

Complex Event Forecasting: a Formal Framework

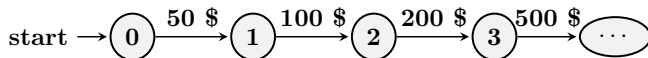
Elias Alevizos

Department of Informatics
National and Kapodistrian University of Athens,
Institute of Informatics & Telecommunications
National Centre for Scientific Research “Demokritos”

January 12, 2022

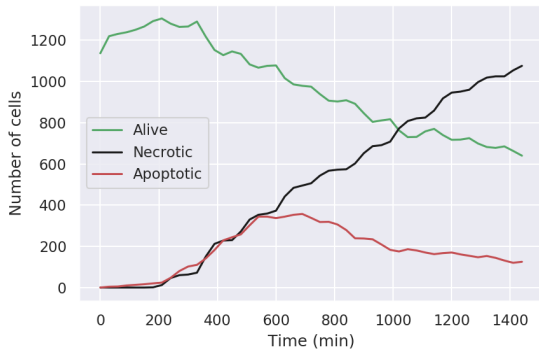
Introduction

Motivation



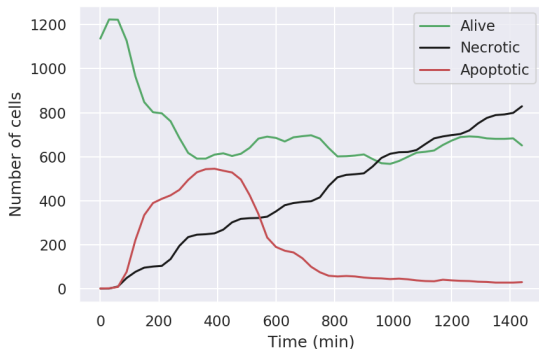
- ▶ Is this a fraud?
- ▶ How long will it last?
- ▶ With what probability?

Motivation



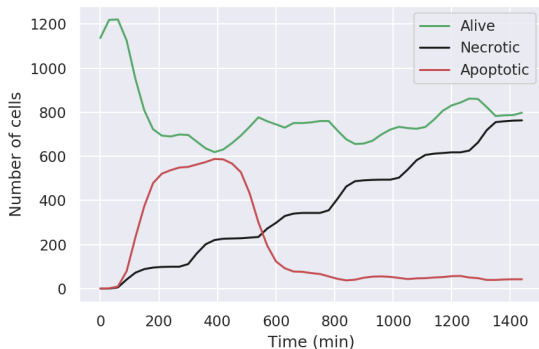
► Is this a promising therapeutic regime?

Motivation



► How about this?

Motivation

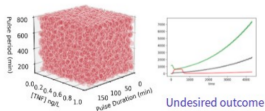


► This one?

Motivation

Standard Model Exploration

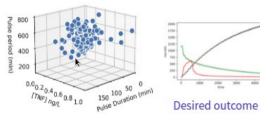
- Number of simulations : 3387
- Wall time: **10h 26min**
- Early discarded → 0



Undesired outcome

Online Exploration + early killing

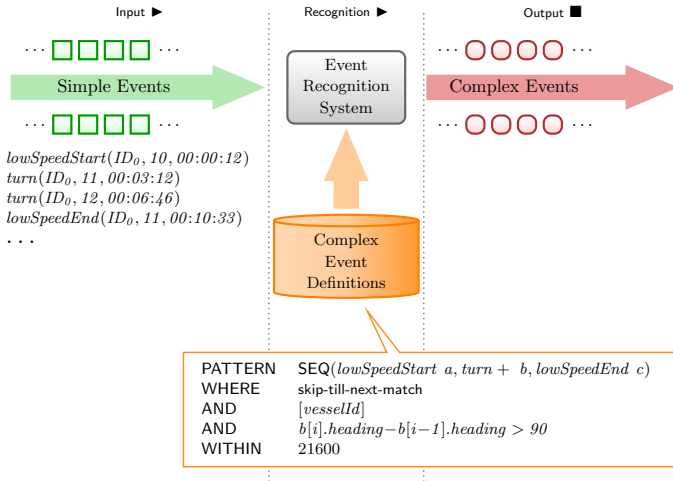
- Number of simulations : 3387
- Wall time: **3h 42min**
- Early discarded → **3086 (~90%)**



Desired outcome

- ▶ Can we kill non-pertinent simulations early?
- ▶ Can we forecast their outcome?
- ▶ How early? How accurately?

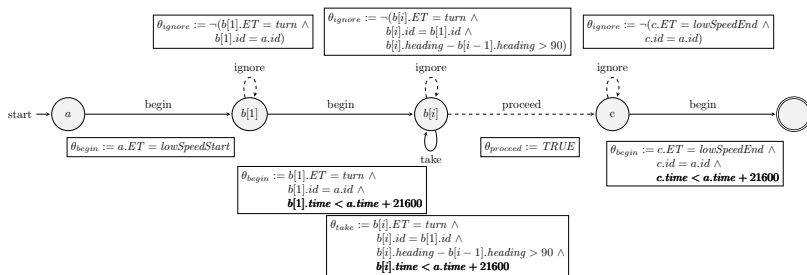
Complex Event Recognition



CER language

$ce ::= \sigma_{\theta}(sde)$	Base case (Boolean expression)
$ce_1 \cdot ce_2$	Sequence
$ce_1 + ce_2$	Disjunction
ce^*	Iteration
$ce_1 \bowtie ce_2$	Conjunction
$! ce$	Negation
$\sigma_{\theta}(ce)$	Selection
$\pi_m(ce)$	Projection
$[ce]_{T_1}^{T_2}$	Windowing

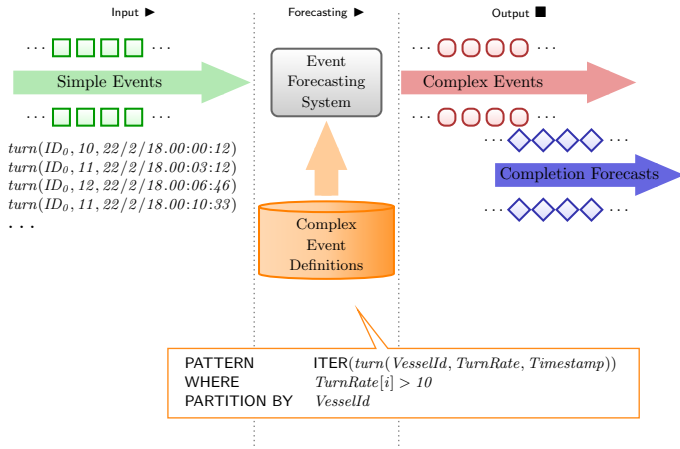
Automata



Logic

$\text{initiatedAt}(\text{withinArea}(\text{VesselId}) = \text{AreaType}, T) \leftarrow$
 $\text{happensAt}(\text{entersArea}(\text{VesselId}, \text{Area}), T),$
 $\text{typeOf}(\text{Area}, \text{AreaType}).$
 $\text{terminatedAt}(\text{withinArea}(\text{VesselId}) = _, T) \leftarrow$
 $\text{happensAt}(\text{exitsArea}(\text{VesselId}, \text{Area}), T).$

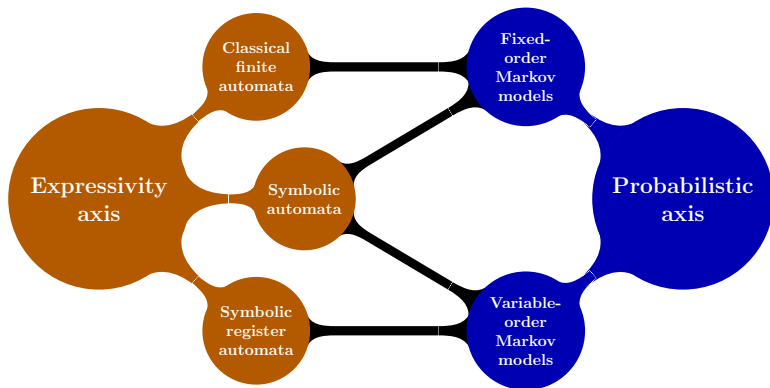
Complex Event Forecasting



Related Work

- ▶ Time-series forecasting [MJK15].
- ▶ Sequence prediction (compression) based on Markov models [BEY04, BW⁺99, RST96, RST93, CW84, WST95].
- ▶ Sequence prediction based on neural networks [LDL18, CPP⁺18].
- ▶ Temporal mining [VM02, LTW08, FBB14, ZCG15, CWY⁺11].
- ▶ Process mining [VDA11, MRR18, FGMM18].
- ▶ CEF, mostly conceptual [FBT⁺12, EE11, CKK16]. Others: [MLJ10, ACMZ15, PNC11, LGC20].
- ▶ Limitations: no language for patterns, focus on discrete symbols or real values (not both), usually target SDE forecasting.

