Can Computers Understand what is Happening? A Tutorial on Complex Event Recognition (CER)

Alexander Artikis[1,2]
Periklis Mantenoglou[1,3]

[1]NCSR Demokritos, Athens, Greece
[2]University of Piraeus, Greece
[3]NKUA, Greece

https://cer.iit.demokritos.gr

About the Tutorial

Contributors:

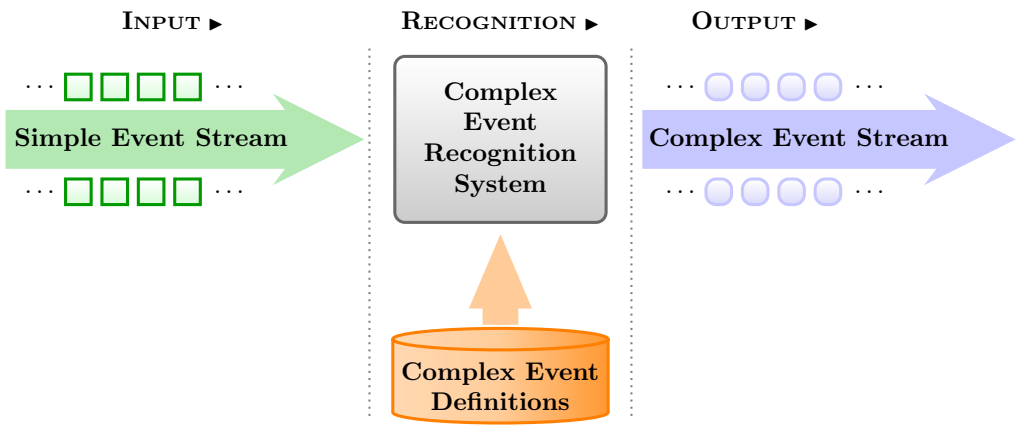
- Elias Alevizos.
- Manolis Pitsikalis.
- Efthimis Tsilionis.

Resources: http://cer.iit.demokritos.gr

- Slides: http://cer.iit.demokritos.gr/talks
- Code: http://cer.iit.demokritos.gr/software
- Data: http://cer.iit.demokritos.gr/datasets
- Opportunities for (funded) collaboration: job openings and topics for BSc/MSc theses and internships

Complex Event Recognition & Applications

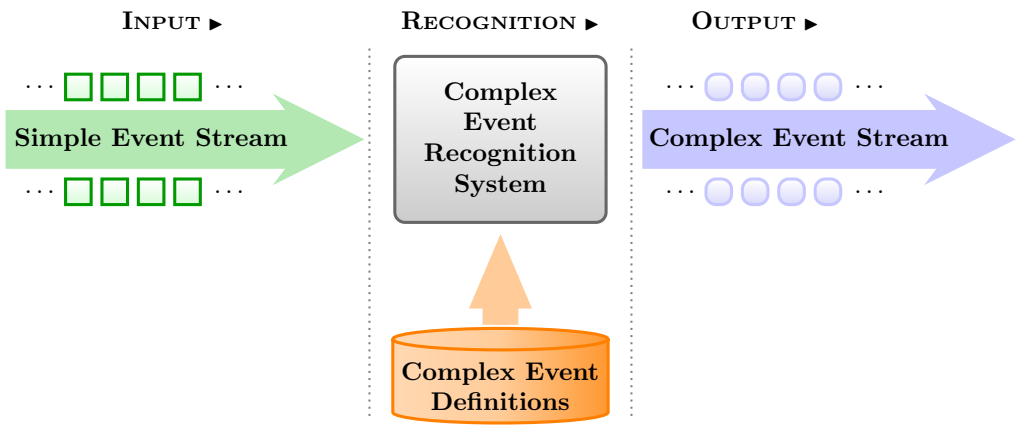
Complex Event Recognition (Event Pattern Matching)*,†

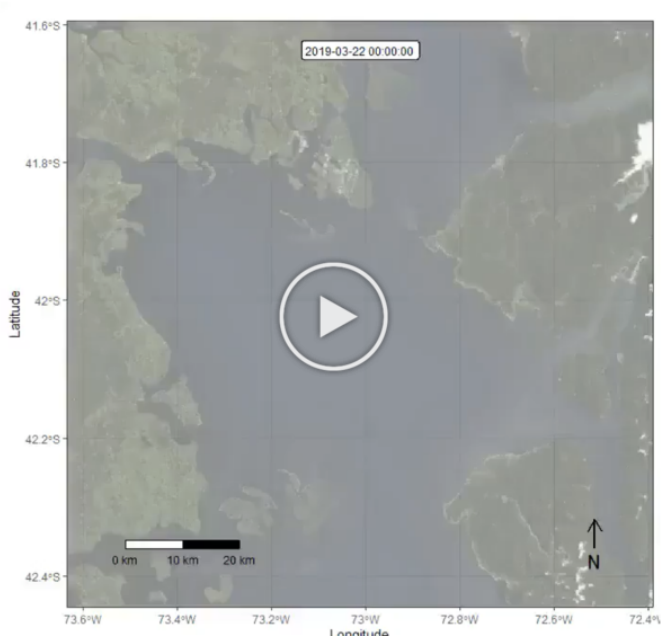


*Giatrakos et al, Complex Event Recognition in the Big Data Era: A Survey. VLDB Journal, 2020.

†Alevizos et al, Probabilistic Complex Event Recognition: A Survey. ACM Computing Surveys, 2017.

