of the annotation is significantly greater than that of the clauses already learnt.

Weight learning:

- Extended clauses inherit initially the weights of their ancestors.
- Optimize the weights of all clauses.

Machine Learning for Event Recognition: Summary

- CE definition construction
  - using large (streaming) data;
  - tolerant to noise.
- Acceptable predictive accuracy.

Machine Learning for Event Recognition: Summary

- CE definition construction
  - using large (streaming) data;
  - tolerant to noise.
- Acceptable predictive accuracy.
- But:
  - Constructed CE definitions tend to overfit the data.
  - There is room for improvement with respect to efficiency.

Open Issues

Issue (1): Real-time Probabilistic Event Recognition

Types of uncertainty:

- Noisy input stream.
- Imprecise CE definitions.

Existing solution:

- Probabilistic graphical models & logic-based approaches.

Issue (1): Real-time Probabilistic Event Recognition

Types of uncertainty:

- Noisy input stream.
- Imprecise CE definitions.

Existing solution:

- Probabilistic graphical models & logic-based approaches.

Major drawback:

- Large overhead that does not allow for real-time performance.

Issue (2): Distributed Recognition

Distribution of CE recognition among multiple nodes:

- Data distribution.
- Distribution of CE (sub-)definitions.

Issue (2): Distributed Recognition

Distribution of CE recognition among multiple nodes:

- Data distribution.
- Distribution of CE (sub-)definitions.

But, virtually no work on distributing probabilistic CE recognition.

Issue (2): Distributed Recognition

Distribution of CE recognition among multiple nodes:

- Data distribution.
- Distribution of CE (sub-)definitions.

But, virtually no work on distributing probabilistic CE recognition.

Interplay with communication efficiency:

- Reduce communication by decomposing recognition in set of local constraints at event sources
  - By sketching.
  - By geometric models.
- Limited to particular types of CE: functions over aggregate values.

Issue (3): Multi-scale Temporal Aggregation of Events

CE evolve over multiple scales of time and space:

- SDE streams.
- Context information, e.g. historic data.

Issue (3): Multi-scale Temporal Aggregation of Events

CE evolve over multiple scales of time and space:

- SDE streams.
- Context information, e.g. historic data.

The recognition system should be adaptable, computing dynamically the appropriate scales

- Appropriate lengths of multi-granular windows.
- Appropriate slice of context information.

Issue (3): Multi-scale Temporal Aggregation of Events

CE evolve over multiple scales of time and space:

- SDE streams.
- Context information, e.g. historic data.

The recognition system should be adaptable, computing dynamically the appropriate scales

- Appropriate lengths of multi-granular windows.
- Appropriate slice of context information.

Semantics and processing guarantees, despite adaptation.

Issue (4): Event Forecasting

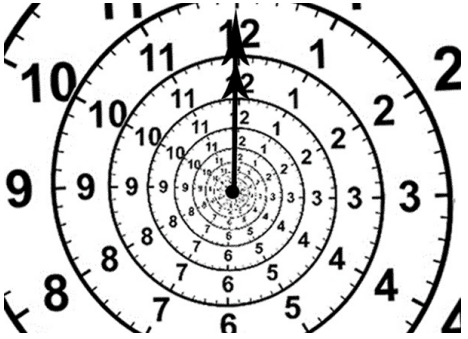
Current work focuses on

- Present (monitoring).
- Past (post mortem analysis).

Issue (4): Event Forecasting

Current work focuses on

- Present (monitoring).
- Past (post mortem analysis).



Stream processing will focus on future

- Predictive analysis.
- Trend detection.

Issue (4): Event Forecasting

Current work focuses on

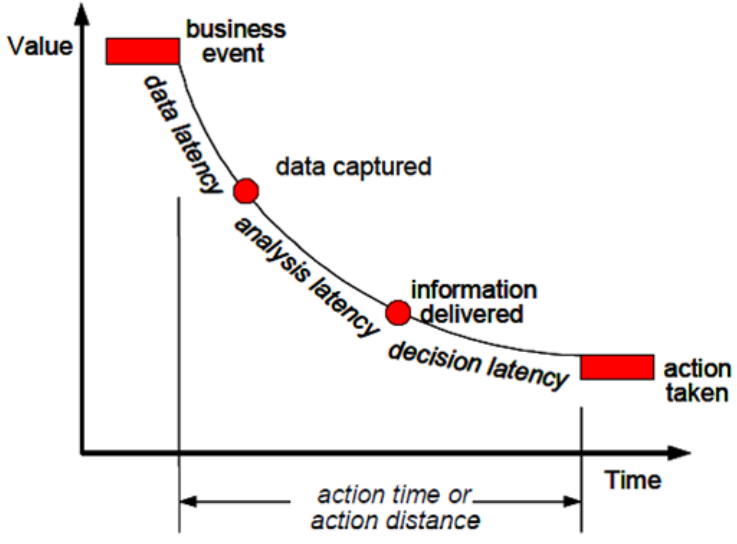
- Present (monitoring).
- Past (post mortem analysis).

Stream processing will focus on future

- Predictive analysis.
- Trend detection.

The earlier the better

- Time to react to an anticipated situation.
- Avoidance of situation or initialisation of counter measures.



Resources

- Tutorial paper: A. Artikis, A. Skarlatidis, F. Portet, G. Paliouras: Logic-based event recognition. Knowledge Engineering Review 27(4): 469-506 (2012)
- Slides, complex event recognition software, datasets: `http://cer.iit.demokritos.gr`


Questions?

Acknowledgements

- Marek Sergot, Imperial College London
- Anastasios Skarlatidis, Pollfish
- Matthias Weidlich, Humboldt-Universität zu Berlin