The two axes



Publications (basic)

► Main

- Alevizos, Artikis, Paliouras, **Symbolic Register Automata for Complex Event Recognition and Forecasting**, submitted as journal article
- Alevizos, Artikis, Paliouras, **Complex Event Forecasting with Prediction Suffix Trees**, *VLDB journal*, 2021 [AAP21]
- Alevizos, Artikis, Paliouras, **Wayeb: a Tool for Complex Event Forecasting**, *LPAR*, 2018 [AAP18]
- Alevizos, Artikis, Paliouras, **Event Forecasting with Pattern Markov Chains**, *DEBS*, 2017 [AAP17]

► Surveys

- Giatrakos, Alevizos, Artikis, Deligianakis, Garofalakis, **Complex Event Recognition in the Big Data Era: a Survey**, *VLDB journal*, 2020 [GAA⁺20]
- Alevizos, Artikis, Paliouras, **Probabilistic Complex Event Recognition: a Survey**, *ACM Computing Surveys*, 2017 [ASAP17]

Publications (demos and application papers)

- ▶ Ntoulas, Alevizos, Artikis, Akasiadis, Koumparos, **Online Trajectory Analysis with Scalable Event Recognition**, *GeoInformatica*, 2022 [NAA⁺22]
- ▶ Voudas et al, **Online Distributed Maritime Event Detection and Forecasting over Big Vessel Tracking Data**, *IEEE Big Data*, 2021 [VBK⁺21]
- ▶ Ntoulas, Alevizos, Artikis, Koumparos, **Online Trajectory Analysis with Scalable Event Recognition**, *EDBT Workshops*, 2021 [NAAK21]
- ▶ Voudas et al, **Increasing Maritime Situation Awareness via Trajectory Detection, Enrichment and Recognition of Events**, *W2GIS*, 2018 [VVS⁺18b]
- ▶ Qadah, Mock, Alevizos, Fuchs, **A Distributed Online Learning Approach for Pattern Prediction over Movement Event Streams with Apache Flink**, *EDBT Workshops*, 2018 [QMAF18]
- ▶ Voudas et al, **Big Data Analytics for Time Critical Mobility Forecasting: Recent Progress and Research Challenges**, *EDBT*, 2018 [VVS⁺18a]

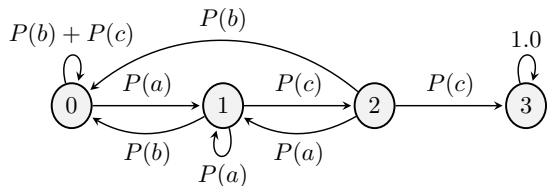
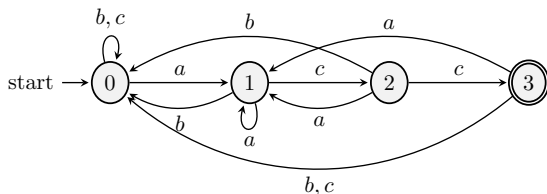
Complex Event Forecasting with Classical Automata and Fixed-order Markov Models [AAP17]

Where do you start?

- ▶ We need to build a probabilistic model to estimate if/when a CE is expected to happen.
- ▶ A probabilistic model of what?
- ▶ Too many formalisms and computational models.
- ▶ Let's start with something simple. Classical regular expressions and automata.

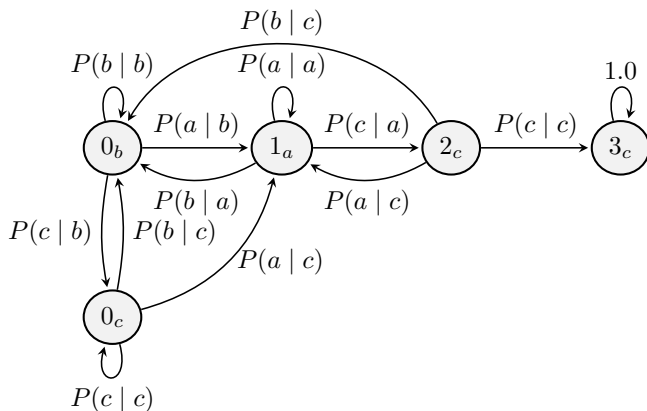
Regular Expression \rightarrow Pattern Markov Chain (PMC)

$R = a \cdot c \cdot c.$ $\Sigma = \{a, b, c\}.$ No memory.



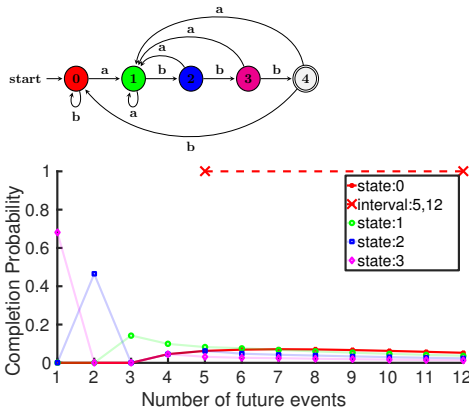
Regular Expression \rightarrow Pattern Markov Chain (PMC)

$$m = 1$$

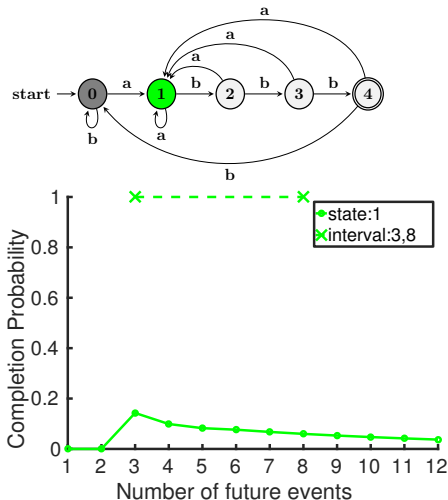


Waiting-Time and Forecasts

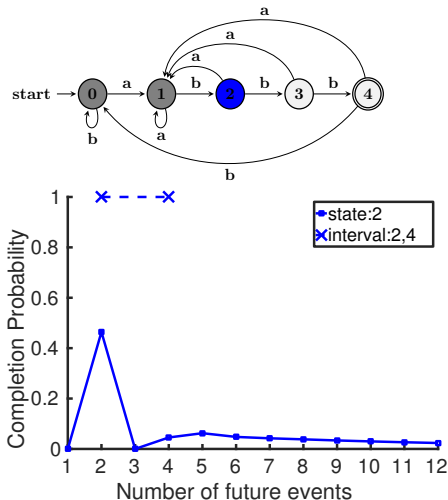
- ▶ Warm-up period to learn distributions.
- ▶ Set a threshold, e.g., $\theta_{fc} = 50\%$.



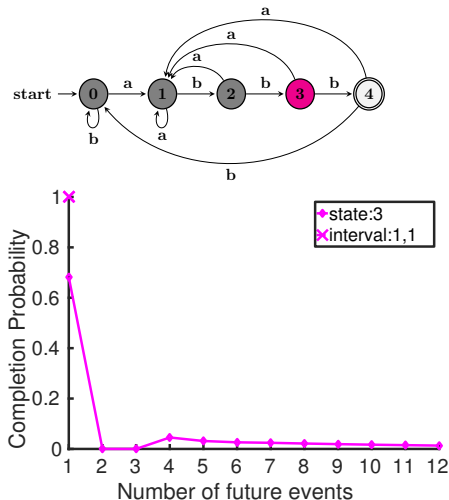
Example: $R = a \cdot b \cdot b \cdot b$.



Example: $R = a \cdot b \cdot b \cdot b$.



Example: $R = a \cdot b \cdot b \cdot b$.



Contributions

- ▶ Regular expressions as opposed to sequential patterns.
- ▶ Forecasts with guaranteed precision, if Markov process.

Limitations

- ▶ Input events are tuples with multiple (numerical and categorical) attributes.
- ▶ Classical automata work on symbols only.
- ▶ Alternative: use symbolic expressions and automata.
- ▶ High-order Markov models lead to a combinatorial explosion of the states.
- ▶ Alternative: use variable-order Markov models, i.e., try to remember only what is “informative”.

Complex Event Forecasting with Symbolic Automata and Fixed-order Markov Models [AAP18]

Symbolic Regular Expressions and Automata

$R_{fish} := x \cdot y^* \cdot z$ WHERE
 ($IsFishingVessel(x) \wedge \neg InArea(x, FishingArea)$) AND
 ($InArea(y, FishingArea) \wedge SpeedBetween(y, 9.0, 20.0)$) AND
 ($InArea(z, FishingArea) \wedge SpeedBetween(z, 1.0, 9.0)$)
PARTITION BY *vesselId*

- ▶ “Terminal symbols”: any unary boolean formula.
- ▶ More complex, but (usually) fewer states and transitions.
- ▶ Proposed in [DV17].

Properties of Symbolic Automata

- ▶ Symbolic automata have nice closure properties.
- ▶ But, can we use them for forecasting, i.e., derive a probabilistic model for them?

Properties of Symbolic Automata

- ▶ Symbolic automata have nice closure properties.
- ▶ But, can we use them for forecasting, i.e., derive a probabilistic model for them?
- ▶ Yes!
 - ▶ Symbolic automata are determinizable.
 - ▶ They can be mapped to “equivalent” classical automata (simple algebraic argument using isomorphism).
 - ▶ Same technique for mapping to Markov Chains.
 - ▶ Matrix probabilities:
$$P(\psi_1(t_{i+1}) \wedge \psi_2(t_{i+1}) = \text{TRUE} \mid \neg\psi_1(t_i) \wedge \neg\psi_2(t_i) = \text{TRUE}).$$

Contributions

- ▶ Probabilistic description of symbolic automata.
- ▶ User-configurable order of assumed Markov process.
- ▶ Nice compositional properties.
- ▶ Works on infinite alphabets.

Limitations

- ▶ Still, need to reduce search space of the (symbolic) past.
- ▶ High-order models lead to a combinatorial explosion of the states.
- ▶ Alternative: variable-order Markov models.
- ▶ Yet more expressive patterns.

Complex Event Forecasting with Symbolic Automata and Variable-order Markov Models [AAP21]

- ▶ Problem: how to avoid the combinatorial explosion on the number of states of a Markov chain.
- ▶ Proposal
 - ▶ A CEF framework that is both formal, compositional and easy to use.
 - ▶ Can uncover deep dependencies by using a variable-order Markov model.

Symbolic regular expression

Definition (Symbolic regular expression)

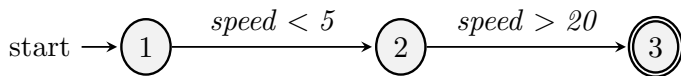
A symbolic regular expression (*SRE*) over an effective Boolean algebra $(\mathcal{D}, \Psi, \llbracket - \rrbracket, \perp, \top, \vee, \wedge, \neg)$ is recursively defined as follows:

- ▶ If $\psi \in \Psi$, then $R := \psi$ is a symbolic regular expression, with $\mathcal{L}(\psi) = \llbracket \psi \rrbracket$, i.e., the language of ψ is the subset of \mathcal{D} for which ψ evaluates to TRUE;
- ▶ Disjunction / Union $R := R_1 + R_2$, with $\mathcal{L}(R) = \mathcal{L}(R_1) \cup \mathcal{L}(R_2)$;
- ▶ Concatenation / Sequence $R := R_1 \cdot R_2$, with $\mathcal{L}(R) = \mathcal{L}(R_1) \cdot \mathcal{L}(R_2)$;
- ▶ Iteration / Kleene-star $R' := R^*$, with $\mathcal{L}(R^*) = (\mathcal{L}(R))^*$.