Complex Event Recognition (Event Pattern Matching)*,†



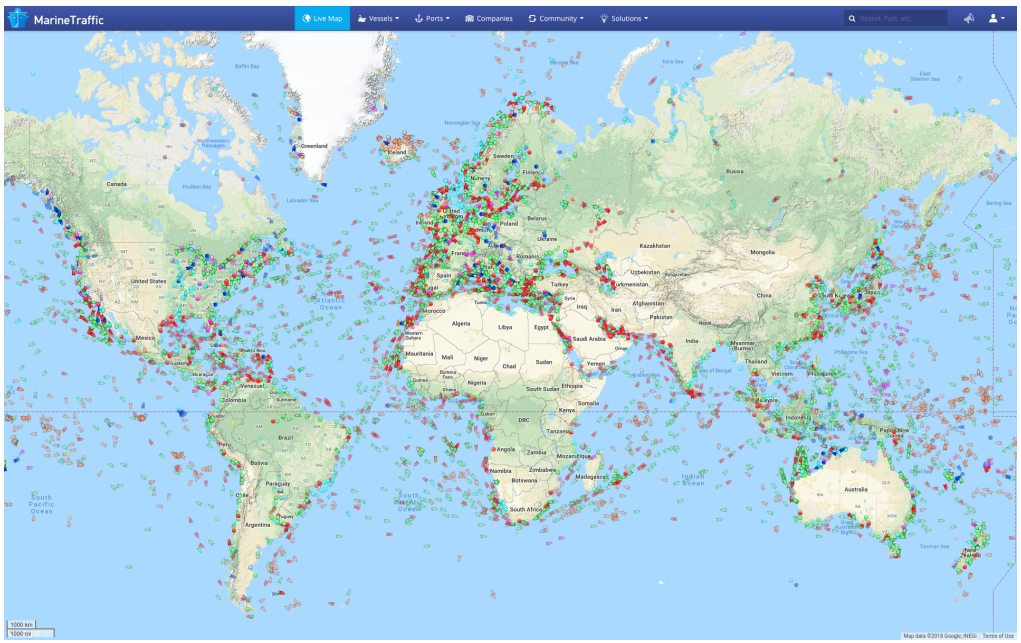


https://rdcu.be/cNkQE

*Giatrakos et al, Complex Event Recognition in the Big Data Era: A Survey. VLDB Journal, 2020.

†Alevizos et al, Probabilistic Complex Event Recognition: A Survey. ACM Computing Surveys, 2017.

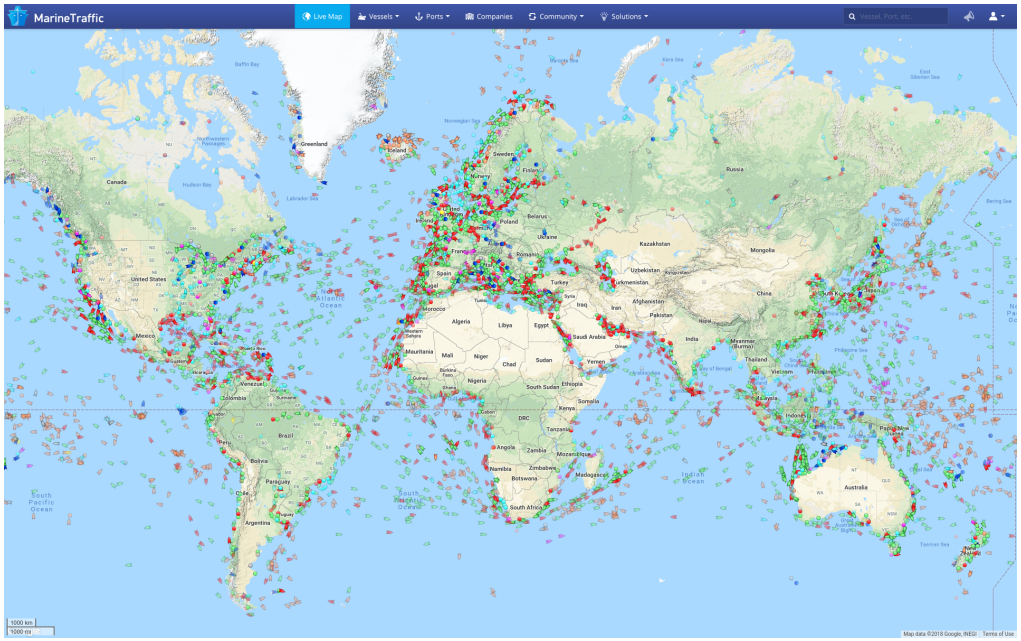
Maritime Situational Awareness*



http://www.marinetraffic.com

*Artikis and Zissis, Guide to Maritime Informatics, Springer, 2021.

Maritime Situational Awareness*



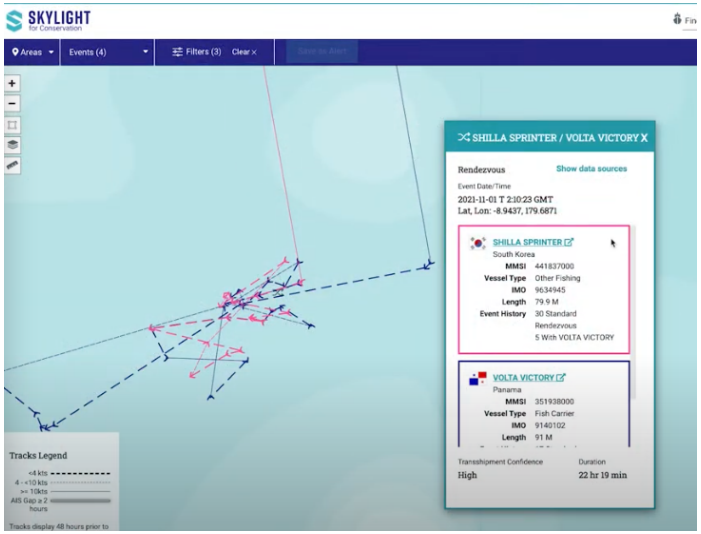
http://www.marinetraffic.com

https://cer.iit.demokritos.gr (fishing vessel)