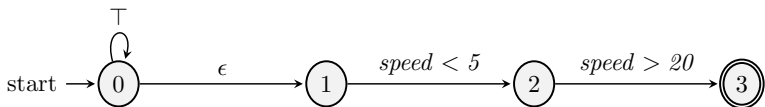
Table: Example event stream from the maritime domain.

Navigationals status	fishing	fishing	fishing	under way	under way	under way	...
vessel id	78986	78986	78986	78986	78986	78986	...
speed	2	1	3	22	19	27	...
timestamp	1	2	3	4	5	6	...

- ▶ $R := (speed < 5) \cdot (speed > 20)$.
- ▶ The third and fourth events belong to the language of R .



(a) SFA for the SRE $R := (speed < 5) \cdot (speed > 20)$.



(b) Streaming SFA for $R := (speed < 5) \cdot (speed > 20)$.

Proposition

For every symbolic regular expression R there exists a symbolic finite automaton M such that $\mathcal{L}(R) = \mathcal{L}(M)$.

Variable-order Markov models

- ▶ Let Σ denote an alphabet, $\sigma \in \Sigma$ a symbol from that alphabet and $s \in \Sigma^m$ a string of length m of symbols from that alphabet.
- ▶ Goal: derive a predictor \hat{P} such that the average log-loss on a test sequence $S_{1..k}$ is minimized.
- ▶ For full-order Markov models, \hat{P} derived through conditional distributions $\hat{P}(\sigma \mid s)$, with m constant.
- ▶ VMMs relax the assumption of m being fixed. Length of “context” s may vary.

Isomorphism

Lemma

For every deterministic symbolic finite automaton (DSFA) M_s there exists a deterministic classical finite automaton (DFA) M_c such that $\mathcal{L}(M_c)$ is the set of strings induced by applying $N = \text{Minterms}(\text{Predicates}(M_s))$ to $\mathcal{L}(M_s)$.

- We can use symbols from now on instead of predicates.

Prediction Suffix Trees

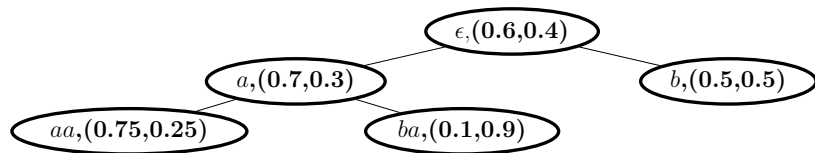
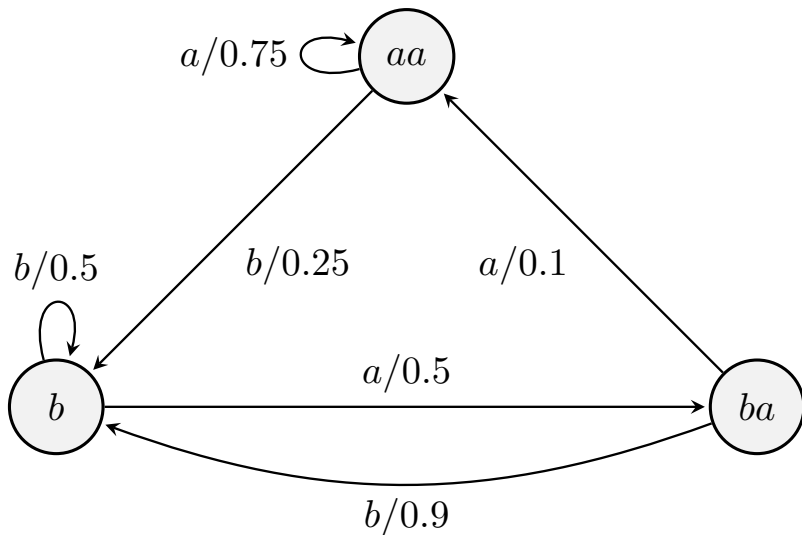


Figure: Example *PST* T for $\Sigma = \{a, b\}$ and $m = 2$. Each node contains the label and the next symbol probability distribution for a and b .

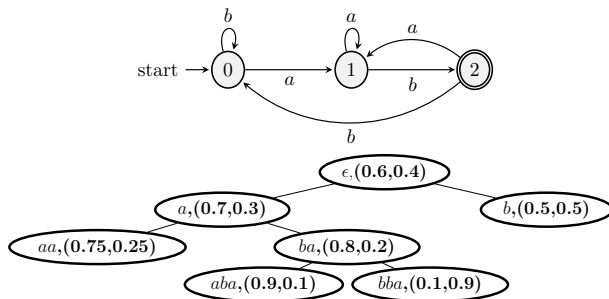
Probabilistic Suffix Automata



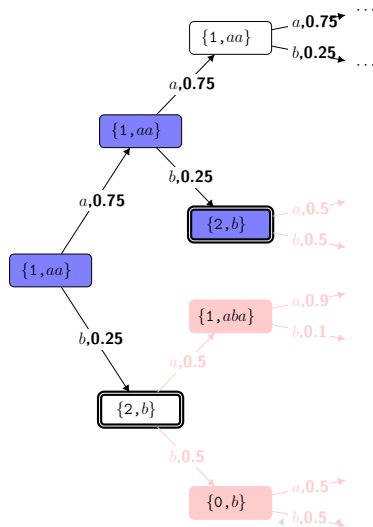
Emitting forecasts

- ▶ Trade-off between memory and computation efficiency.
- ▶ If online performance critical, use *PSA*.
- ▶ If high accuracy (thus high order values) necessary, use *PST* bypassing the *PSA*.

Forecasting with *PST*



Forecasts with *PST*



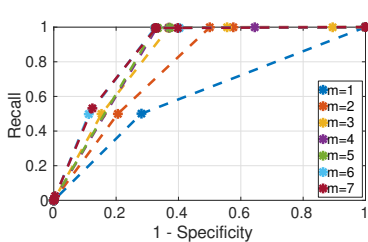
Credit card fraud management

$$R := (amountDiff > 0) \cdot (amountDiff > 0) \cdot (amountDiff > 0) \cdot \\ (amountDiff > 0) \cdot (amountDiff > 0) \cdot (amountDiff > 0) \cdot \\ (amountDiff > 0)$$

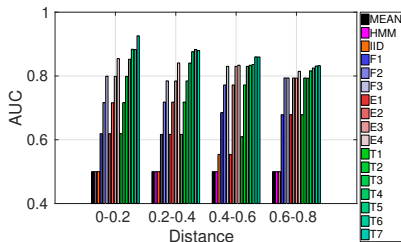
Dataset

- ▶ 1,000,000 transactions in total from 100 different cards.
- ▶ About 20% of the transactions are fraudulent.
- ▶ 7 different types of known fraudulent patterns.
- ▶ Each fraudulent sequence for the increasing trend consists of eight consecutive transactions with increasing amounts, where the amount is increased each time by 100 monetary units or more.
- ▶ 75% of the original dataset for training and the rest for testing.

ROC

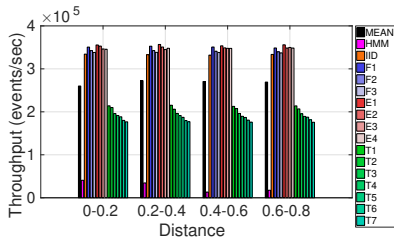


(a) ROC curves for the variable-order model using the *PST* models for various values of the maximum order. $distance \in [0.4, 0.6]$.

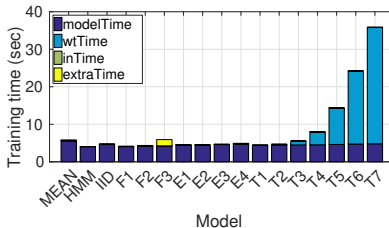


(b) AUC for ROC curves for all

Performance



(a) Throughput.



(b) Training time.

Memory footprint

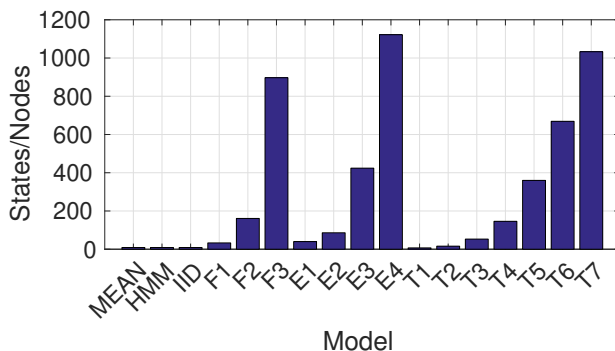


Figure: Number of states/nodes.

Maritime



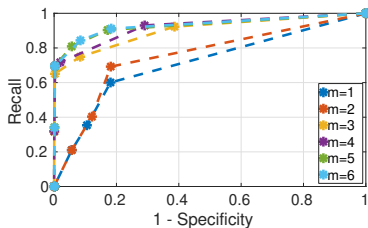
Pattern

$$R := (\neg \text{InsidePort}(\text{Brest}))^* \cdot (\neg \text{InsidePort}(\text{Brest})) \cdot \\ (\neg \text{InsidePort}(\text{Brest})) \cdot (\text{InsidePort}(\text{Brest}))$$

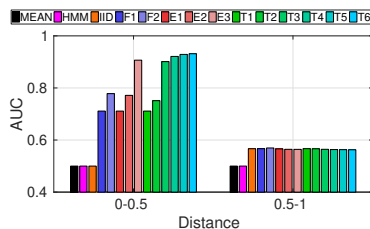
Dataset

- ▶ AIS kinematic messages from vessels sailing in the Atlantic Ocean around the port of Brest, France.
- ▶ Spans a period from 1 October 2015 to 31 March 2016.
- ▶ The vessel with the most matches: 368 matches and ≈ 30.000 SDEs.
- ▶ Vessels with more than 100 matches. 9 such vessels with ≈ 222.000 events.