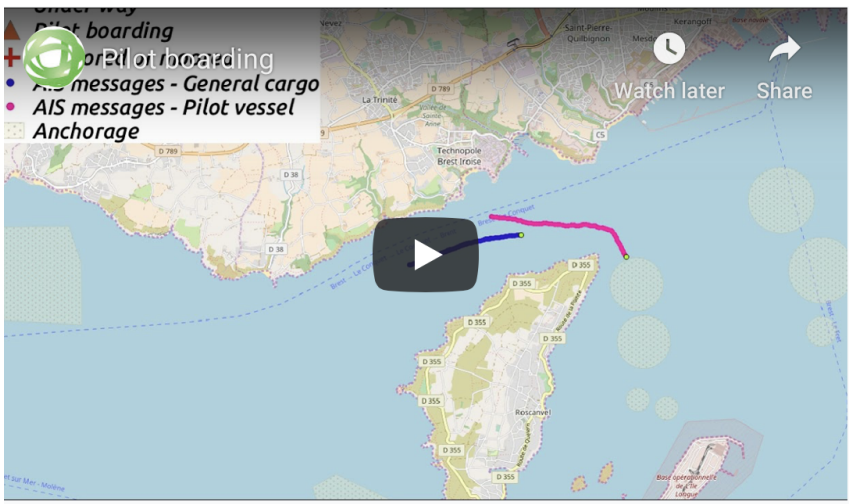
*Artikis and Zissis, Guide to Maritime Informatics, Springer, 2021.

Maritime Situational Awareness*

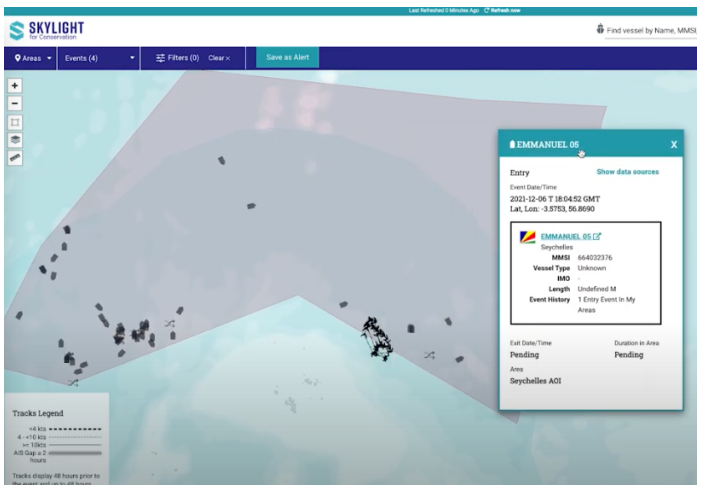
https://cer.iit.demokritos.gr (tugging)



https://www.skylight.global (rendez-vous)



https://cer.iit.demokritos.gr (pilot boarding)



https://www.skylight.global (enter area)

*Artikis and Zissis, Guide to Maritime Informatics, Springer, 2021.

Data Challenges

- Velocity, Volume: Millions of position signals/day at European scale.

Data Challenges

- Velocity, Volume: Millions of position signals/day at European scale.
- Variety: Position signals need to be combined with other data streams
  - Weather forecasts, sea currents, etc.
- ... and static information
  - NATURA areas, shallow waters areas, coastlines, etc.

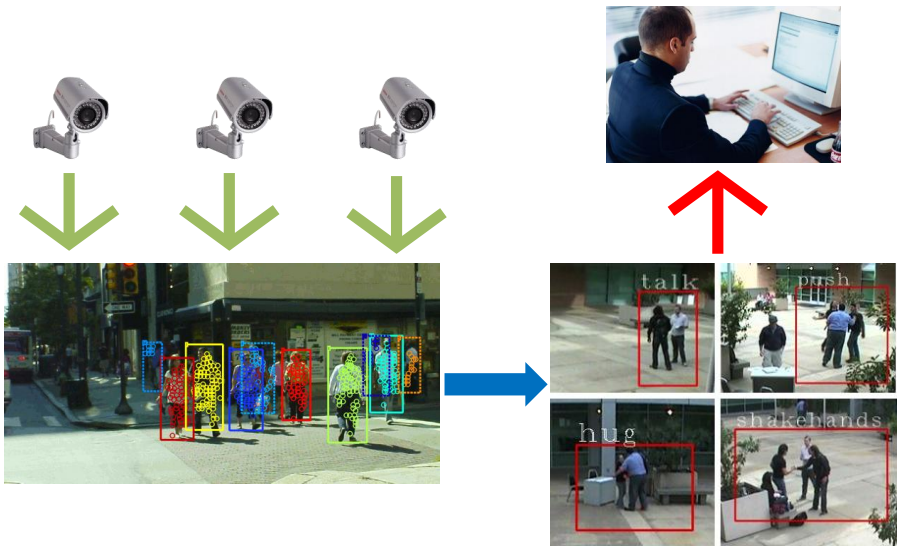
Data Challenges

- Velocity, Volume: Millions of position signals/day at European scale.
- Variety: Position signals need to be combined with other data streams
  - Weather forecasts, sea currents, etc.
- ... and static information
  - NATURA areas, shallow waters areas, coastlines, etc.
- Lack of Veracity: GPS manipulation, vessels reporting false identity, communication gaps.