ROC

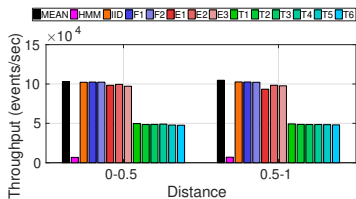


(a) ROC curves for the variable-order model using the *PST* models for various values of the maximum order. $distance \in [0.0, 0.5]$.

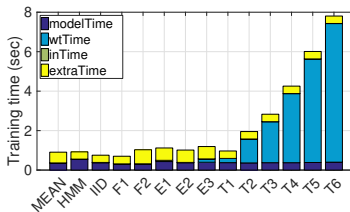


(b) AUC for ROC curves for all

Performance

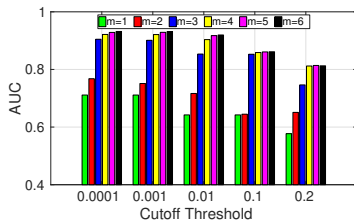


(a) Throughput.

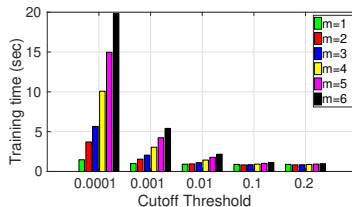


(b) Training time.

Effect of cutoff threshold



(a) AUC-ROC.



(b) Training time.

Contributions

- ▶ A formal, compositional and easy to use CEF framework.
- ▶ Can uncover deep dependencies.
- ▶ Various types of forecasting. Subsumes previous methods restricted to one type of forecasting.
- ▶ Proposed a more comprehensive set of metrics.

Complex Event Forecasting with Symbolic Register Automata and Variable-order Markov Models

Motivation

- ▶ Need to be able to use n -ary expressions.
- ▶ We thus need automata with memory,
- ▶ Need to understand the closure properties of automata with memory.
- ▶ We must show whether such automata can be used for forecasting.

Definition (Symbolic regular expression with memory (*SREM*))

A symbolic regular expression with memory over a \mathcal{V} -structure \mathcal{M} and a set of register variables $R = \{r_1, \dots, r_k\}$ is inductively defined as follows:

1. ϵ and \emptyset are *SREM*.
2. If ϕ is a condition, then ϕ is a *SREM*.
3. If ϕ is a condition, then $\phi \downarrow r_i$ is a *SREM*.
4. If e_1 and e_2 are *SREM*, then $e_1 + e_2$ is also a *SREM*.
5. If e_1 and e_2 are *SREM*, then $e_1 \cdot e_2$ is also a *SREM*.
6. If e is a *SREM*, then e^* is also a *SREM*. ◀

Table: Example stream.

type	T	T	T	H	H	T	...
id	1	1	2	1	1	2	...
value	22	24	32	70	68	33	...
index	1	2	3	4	5	6	...

$$e_1 := (TypeIsT(\sim) \downarrow r_1) \cdot (\top)^* \cdot (TypeIsH(\sim) \wedge EqualId(\sim, r_1))$$

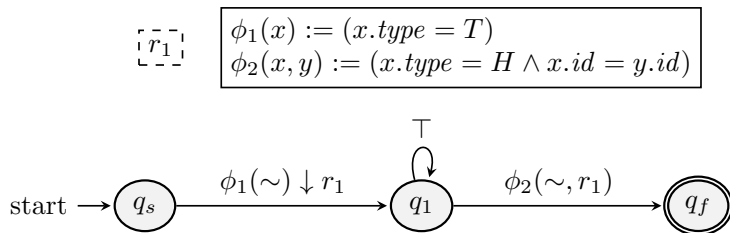


Table: Example stream.

type	T	T	T	H	H	T	...
id	1	1	2	1	1	2	...
value	22	24	32	70	68	33	...
index	1	2	3	4	5	6	...

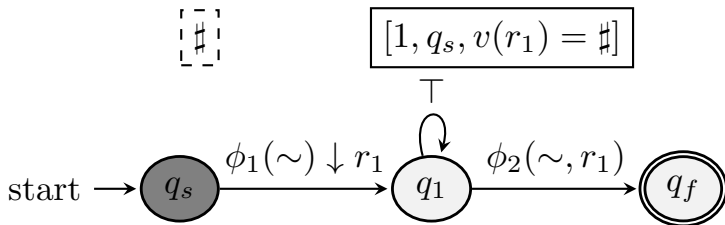


Table: Example stream.

type	T	T	T	H	H	T	...
id	1	1	2	1	1	2	...
value	22	24	32	70	68	33	...
index	1	2	3	4	5	6	...

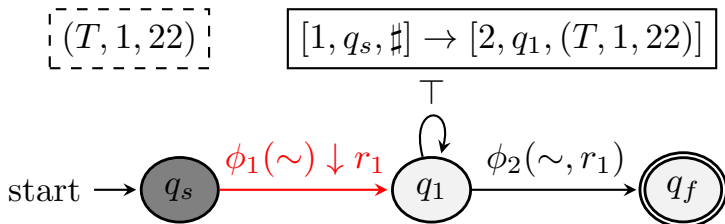


Table: Example stream.

type	T	T	T	H	H	T	...
id	1	1	2	1	1	2	...
value	22	24	32	70	68	33	...
index	1	2	3	4	5	6	...

$$(T, 1, 22)$$

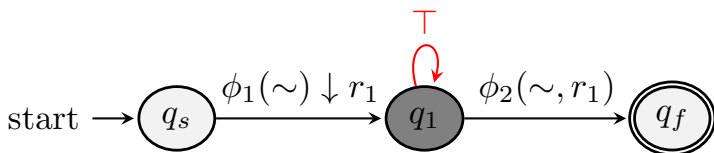
$$[1, q_s, \#] \rightarrow [2, q_1, (T, 1, 22)] \rightarrow [3, q_1, (T, 1, 22)]$$


Table: Example stream.

type	T	T	T	H	H	T	...
id	1	1	2	1	1	2	...
value	22	24	32	70	68	33	...
index	1	2	3	4	5	6	...

$$[(T, 1, 22)]$$

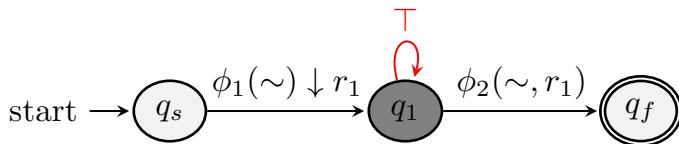
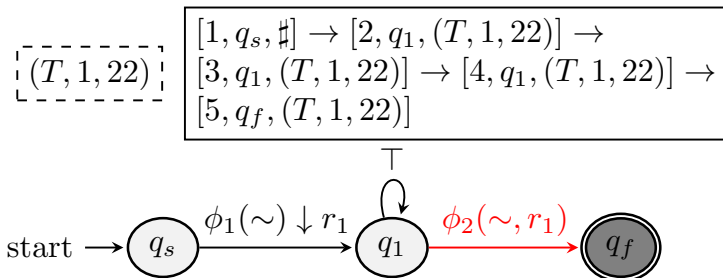
$$[1, q_s, \#] \rightarrow [2, q_1, (T, 1, 22)] \rightarrow [3, q_1, (T, 1, 22)] \rightarrow [4, q_1, (T, 1, 22)]$$


Table: Example stream.

type	T	T	T	H	H	T	...
id	1	1	2	1	1	2	...
value	22	24	32	70	68	33	...
index	1	2	3	4	5	6	...



Theorem

For every SREM e there exists an equivalent SRA A , i.e., a SRA such that $\mathcal{L}(e) = \mathcal{L}(A)$.

Theorem

For every SRA A there exists an equivalent SREM e , i.e., a SREM such that $\mathcal{L}(A) = \mathcal{L}(e)$.

Theorem

SRA and SREM are closed under union, intersection, concatenation and Kleene-star.

Theorem

SRA and SREM are not closed under complement.

Theorem

SRA are not closed under determinization.

Definition (Windowed *SREM*)

Let e be a *SREM* over a \mathcal{V} -structure \mathcal{M} and a set of register variables $R = \{r_1, \dots, r_k\}$, S a string constructed from elements of the universe of \mathcal{M} and $v, v' \in F(r_1, \dots, r_k)$. A windowed *SREM* (*wSREM*) is an expression of the form $e' := e^{[1..w]}$, where $w \in \mathbb{N}_1$. We define the relation $(e', S, v) \vdash v'$ as follows:
 $(e, S, v) \vdash v'$ and $|S| \leq w$.

Theorem

For every windowed SREM there exists an equivalent deterministic SRA.

Corollary

Windowed SRA are closed under complement.

- ▶ We have only unary conditions applied to the last event and an arbitrary (finite or infinite) universe. In this case, we do not need registers.
- ▶ We have n -ary conditions (with $n \geq 1$) and a finite universe. In this case, registers are helpful, but may not be necessary. If we have a register automaton A and a finite universe \mathcal{U} , we can always create an automaton $A_{\mathcal{U}}$ with states $A.Q \times \mathcal{U}$ and appropriate transitions so that $A_{\mathcal{U}}$ is equivalent to A but has no registers. Its states can implicitly remember past elements.
- ▶ The most complex case is when we have n -ary conditions and an infinite universe, as is typically assumed in CER/F. Registers are necessary in this case. Symbols generated by automaton.

Contributions

- ▶ Presented an automaton model, *SRA*, that can act as a computational model for patterns with n -ary conditions.
- ▶ Extend the expressive power of symbolic automata and register automata.
- ▶ Most of the standard operators in CER, such as concatenation/sequence, union/disjunction, intersection/conjunction and Kleene-star/iteration, may be used freely.
- ▶ Complement may be used and determinization is also possible, if a window operator is used.