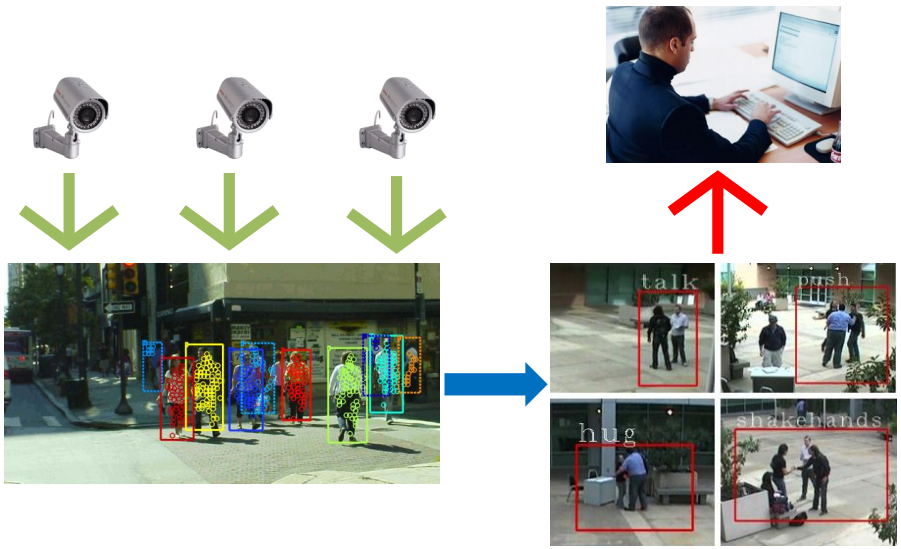
Data Challenges

- Velocity, Volume: Millions of position signals/day at European scale.
- Variety: Position signals need to be combined with other data streams
  - Weather forecasts, sea currents, etc.
- ... and static information
  - NATURA areas, shallow waters areas, coastlines, etc.
- Lack of Veracity: GPS manipulation, vessels reporting false identity, communication gaps.
- Distribution: Vessels operating across the globe.

Human Activity Recognition



Human Activity Recognition



https://cer.iit.demokritos.gr (activity recognition)

Human Activity Recognition

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

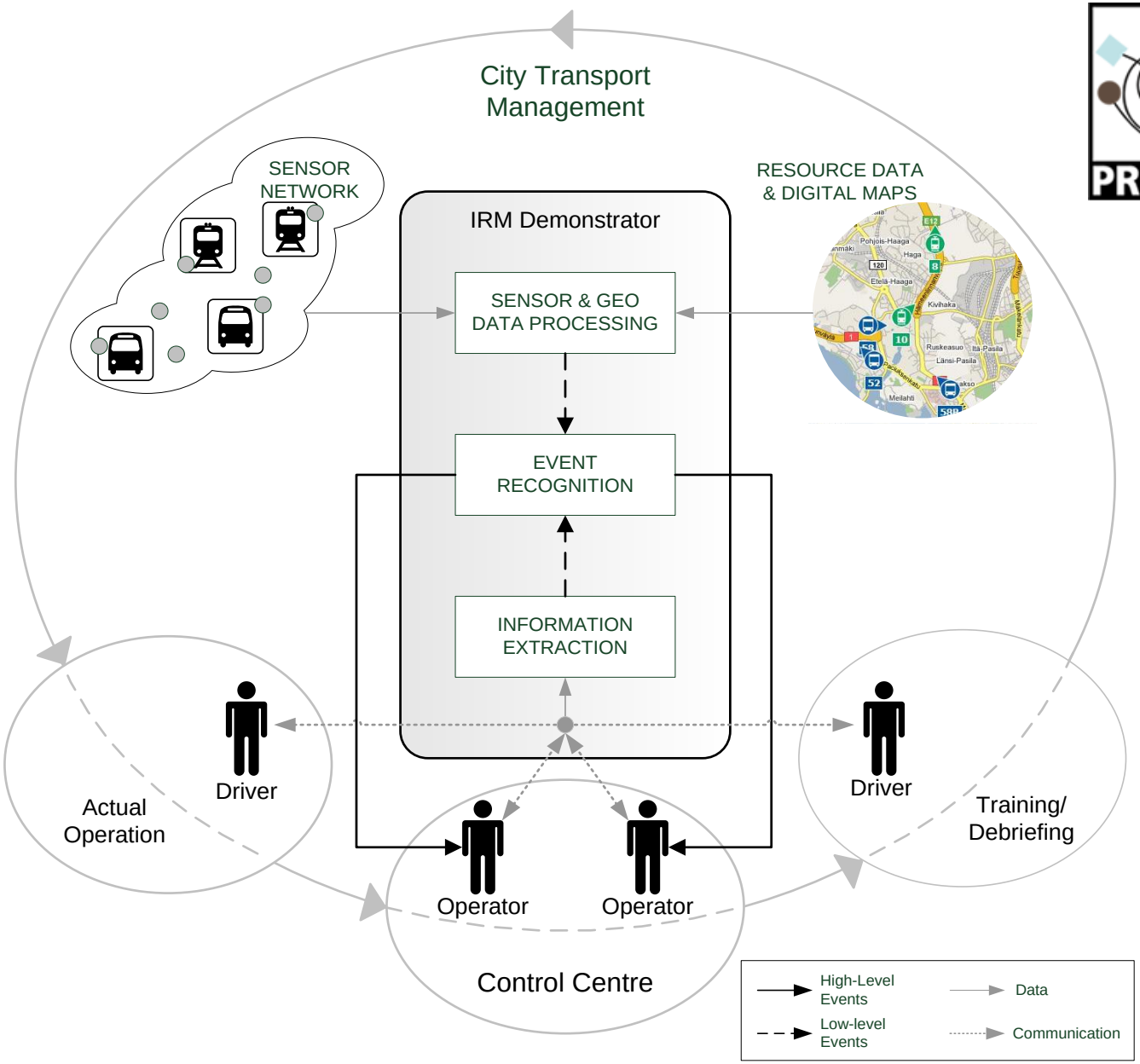
Human Activity Recognition

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

Human Activity Recognition

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



Credit Card Fraud Recognition

SPEEDD

Simple events:

- Credit card transactions from all over the world.

Complex events:

- 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

- Fraud must be detected within a matter of milliseconds.

Credit Card Fraud Recognition

SPEEDD

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

Credit Card Fraud Recognition

SPEEDD

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

Requirements

- Expressive representation
  - to capture complex relationships between the events that stream into the system.

Requirements

- Expressive representation
  - to capture complex relationships between the events that stream into the system.
- Efficient reasoning
  - to support real-time decision-making in large-scale, (geographically) distributed applications.

Requirements

- Expressive representation
  - to capture complex relationships between the events that stream into the system.
- Efficient reasoning
  - to support real-time decision-making in large-scale, (geographically) distributed applications.
- Automated knowledge construction
  - to avoid the time-consuming, error-prone manual CE definition development.

Requirements

- Expressive representation
  - to capture complex relationships between the events that stream into the system.
- Efficient reasoning
  - to support real-time decision-making in large-scale, (geographically) distributed applications.
- Automated knowledge construction
  - to avoid the time-consuming, error-prone manual CE definition development.
- Reasoning under uncertainty
  - to deal with various types of noise.

Requirements

- Expressive representation
  - to capture complex relationships between the events that stream into the system.
- Efficient reasoning
  - to support real-time decision-making in large-scale, (geographically) distributed applications.
- Automated knowledge construction
  - to avoid the time-consuming, error-prone manual CE definition development.
- Reasoning under uncertainty
  - to deal with various types of noise.
- Complex event forecasting
  - to support proactive decision-making.

Course Structure

- Introduction to Complex Event Recognition (CER).
  - We have Deep Learning and it seems to work — can we go home now?
- Formal Models for CER
  - ... including interval-based and incremental CER.
- Probabilistic CER.
- Complex Event Forecasting.
- Open issues & further research.

Complex Event Recognition & Related Research

Complex Event Recognition vs DataBase Management Systems*

Complex event recognition (CER) systems:

- Process data without storing them.

*Gugola and Margara, Processing Flows of Information: From Data Stream to Complex Event Processing. ACM Computing Surveys, 2012.

Complex Event Recognition vs DataBase Management Systems*

Complex event recognition (CER) systems:

- Process data without storing them.
- Data are continuously updated.
  - Data stream into the system in high velocity.
  - Data streams are large (usually unbounded).

*Gugola and Margara, Processing Flows of Information: From Data Stream to Complex Event Processing. ACM Computing Surveys, 2012.

Complex Event Recognition vs DataBase Management Systems*

Complex event recognition (CER) systems:

- Process data without storing them.
- Data are continuously updated.
  - Data stream into the system in high velocity.
  - Data streams are large (usually unbounded).
- No assumption can be made on data arrival order.

*Gugola and Margara, Processing Flows of Information: From Data Stream to Complex Event Processing. ACM Computing Surveys, 2012.

Complex Event Recognition vs DataBase Management Systems*

Complex event recognition (CER) systems:

- Process data without storing them.
- Data are continuously updated.
  - Data stream into the system in high velocity.
  - Data streams are large (usually unbounded).
- No assumption can be made on data arrival order.
- Users install standing/continuous queries:
  - Queries deployed once and executed continuously until removed.
  - Online reasoning.

*Gugola and Margara, Processing Flows of Information: From Data Stream to Complex Event Processing. ACM Computing Surveys, 2012.

Complex Event Recognition vs DataBase Management Systems*

Complex event recognition (CER) systems:

- Process data without storing them.
- Data are continuously updated.
  - Data stream into the system in high velocity.
  - Data streams are large (usually unbounded).
- No assumption can be made on data arrival order.
- Users install standing/continuous queries:
  - Queries deployed once and executed continuously until removed.
  - Online reasoning.
- Latency requirements are very strict.

*Gugola and Margara, Processing Flows of Information: From Data Stream to Complex Event Processing. ACM Computing Surveys, 2012.

Complex Event Recognition vs Data Stream Management Systems

Data stream management systems:

- Handle (unbounded) streams as opposed to tables.
- Typically perform a single-pass over the data.
- Support continuous queries.

Complex Event Recognition vs Data Stream Management Systems

Data stream management systems:

- Handle (unbounded) streams as opposed to tables.
- Typically perform a single-pass over the data.
- Support continuous queries.
- Language with limited expressivity.

Complex Event Recognition vs Data Stream Management Systems

Data stream management systems:

- Handle (unbounded) streams as opposed to tables.
- Typically perform a single-pass over the data.
- Support continuous queries.
- Language with limited expressivity.

CER systems:

- Represent complex temporal patterns of events.

Complex Event Recognition vs Data Stream Management Systems

Data stream management systems:

- Handle (unbounded) streams as opposed to tables.
- Typically perform a single-pass over the data.
- Support continuous queries.
- Language with limited expressivity.

CER systems:

- Represent complex temporal patterns of events.
- The patterns may also be extended by spatial operators.

Complex Event Recognition vs Data Stream Management Systems

Data stream management systems:

- Handle (unbounded) streams as opposed to tables.
- Typically perform a single-pass over the data.
- Support continuous queries.
- Language with limited expressivity.

CER systems:

- Represent complex temporal patterns of events.
- The patterns may also be extended by spatial operators.
  - A sequence of transactions using the same credit card, with an increasing amount withdrawn or spent, in a short period of time, in large distances.