Appendix

Appendix

Time-series forecasting

- ▶ Typically focuses on streams of (mostly) real-valued variables.
- ▶ Goal is to forecast relatively simple patterns.
- ▶ Does not provide a language with which we can define complex patterns.
- ▶ Tries to forecast the next value(s) from the input stream/series, i.e., SDE forecasting.
 - ▶ Not very useful for CER.
 - ▶ Majority of SDE instances ignored.
 - ▶ Forward recognition on projected stream does not work.
- ▶ [MJK15]

Sequence prediction (compression) based on Markov models

- ▶ Does not provide a language for patterns.
- ▶ Focuses exclusively on next symbol prediction.
- ▶ Works on single-variable discrete sequences of symbols.
- ▶ [BEY04, BW⁺99, RST96, RST93, CW84, WST95]

Sequence prediction based on neural networks

- ▶ They do not provide a language for defining complex patterns.
- ▶ Focus is on SDE forecasting.
- ▶ Statistical methods have often been proven to be more accurate and less demanding in terms of computational resources than ML ones in time-series forecasting.
- ▶ [LDL18, CPP⁺18]

Temporal mining

- ▶ Association rule mining, HMMs, DAGs, etc.
- ▶ They target simple patterns, defined either as strict sequences or as sets of input events.
- ▶ The input stream is composed of symbols from a finite alphabet.
- ▶ [VM02, LTW08, FBB14, ZCG15, CWY⁺11]

Process mining

- ▶ Processes are usually given directly as transition systems.
- ▶ Processes are usually composed of long sequences of events.
- ▶ CER patterns are shorter, may involve Kleene-star, iteration operators (usually not present in processes) and may even be instantaneous.
- ▶ Process prediction focuses on traces, which are complete, full matches.
- ▶ CER focuses on continuously evolving, highly imbalanced streams which may contain many irrelevant events.
- ▶ [VDA11, MRR18, FGMM18]

Complex Event Forecasting

- ▶ First concrete attempt at CEF: [MLJ10]
- ▶ A variant of regular expressions used to define CE patterns.
- ▶ Compiled into automata.
- ▶ Automata translated to Markov chains through a direct mapping.
- ▶ Frequency counters on the transitions used to estimate transition matrix.
- ▶ In the worst case, such an approach assumes that all SDEs are independent.

Complex Event Forecasting

- ▶ [ACMZ15]
- ▶ Support Vector Regression to predict the next input event(s) within some future window.
- ▶ Targets the prediction of the (numerical) values of the attributes of the input events (SDE forecasting).

Complex Event Forecasting

- ▶ [PNC11]
- ▶ HMMs used to construct a probabilistic model for the behavior of a transition system describing a CE.
- ▶ HMMs are hard to train ([BEY04, AW92])
- ▶ Require elaborate domain modeling.

Complex Event Forecasting

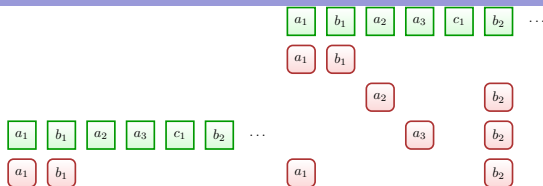
- ▶ [LGC20]
- ▶ Knowledge graphs used to encode events and their timing relationships.
- ▶ More like SDE forecasting, as it does not target complex events.

Forecasting definition

- ▶ “Forecasting” and “prediction” used interchangeably as equivalent terms? No.
- ▶ Prediction in ML
 - ▶ Goal: “predict” the output of a function on previously unseen input data.
 - ▶ Input data need not necessarily have a temporal dimension.
 - ▶ “Prediction” refers to the output of the learned function on a new data point.
- ▶ We need
 - ▶ to predict the temporally future output of some function or the occurrence of an event.
 - ▶ From the (current) timepoint where a forecast is produced until the (future) timepoint for which we try to make a forecast, no data is available.
- ▶ We prefer the term “forecasting”.

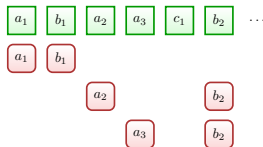
What is Complex Event Forecasting?

- ▶ Input event forecasting: predict the most probable next input event in the stream (akin to next symbol prediction). Not very useful in itself.
- ▶ Complex event forecasting (regression): given the occurrence timestamp of a CE, generate forecasts about this timestamp k events beforehand.
- ▶ Complex event forecasting (classification): at an estimated distance $p\%$ before a “possible” CE occurrence, decide whether a CE will occur within the next k events.

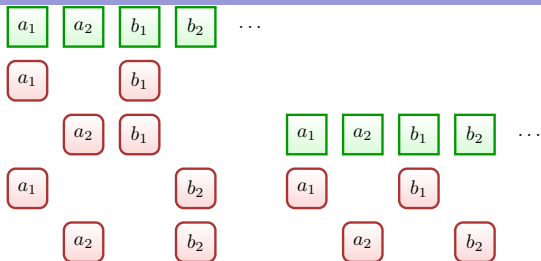


(a) Matches under
strict-contiguity.

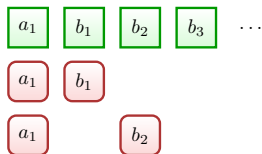
(b) Matches under
skip-till-any-match.



(c) Matches under
skip-till-next-match.

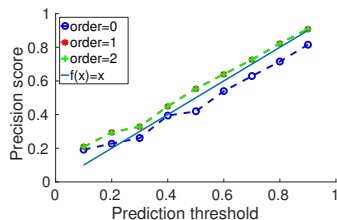


(a) Matches under reuse. (b) Matches under consume.

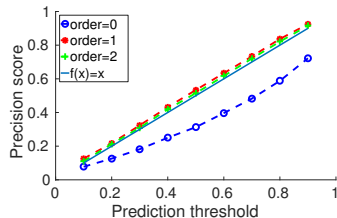


(c) Matches under bounded-reuse.

Validation tests

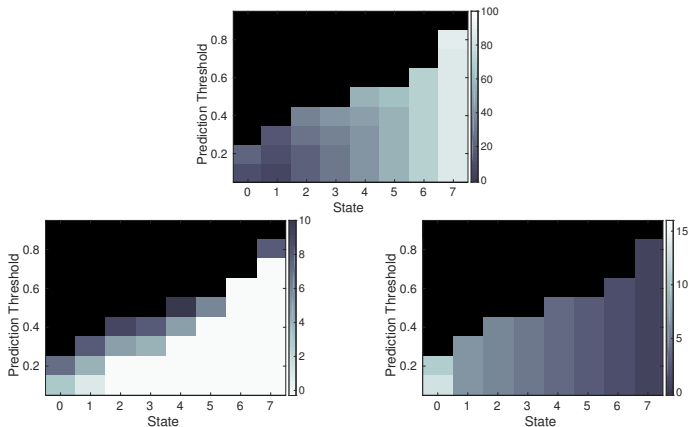


$$(a) \quad R = a \cdot (a + b)^* \cdot c$$

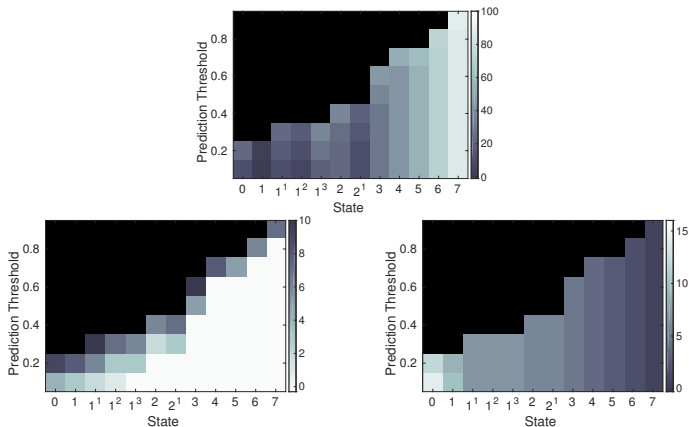


$$(b) \quad R = a \cdot b \cdot c$$

Credit Card Fraud Management: Real Dataset ($m = 1$).



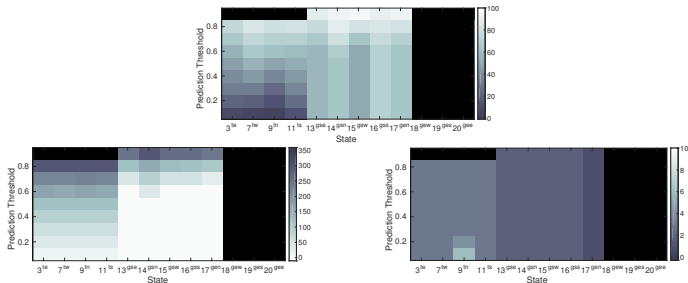
Credit Card Fraud Management: Real Dataset ($m = 3$).

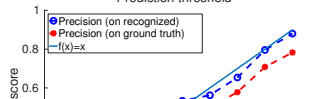
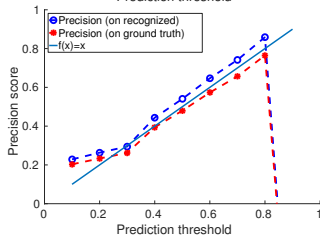
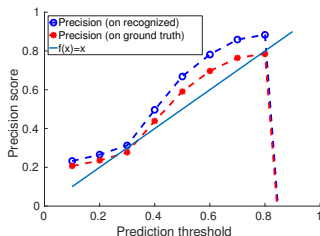


Maritime Monitoring: Real Dataset.

$$R = \text{Turn} \cdot \text{GapStart} \cdot \text{GapEnd} \cdot \text{Turn},$$

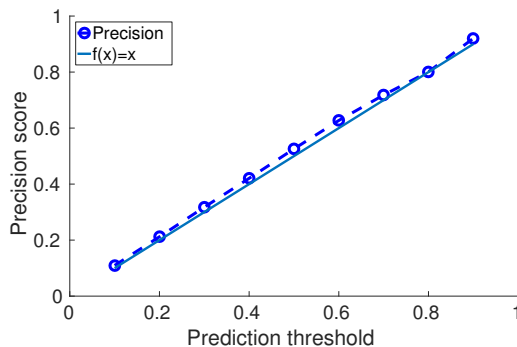
where $\text{Turn} = (\text{TurnNorth} + \text{TurnEast} + \text{TurnSouth} + \text{TurnWest})$



Credit cards (precision for $m = 1, 2, 3$)

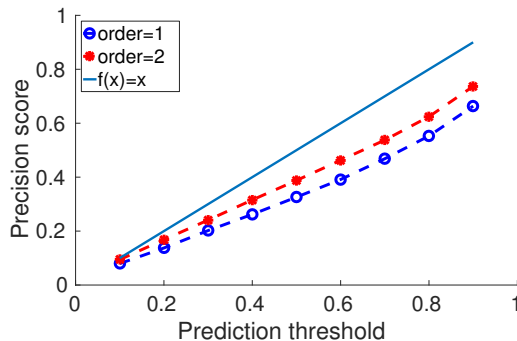
Maritime (precision)

$$R = Turn \cdot GapStart \cdot GapEnd \cdot Turn$$

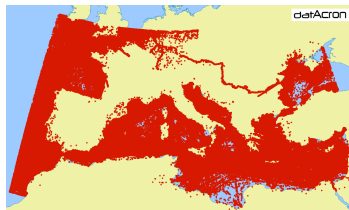
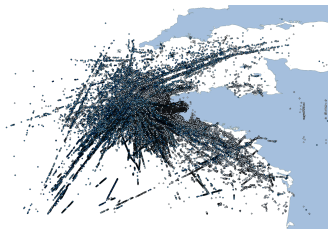


Maritime (precision)

$$R = TurnNorth \cdot (TurnNorth + TurnEast)^* \cdot TurnSouth$$



Datasets



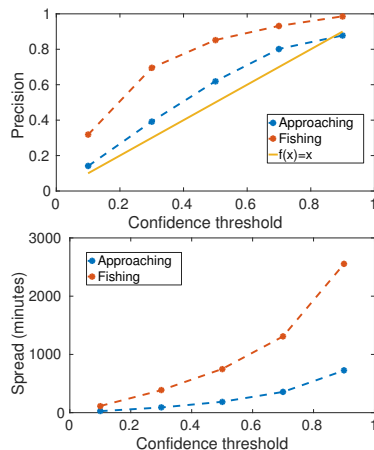
- ▶ AIS messages from the Atlantic Ocean around Brest (1 month) and from most of European seas (6 months).
- ▶ $\approx 1.3\text{M}$ points for Brest, $\approx 2.4\text{M}$ points for Europe (after cleaning).

Patterns: Approaching Port, Fishing

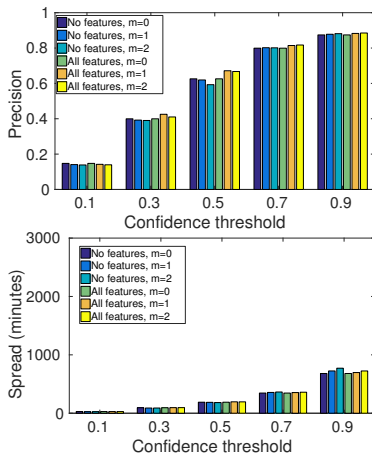
$R_{approach} := x \cdot y^+ \cdot z$ WHERE
 $Distance(x, PortCoords, 7.0, 10.0)$ AND
 $Distance(y, PortCoords, 5.0, 7.0)$ AND
 $WithinCircle(z, PortCoords, 5.0)$
PARTITION BY *vesselId*

$R_{fish} := x \cdot y^* \cdot z$ WHERE
 $(IsFishingVessel(x) \wedge \neg InArea(x, FishingArea))$ AND
 $(InArea(y, FishingArea) \wedge SpeedBetween(y, 9.0, 20.0))$ AND
 $(InArea(z, FishingArea) \wedge SpeedBetween(z, 1.0, 9.0))$
PARTITION BY *vesselId*

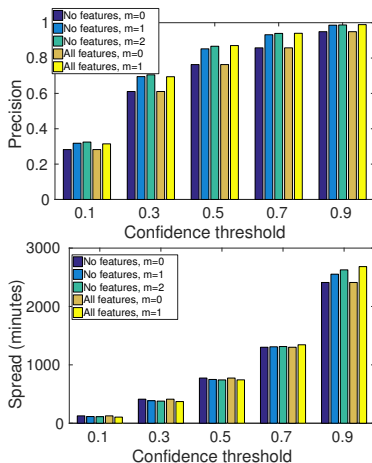
Empirical Evaluation



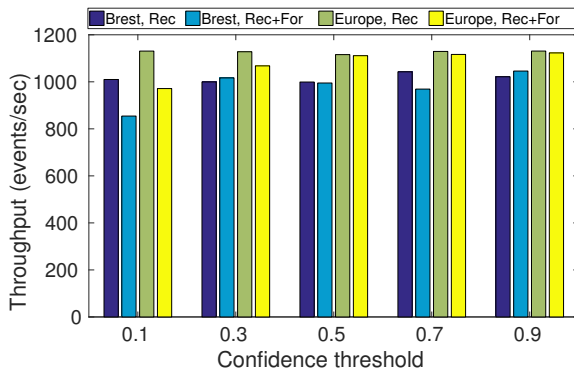
Empirical Evaluation (detailed): approaching



Empirical Evaluation (detailed): fishing



Empirical Evaluation: throughput



Symbolic regular expression (1)

Definition (Symbolic regular expression)

A symbolic regular expression (*SRE*) over an effective Boolean algebra $(\mathcal{D}, \Psi, \llbracket - \rrbracket, \perp, \top, \vee, \wedge, \neg)$ is recursively defined as follows:

- ▶ The constants ϵ and \emptyset are symbolic regular expressions with $\mathcal{L}(\epsilon) = \{\epsilon\}$ and $\mathcal{L}(\emptyset) = \{\emptyset\}$;
- ▶ If $\psi \in \Psi$, then $R := \psi$ is a symbolic regular expression, with $\mathcal{L}(\psi) = \llbracket \psi \rrbracket$, i.e., the language of ψ is the subset of \mathcal{D} for which ψ evaluates to TRUE;

Symbolic regular expression (2)

Definition (Symbolic regular expression)

A symbolic regular expression (*SRE*) over an effective Boolean algebra $(\mathcal{D}, \Psi, \llbracket - \rrbracket, \perp, \top, \vee, \wedge, \neg)$ is recursively defined as follows:

- ▶ Disjunction / Union If R_1 and R_2 are symbolic regular expressions, then $R := R_1 + R_2$ is also a symbolic regular expression, with $\mathcal{L}(R) = \mathcal{L}(R_1) \cup \mathcal{L}(R_2)$;
- ▶ Concatenation / Sequence If R_1 and R_2 are symbolic regular expressions, then $R := R_1 \cdot R_2$ is also a symbolic regular expression, with $\mathcal{L}(R) = \mathcal{L}(R_1) \cdot \mathcal{L}(R_2)$, where \cdot denotes concatenation. $\mathcal{L}(R)$ is then the set of all strings constructed from concatenating each element of $\mathcal{L}(R_1)$ with each element of $\mathcal{L}(R_2)$;

Symbolic regular expression (3)

Definition (Symbolic regular expression)

A symbolic regular expression (*SRE*) over an effective Boolean algebra $(\mathcal{D}, \Psi, \llbracket - \rrbracket, \perp, \top, \vee, \wedge, \neg)$ is recursively defined as follows:

- Iteration / Kleene-star If R is a symbolic regular expression, then $R' := R^*$ is a symbolic regular expression, with $\mathcal{L}(R^*) = (\mathcal{L}(R))^*$, where $\mathcal{L}^* = \bigcup_{i \geq 0} \mathcal{L}^i$ and \mathcal{L}^i is the concatenation of \mathcal{L} with itself i times.

Expressive power of symbolic regular expressions

- ▶ Negation: $R' := !R$, can be supported (*SFA* closed under complement).
- ▶ Conjunction: $R := R_1 \wedge R_2 := (R_1 \cdot R_2) + (R_2 \cdot R_1)$.
- ▶ skip-till-any-match selection policy: If R_1, R_2, \dots, R_n are symbolic regular expressions, then $R' := \#(R_1, R_2, \dots, R_n)$ is a symbolic regular expression, with $R' := R_1 \cdot \top^* \cdot R_2 \cdot \top^* \dots \top^* \cdot R_n$.
- ▶ skip-till-next-match selection policy: If R_1, R_2, \dots, R_n are symbolic regular expressions, then $R' := @(R_1, R_2, \dots, R_n)$ is a symbolic regular expression, with $R' := R_1 \cdot !(\top^* \cdot R_2 \cdot \top^*) \cdot R_2 \dots !(\top^* \cdot R_n \cdot \top^*) \cdot R_n$.

Symbolic finite automaton

Definition (Symbolic finite automaton)

A symbolic finite automaton (*SFA*) is a tuple $M = (\mathcal{A}, Q, q^s, Q^f, \Delta)$, where

- ▶ \mathcal{A} is an effective Boolean algebra;
- ▶ Q is a finite set of states;
- ▶ $q^s \in Q$ is the initial state;
- ▶ $Q^f \subseteq Q$ is the set of final states;
- ▶ $\Delta \subseteq Q \times \Psi_{\mathcal{A}} \times Q$ is a finite set of transitions.

Language of *SFA*

Definition

A string $w = a_1 a_2 \cdots a_k$ is accepted by a *SFA* M iff, for $1 \leq i \leq k$, there exist transitions $q_{i-1} \xrightarrow{a_i} q_i$ such that $q_0 = q^s$ and $q_k \in Q^f$.

We refer to the set of strings accepted by M as the language of M , denoted by $\mathcal{L}(M)$.

Definition (Detection of *SRE*)

If $S_{1..k} = \dots, t_{k-1}, t_k$ is the prefix of S up to the index k , we say that an instance of a *SRE* R is detected at k iff there exists a suffix $S_{m..k}$ of $S_{1..k}$ such that $S_{m..k} \in \mathcal{L}(R)$.

In order to detect CEs of a *SRE* R on a stream, we use a streaming version of *SRE* and *SFA*.

Definition (Streaming *SRE* and *SFA*)

If R is a *SRE*, then $R_s = \top^* \cdot R$ is called the streaming *SRE* (*sSRE*) corresponding to R . A *SFA* M_{R_s} constructed from R_s is called a streaming *SFA* (*sSFA*) corresponding to R .