Complex Event Recognition vs Deep Learning

We have Deep Learning and it seems to work. Can we go home?

Complex Event Recognition vs Deep Learning

We have Deep Learning and it seems to work. Can we go home?

CER:

- Formal semantics* for trustworthy models.

*Grez et al, A Formal Framework for Complex Event Recognition. ACM Transactions on Database Systems, 2021.

Complex Event Recognition vs Deep Learning

We have Deep Learning and it seems to work. Can we go home?

CER:

- Formal semantics* for trustworthy models.
- Explanation — why did we detect a complex event?

*Grez et al, A Formal Framework for Complex Event Recognition. ACM Transactions on Database Systems, 2021.

Complex Event Recognition vs Deep Learning

We have Deep Learning and it seems to work. Can we go home?

CER:

- Formal semantics* for trustworthy models.
- Explanation — why did we detect a complex event?
- Machine Learning is necessary. But:
  - Complex events are rare.
  - Supervision is scarce.

*Grez et al, A Formal Framework for Complex Event Recognition. ACM Transactions on Database Systems, 2021.

Complex Event Recognition vs Deep Learning

We have Deep Learning and it seems to work. Can we go home?

CER:

- Formal semantics* for trustworthy models.
- Explanation — why did we detect a complex event?
- Machine Learning is necessary. But:
  - Complex events are rare.
  - Supervision is scarce.
- More often than not, background knowledge is available — let's use it!

*Grez et al, A Formal Framework for Complex Event Recognition. ACM Transactions on Database Systems, 2021.

Models of Complex Event Recognition

Models of Complex Event Recognition Systems

- Data model.
- Time model.
- Pattern language model.
- Processing model.
- Deployment model.

Data Model

- An event is an object in the form of a tuple of data components, signifying an activity and holding certain relationships to other events by time, causality and aggregation.
- An event with N attributes can be represented as $EventType(Attr_1, \ldots, Attr_N, T)$ where the timestamp T is
  - a point for an instantaneous event;
  - an interval for a durative event.

Data Model

- An event is an object in the form of a tuple of data components, signifying an activity and holding certain relationships to other events by time, causality and aggregation.
- An event with N attributes can be represented as $EventType(Attr_1, \ldots, Attr_N, T)$ where the timestamp T is
  - a point for an instantaneous event;
  - an interval for a durative event.
- For input events, T represents the event occurrence time or the time at which the event arrives at the CER system.
- For output events, T is typically the result of reasoning on the timestamps of input events.

Data Model

- An event is an object in the form of a tuple of data components, signifying an activity and holding certain relationships to other events by time, causality and aggregation.
- An event with N attributes can be represented as $EventType(Attr_1, \ldots, Attr_N, T)$ where the timestamp T is
  - a point for an instantaneous event;
  - an interval for a durative event.
- For input events, T represents the event occurrence time or the time at which the event arrives at the CER system.
- For output events, T is typically the result of reasoning on the timestamps of input events.
- Event streams are typically heterogeneous: events have different payload (number and type of attributes).

Data Model: Examples

Maritime situational awareness:

- Instantaneous input/simple event:
  $speedChange(vessel_{17}, high, 10{:}15{:}02)$.
- Durative input/simple event:
  $stopped(vessel_{22}, [10{:}23{:}12, 10{:}32{:}10])$.

Data Model: Examples

Maritime situational awareness:

- Instantaneous input/simple event:
  $speedChange(vessel_{17}, high, 10{:}15{:}02)$.
- Durative input/simple event:
  $stopped(vessel_{22}, [10{:}23{:}12, 10{:}32{:}10])$.
- Output/complex events (CEs) are typically durative.
- Durative CE: $illegalFishing(vessel_{45}, [9{:}26{:}12, 12{:}42{:}16])$.

Data Model: Examples

Maritime situational awareness:

- Instantaneous input/simple event:
  $speedChange(vessel_{17}, high, 10{:}15{:}02)$.
- Durative input/simple event:
  $stopped(vessel_{22}, [10{:}23{:}12, 10{:}32{:}10])$.
- Output/complex events (CEs) are typically durative.
- Durative CE: $illegalFishing(vessel_{45}, [9{:}26{:}12, 12{:}42{:}16])$.
- Output events (CEs) are often relational.
- Durative, relational CE:
  $tugging(vessel_{72}, vessel_{33}, [10{:}23{:}12, 10{:}57{:}10])$.

Time Model

Implicit representation (eg, in data stream management systems):

- Event timestamps are used for ordering events before entering the CER engine, and ignored afterwards.
- Sometimes the timestamps are also used for selecting a subset of the input stream.

Time Model

Implicit representation (eg, in data stream management systems):

- Event timestamps are used for ordering events before entering the CER engine, and ignored afterwards.
- Sometimes the timestamps are also used for selecting a subset of the input stream.

Explicit representation (CER):

- Event timestamps are explicitly used in pattern matching.
  - Human activity recognition: two people are said to be moving together if they are walking at the same time.
  - Credit card fraud detection: increasing amounts withdrawn within minutes.

Pattern Language Model

- CER refers to matching patterns among the incoming streams of simple events (SE)s.
- Thus, we need a language for expressing such patterns.
- We present a basic event algebra with common operators.
- Some systems extend this algebra with additional operators.

A Simple Unifying Event Algebra

$$
\begin{array}{lll}
ce ::= & se & \mid \\
& ce_1 \ ; \ ce_2 & \mid \textit{Sequence} \\
& ce_1 \ \vee \ ce_2 & \mid \textit{Disjunction} \\
& ce^* & \mid \textit{Iteration} \\
& \neg \ ce & \mid \textit{Negation} \\
& \sigma_\theta(ce) & \mid \textit{Selection} \\
& \pi_m(ce) & \mid \textit{Projection} \\
& [ce]_{T_1}^{T_2} & \textit{Windowing (from } T_1 \text{ to } T_2\text{)}
\end{array}
$$

- *Sequence*: Two events following each other in time.