As an example, if $R := (speed < 5) \cdot (speed > 20)$ is the pattern for sudden acceleration, then its $sSRE$ would be $R_s := \top^* \cdot (speed < 5) \cdot (speed > 20)$. After reading the fourth event of the stream of Table 1, $S_{1..4}$ would belong to the language of $\mathcal{L}(R_s)$ and $S_{3..4}$ to the language of $\mathcal{L}(R)$. Figure 6b shows an example $sSFA$.

Configuration

The streaming behavior of a *sSFA* as it consumes a stream S can be formally defined using the notion of configuration:

Definition (Configuration of sSFA)

Assume $S = t_1, t_2, \dots$ is a stream of domain elements from an effective Boolean algebra, R a symbolic regular expression over the same algebra and M_{R_s} a *sSFA* corresponding to R . A configuration c of M_{R_s} is a tuple $[i, q]$, where i is the current position of the stream, i.e., the index of the next event to be consumed, and q the current state of M_{R_s} . We say that $c' = [i', q']$ is a successor of c iff:

- ▶ $\exists \delta \in M_{R_s}.\Delta : \delta = (q, \psi, q') \wedge (t_i \in \llbracket \psi \rrbracket \vee \psi = \epsilon)$;
- ▶ $i = i'$ if $\delta = \epsilon$. Otherwise, $i' = i + 1$.

We denote a succession by $[i, q] \xrightarrow{\delta} [i', q']$.

Run

The actual behavior of a *sSFA* upon reading a stream is captured by the notion of the run:

Definition (Run of *sSFA* over stream)

A run ϱ of a *sSFA* M over a stream $S_{1..k}$ is a sequence of successor configurations $[1, q_1 = M.q^s] \xrightarrow{\delta_1} [2, q_2] \xrightarrow{\delta_2} \dots \xrightarrow{\delta_k} [k+1, q_{k+1}]$. A run is called accepting iff $q_{k+1} \in M.Q^f$.

A run ϱ of a *sSFA* M_{R_s} over a stream $S_{1..k}$ is accepting iff $S_{1..k} \in \mathcal{L}(R_s)$, since M_{R_s} , after reading $S_{1..k}$, must have reached a final state. Therefore, for a *sSFA* that consumes a stream, the existence of an accepting run with configuration index $k+1$ implies that a CE for the *SRE* R has been detected at the stream index k .

Deterministic SFA

Definition (Deterministic SFA [DV17])

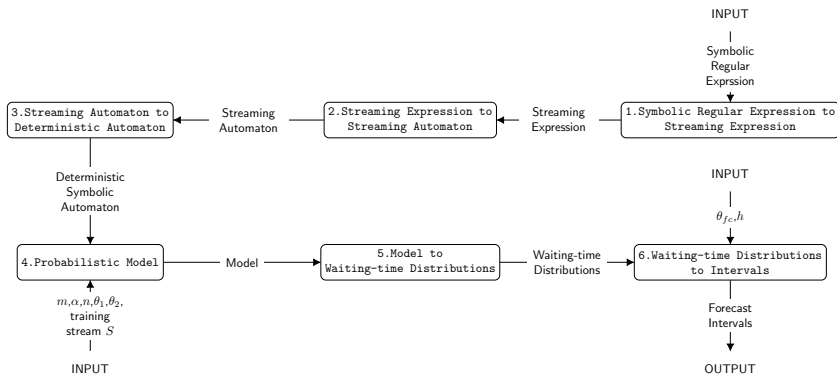
A SFA M is deterministic if, for all transitions

$(q, \psi_1, q_1), (q, \psi_2, q_2) \in M.\Delta$, if $q_1 \neq q_2$ then $\llbracket \psi_1 \wedge \psi_2 \rrbracket = \emptyset$.

Determinization of *SFA*

- ▶ First create the *Minterms*.
- ▶ Assume $Predicates(M) = \{\psi_1, \psi_2\}$.
- ▶ Then $Minterms(Predicates(M)) = \{\psi_1 \wedge \psi_2, \psi_1 \wedge \neg\psi_2, \neg\psi_1 \wedge \psi_2, \neg\psi_1 \wedge \neg\psi_2\}$.
- ▶ Map each tuple to exactly one of these 4 minterms: the one that evaluates to TRUE when applied to the element.
- ▶ Assume $S = t_1, \dots, t_k$ is an event stream.
- ▶ Can be mapped to $S' = a, \dots, b$.
- ▶ a corresponds to $\psi_1 \wedge \neg\psi_2$ if $\psi_1(t_1) \wedge \neg\psi_2(t_1) = \text{TRUE}$, b to $\psi_1 \wedge \psi_2$, etc.

Workflow



Learning a PST

- ▶ Incrementally learn a $PST \hat{T}$ by adding new nodes only when it is necessary.
- ▶ Start with a tree having only a single node, corresponding to the empty string ϵ .
- ▶ Decide whether to add a new context/node s by checking
 - ▶ First, there must exist $\sigma \in \Sigma$ such that $\hat{P}(\sigma | s) > \theta_1$ must hold, i.e., σ must appear “often enough” after the suffix s ;
 - ▶ Second, $\frac{\hat{P}(\sigma|s)}{\hat{P}(\sigma|suffix(s))} > \theta_2$ (or $\frac{\hat{P}(\sigma|s)}{\hat{P}(\sigma|suffix(s))} < \frac{1}{\theta_2}$) must hold, i.e., it is “meaningful enough” to expand to s because there is a significant difference in the conditional probability of σ given s with respect to the same probability given the shorter context $suffix(s)$, where $suffix(s)$ is the longest suffix of s that is different from s .

Embedding

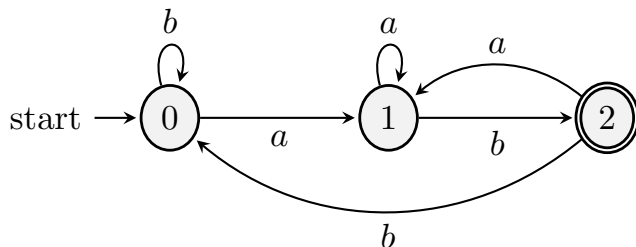


Figure: DSFA M_R for $R := a \cdot b$ and $\Sigma = \{a, b\}$.

Embedding

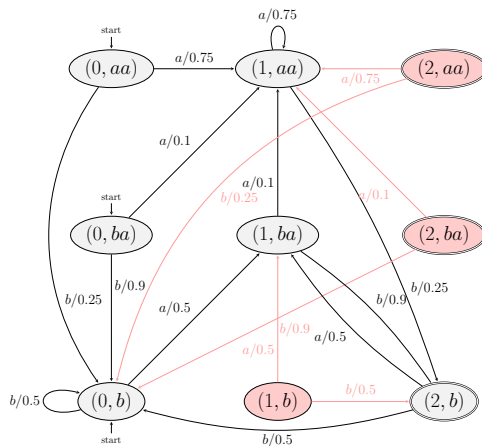


Figure: Embedding of M_S in M_R .

Transition matrix

$$\Pi = \begin{pmatrix} \mathbf{N} & \mathbf{N}_F \\ \mathbf{F}_N & \mathbf{F} \end{pmatrix}$$

where

- ▶ \mathbf{N} is the sub-matrix containing the probabilities of transitions from non-final to non-final states,
- ▶ \mathbf{F} the probabilities from final to final states,
- ▶ \mathbf{F}_N the probabilities from final to non-final states and
- ▶ \mathbf{N}_F the probabilities from non-final to final states.

Reaching a final state

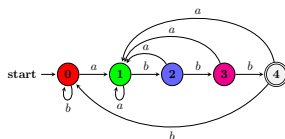
Theorem ([FL03])

Let ξ_{init} be the initial state distribution. The probability for the time index n when the system first enters the set of states F , starting from a state in N , can be obtained from

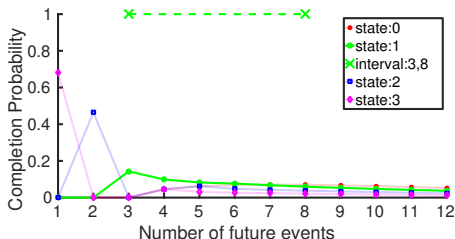
$$P(Y_n \in F, Y_{n-1} \in N, \dots, Y_2 \in N, Y_1 \in N \mid \xi_{init}) = \xi_N^T \mathbf{N}^{n-1} (\mathbf{I} - \mathbf{N}) \mathbf{1}$$

where ξ_N is the vector consisting of the elements of ξ_{init} corresponding to the states of N .

Waiting-time distributions

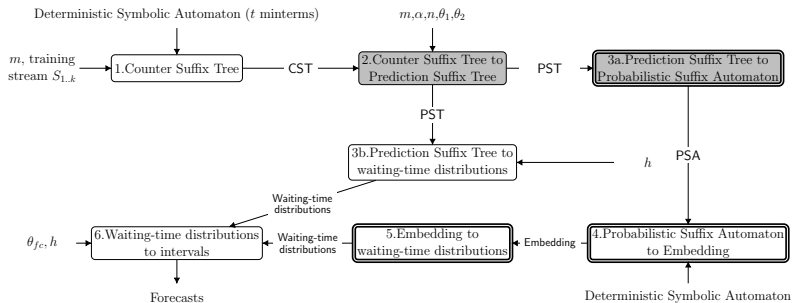


(a) DFA.



(b) Waiting-time distributions and shortest interval, i.e. $[3, 8]$, exceeding a confidence threshold $\theta_{fc} = 50\%$ for state 1.

Flow



Forecasting types

- ▶ *REGRESSION-ARGMAX*: select the future point with the highest probability.
- ▶ *CLASSIFICATION-NEXTW*: how likely it is that a CE will occur within the next w input events.
- ▶ *REGRESSION-INTERVAL*: select the shortest possible interval I above a threshold θ_{fc} .

Evaluation

- ▶ In CE forecasting, emitting a forecast after every new SDE is feasible in principle, but not very useful and can also produce results that are misleading.
- ▶ CEs are relatively rare within a stream of input SDEs.
- ▶ If we emit a forecast after every new SDE, some of these forecasts (possibly even the vast majority) will have a significant temporal distance from the CE to which they refer.
- ▶ The scores and metrics that we use to evaluate the quality of the forecasts will be dominated by these, necessarily low-quality, distant forecasts.
- ▶ We need to establish checkpoints where forecasts are allowed to be emitted.