A Simple Unifying Event Algebra

$$
\begin{array}{lll}
ce ::= & se & \\
& ce_1 \ ; \ ce_2 & \mid \textit{Sequence} \\
& ce_1 \ \vee \ ce_2 & \mid \textit{Disjunction} \\
& ce^* & \mid \textit{Iteration} \\
& \neg \ ce & \mid \textit{Negation} \\
& \sigma_\theta(ce) & \mid \textit{Selection} \\
& \pi_m(ce) & \mid \textit{Projection} \\
& [ce]_{T_1}^{T_2} & \textit{Windowing (from } T_1 \text{ to } T_2\text{)}
\end{array}
$$

- *Sequence*: Two events following each other in time.
- *Disjunction*: Either of two events occurring, regardless of temporal relations.

A Simple Unifying Event Algebra

$$
\begin{array}{lll}
ce ::= & se & \mid \\
& ce_1 \ ; \ ce_2 & \mid \textit{Sequence} \\
& ce_1 \ \vee \ ce_2 & \mid \textit{Disjunction} \\
& ce^* & \mid \textit{Iteration} \\
& \neg \ ce & \mid \textit{Negation} \\
& \sigma_\theta(ce) & \mid \textit{Selection} \\
& \pi_m(ce) & \mid \textit{Projection} \\
& [ce]_{T_1}^{T_2} & \textit{Windowing (from } T_1 \text{ to } T_2\text{)}
\end{array}
$$

- *Sequence*: Two events following each other in time.
- *Disjunction*: Either of two events occurring, regardless of temporal relations.
- The combination of *Sequence* and *Disjunction* expresses Conjunction (both events occurring).

A Simple Unifying Event Algebra

$$
\begin{array}{lll}
ce ::= & se & | \\
& ce_1 \ ; \ ce_2 & | \ \textit{Sequence} \\
& ce_1 \ \vee \ ce_2 & | \ \textit{Disjunction} \\
& ce^* & | \ \textit{Iteration} \\
& \neg \ ce & | \ \textit{Negation} \\
& \sigma_\theta(ce) & | \ \textit{Selection} \\
& \pi_m(ce) & | \ \textit{Projection} \\
& [ce]_{T_1}^{T_2} & \textit{Windowing (from } T_1 \text{ to } T_2)
\end{array}
$$

- *Iteration*: An event occurring N times in sequence, where $N \geq 0$. This operation is similar to the *Kleene star* operation in regular expressions, the difference being that *Kleene star* is unbounded.

A Simple Unifying Event Algebra

$$
\begin{array}{lll}
ce ::= & se & | \\
& ce_1 \ ; \ ce_2 & | \ \textit{Sequence} \\
& ce_1 \ \vee \ ce_2 & | \ \textit{Disjunction} \\
& ce^* & | \ \textit{Iteration} \\
& \neg \ ce & | \ \textit{Negation} \\
& \sigma_\theta(ce) & | \ \textit{Selection} \\
& \pi_m(ce) & | \ \textit{Projection} \\
& [ce]_{T_1}^{T_2} & \textit{Windowing (from } T_1 \text{ to } T_2\text{)}
\end{array}
$$

- *Negation*: Absence of event occurrence.

A Simple Unifying Event Algebra

$$
\begin{array}{lll}
ce ::= & se & \\
& ce_1 \ ; \ ce_2 & \mid \textit{Sequence} \\
& ce_1 \ \vee \ ce_2 & \mid \textit{Disjunction} \\
& ce^* & \mid \textit{Iteration} \\
& \neg \ ce & \mid \textit{Negation} \\
& \sigma_\theta(ce) & \mid \textit{Selection} \\
& \pi_m(ce) & \mid \textit{Projection} \\
& [ce]_{T_1}^{T_2} & \textit{Windowing (from } T_1 \text{ to } T_2\text{)}
\end{array}
$$

- *Negation*: Absence of event occurrence.
- *Selection*: Select those events whose attributes satisfy a set of predicates/relations θ, temporal or otherwise.

A Simple Unifying Event Algebra

$ce ::=$	se	|
	$ce_1 \ ; \ ce_2$	| *Sequence*
	$ce_1 \ \vee \ ce_2$	| *Disjunction*
	ce^*	| *Iteration*
	$\neg \ ce$	| *Negation*
	$\sigma_\theta(ce)$	| *Selection*
	$\pi_m(ce)$	| *Projection*
	$[ce]_{T_1}^{T_2}$	*Windowing* (from T_1 to T_2)

- *Projection*: Return an event whose attribute values are a possibly transformed subset of the attribute values of its sub-events.

A Simple Unifying Event Algebra

$$
\begin{array}{lll}
ce ::= & se & \\
& ce_1 \ ; \ ce_2 & \mid \textit{Sequence} \\
& ce_1 \ \vee \ ce_2 & \mid \textit{Disjunction} \\
& ce^* & \mid \textit{Iteration} \\
& \neg \ ce & \mid \textit{Negation} \\
& \sigma_\theta(ce) & \mid \textit{Selection} \\
& \pi_m(ce) & \mid \textit{Projection} \\
& [ce]_{T_1}^{T_2} & \textit{Windowing (from } T_1 \text{ to } T_2\text{)}
\end{array}
$$

- *Projection*: Return an event whose attribute values are a possibly transformed subset of the attribute values of its sub-events.
- *Windowing*: Evaluate the conditions of an event pattern within a specified time window.

Types of Window

- Logical (time-based) windows: bounds are defined as a function of time.
  - Example: Match a pattern only on the events received in the last 10 minutes.
- Physical (count-based) windows: bounds depend on the number of items included in the window.
  - Example: Match a pattern only on the last 10 received events.

Types of Window

- Logical (time-based) windows: bounds are defined as a function of time.
  - Example: Match a pattern only on the events received in the last 10 minutes.
- Physical (count-based) windows: bounds depend on the number of items included in the window.
  - Example: Match a pattern only on the last 10 received events.
- In either case, both bounds advance with a pre-defined logical or physical step.
  - Pane windows: overlapping sliding windows.
  - Tumble windows: non-overlapping sliding windows.

Processing Model

Selection strategies filter the set of matched patterns.

- Assume the pattern $\alpha; \beta$ and the stream $(\alpha, 1),\ (\alpha, 2),\ (\beta, 3)$.

Processing Model

Selection strategies filter the set of matched patterns.

- Assume the pattern $\alpha; \beta$ and the stream $(\alpha, 1),\ (\alpha, 2),\ (\beta, 3)$.
- The multiple selection strategy produces $(\alpha, 1),\ (\beta, 3)$ and $(\alpha, 2),\ (\beta, 3)$.

Processing Model

Selection strategies filter the set of matched patterns.

- Assume the pattern $\alpha; \beta$ and the stream $(\alpha, 1),\ (\alpha, 2),\ (\beta, 3)$.
- The multiple selection strategy produces $(\alpha, 1),\ (\beta, 3)$ and $(\alpha, 2),\ (\beta, 3)$.
- The single selection strategy produces either $(\alpha, 1),\ (\beta, 3)$ or $(\alpha, 2),\ (\beta, 3)$.
- The single selection strategy represents a family of strategies, depending on the matches actually chosen among all possible ones.

Processing Model

Consumption policies place constraints on the use of events.

- Assume the pattern $\alpha; \beta$ and the stream $(\alpha, 1),\ (\beta, 2),\ (\beta, 3)$.

Processing Model

Consumption policies place constraints on the use of events.

- Assume the pattern $\alpha; \beta$ and the stream $(\alpha, 1),\ (\beta, 2),\ (\beta, 3)$.
- The zero consumption policy produces $(\alpha, 1),\ (\beta, 2)$ and $(\alpha, 1),\ (\beta, 3)$.
  - ... assuming a multiple selection strategy.

Processing Model

Consumption policies place constraints on the use of events.

- Assume the pattern $\alpha; \beta$ and the stream $(\alpha, 1),\ (\beta, 2),\ (\beta, 3)$.
- The zero consumption policy produces $(\alpha, 1),\ (\beta, 2)$ and $(\alpha, 1),\ (\beta, 3)$.
  - ... assuming a multiple selection strategy.
- The selected consumption policy produces $(\alpha, 1),\ (\beta, 2)$.
  - $(\alpha, 1)$ is consumed when the pattern is matched (at the arrival of $(\beta, 2)$), and thus no longer available when $(\beta, 3)$ arrives.
  - Once $(\alpha, 1)$ is consumed, it is not considered in ANY other pattern!

Deployment Model

- Centralised CER: all incoming event streams are processed by a single node.
- Distributed CER: several nodes are utilised.
- Distribution type:
  - Events. CER on different (possibly disjoint) streams.
  - Patterns. CER on different patterns (possibly compiled by a pattern rewriting process).

Deployment Model

- Centralised CER: all incoming event streams are processed by a single node.
- Distributed CER: several nodes are utilised.
- Distribution type:
  - Events. CER on different (possibly disjoint) streams.
  - Patterns. CER on different patterns (possibly compiled by a pattern rewriting process).
- Distribution method:
  - Cluster. CER by means of strongly connected machines.
  - Wide area network. Mimimise network usage and support data privacy by processing events as close as possible to the sources.
  - In-situ processing. Processing events at the sources.

A Typical Complex Event Recognition Language (SASE)

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).

*Zhang et al. On complexity and optimization of expensive queries in complex event processing. SIGMOD 2014.

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($A, B+, C$).

*Zhang et al. On complexity and optimization of expensive queries in complex event processing. SIGMOD 2014.

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($A, !B, C$).

*Zhang et al. On complexity and optimization of expensive queries in complex event processing. SIGMOD 2014.

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*.

*Zhang et al. On complexity and optimization of expensive queries in complex event processing. SIGMOD 2014.

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 $\mathrm{SEQ}(A, B, C)+$.

Event Algebra

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

Selection Strategies

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

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.

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. For SEQ(A, B, C) e.g. and a_1, b_1, b_2, c_1, only a_1, b_1, c_1 will be detected.

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. For SEQ(A, B, C) e.g. 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 (1)

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

Example (1)

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

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 refer to the same vessel (*vesselId*) and are contiguous with respect to that vessel.

Example (2)

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

Example (2)

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

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.

Summary

What we've seen so far:

- Definition, Challenges & Applications of CER.
- Related research.
- Models of CER.

Summary

What we've seen so far:

- Definition, Challenges & Applications of CER.
- Related research.
- Models of CER.

CER requirements:

- Expressive representation.
- Efficient reasoning.
- Automated knowledge construction.
- Reasoning under uncertainty.
- Complex event forecasting.

Summary

What we've seen so far:

- Definition, Challenges & Applications of CER.
- Related research.
- Models of CER.

CER requirements:

- Expressive representation.
- Efficient reasoning.
- Automated knowledge construction.
- Reasoning under uncertainty.
- Complex event forecasting.

Next: An expressive language with an efficient implementation.