Checkpoints

- ▶ In regression experiments, CEs provide a natural point of reference.
- ▶ In classification experiments, we can use the structure of the automaton itself.
- ▶ For an automaton that is in a final state, the “process” which it describes has been completed or, equivalently, that there remains 0% of the process until completion.
- ▶ A 100% distance refers to the state(s) that are the most distant to a final state.
- ▶ Distances of all other states through shortest paths.
- ▶ We may establish checkpoints by allowing only states with a distance between 40% and 60% to emit forecasts.

Metrics

- ▶ Each forecast is evaluated:
 - ▶ a) as a *true positive* (TP) if the forecast is positive and the CE does indeed occur within the next w events from the forecast;
 - ▶ b) as a *false positive* (FP) if the forecast is positive and the CE does not occur;
 - ▶ c) as a *true negative* (TN) if the forecast is negative and the CE does not occur and
 - ▶ d) as a *false negative* (FN) if the forecast is negative and the CE does occur;

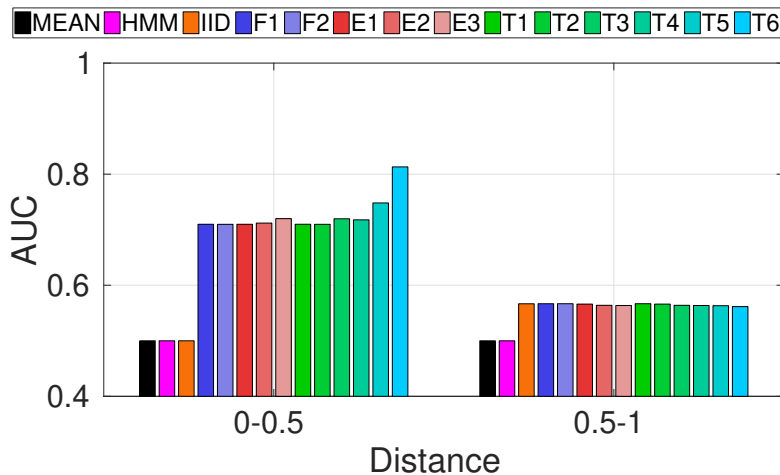


Figure: AUC for ROC curves. Extra features included: concentric rings around the port every 3 km. Single vessel.

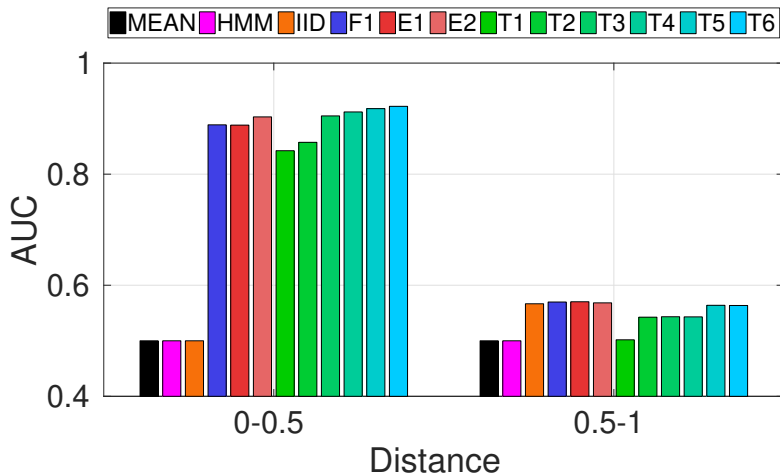


Figure: AUC for ROC curves. Extra features included: concentric rings around the port every 1 km and heading. Single vessel.

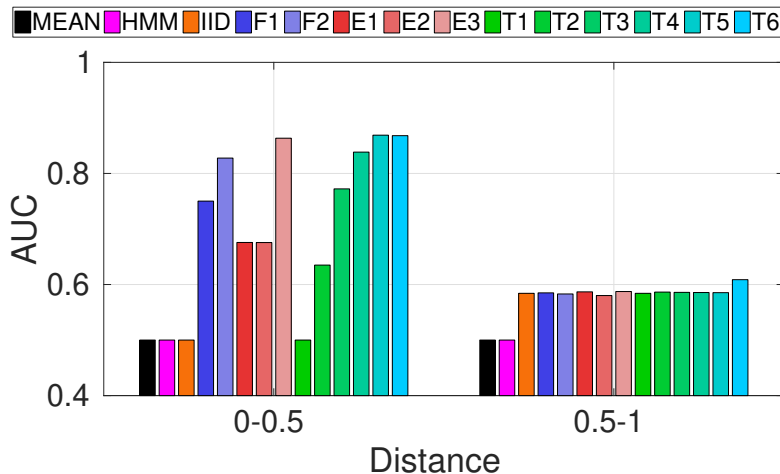


Figure: AUC for ROC curves. Extra features included: concentric rings around the port every 1 km. Model constructed for the 9 vessels that have more than 100 matches.

Definition (Semantics of *SREM*)

Let e be a *SREM* over a \mathcal{V} -structure \mathcal{M} and a set of register variables $R = \{r_1, \dots, r_k\}$, S a string constructed from elements of the universe of \mathcal{M} and $v, v' \in F(r_1, \dots, r_k)$. We define the relation $(e, S, v) \vdash v'$ as follows:

1. $(\epsilon, S, v) \vdash v'$ iff $S = \epsilon$ and $v = v'$.
2. $(\phi, S, v) \vdash v'$ iff $\phi \neq \epsilon$, $S = u$, $(u, v) \models \phi$ and $v' = v$.
3. $(\phi \downarrow r_i, S, v) \vdash v'$ iff $S = u$, $(u, v) \models \phi$ and $v' = v[r_i \leftarrow u]$.
4. $(e_1 \cdot e_2, S, v) \vdash v'$ iff $S = S_1 \cdot S_2$: $(e_1, S_1, v) \vdash v''$ and $(e_2, S_2, v'') \vdash v'$.
5. $(e_1 + e_2, S, v) \vdash v'$ iff $(e_1, S, v) \vdash v'$ or $(e_2, S, v) \vdash v'$.
6. $(e^*, S, v) \vdash v'$ iff

$$\begin{cases} S = \epsilon \text{ and } v' = v \\ S = S_1 \cdot S_2 : (e, S_1, v) \vdash v'' \text{ and } (e^*, S_1, v'') \vdash v' \end{cases} \quad \text{or}$$

Definition (Language accepted by a *SREM*)

We say that (e, S, v) infers v' if $(e, S, v) \vdash v'$. We say that e induces v on a string S if $(e, S, \#) \vdash v$, where $\#$ denotes the valuation in which no $v(r_i)$ is defined, i.e., all registers are empty. The language accepted by a *SREM* e is defined as $\mathcal{L}(e) = \{S \mid (e, S, \#) \vdash v \text{ for some valuation } v\}$. ◀

References I



Elias Alevizos, Alexander Artikis, and Georgios Paliouras, *Event forecasting with pattern markov chains*, DEBS, ACM, 2017, pp. 146–157.



_____, *Wayeb: a tool for complex event forecasting*, LPAR, EPIc Series in Computing, vol. 57, EasyChair, 2018, pp. 26–35.







_____, *Complex event forecasting with prediction suffix trees*, VLDB J. (2021).



Adnan Akbar, François Carrez, Klaus Moessner, and Ahmed Zoha, *Predicting complex events for pro-active iot applications*, WF-IoT, IEEE Computer Society, 2015, pp. 327–332.

References II

-  Elias Alevizos, Anastasios Skarlatidis, Alexander Artikis, and Georgios Paliouras, *Probabilistic complex event recognition: A survey*, ACM Comput. Surv. **50** (2017), no. 5, 71:1–71:31.
-  Naoki Abe and Manfred K. Warmuth, *On the computational complexity of approximating distributions by probabilistic automata*, Machine Learning **9** (1992), 205–260.
-  Ron Begleiter, Ran El-Yaniv, and Golan Yona, *On prediction using variable order markov models*, J. Artif. Intell. Res. **22** (2004), 385–421.
-  Peter Bühlmann, Abraham J Wyner, et al., *Variable length markov chains*, The Annals of Statistics **27** (1999), no. 2, 480–513.

References III



Maximilian Christ, Julian Krumeich, and Andreas W. Kempa-Liehr, *Integrating predictive analytics into complex event processing by using conditional density estimations*, EDOC Workshops, IEEE Computer Society, 2016, pp. 1–8.







Buru Chang, Yonggyu Park, Donghyeon Park, Seongsoon Kim, and Jaewoo Kang, *Content-aware hierarchical point-of-interest embedding model for successive POI recommendation*, IJCAI, ijcai.org, 2018, pp. 3301–3307.






John G. Cleary and Ian H. Witten, *Data compression using adaptive coding and partial string matching*, IEEE Trans. Communications **32** (1984), no. 4, 396–402.




References IV

-  Chung-Wen Cho, Yi-Hung Wu, Show-Jane Yen, Ying Zheng, and Arbee L. P. Chen, *On-line rule matching for event prediction*, VLDB J. **20** (2011), no. 3, 303–334.
-  Loris D'Antoni and Margus Veanes, *The power of symbolic automata and transducers*, CAV (1), Lecture Notes in Computer Science, vol. 10426, Springer, 2017, pp. 47–67.
-  Yagil Engel and Opher Etzion, *Towards proactive event-driven computing*, DEBS, ACM, 2011, pp. 125–136.
-  Lina Fahed, Armelle Brun, and Anne Boyer, *Efficient discovery of episode rules with a minimal antecedent and a distant consequent*, IC3K (Selected Papers), Communications in Computer and Information Science, vol. 553, Springer, 2014, pp. 3–18.

References V

-  Lajos Jeno Fülöp, Árpád Beszédes, Gabriella Toth, Hunor Demeter, László Vidács, and Lóránt Farkas, *Predictive complex event processing: a conceptual framework for combining complex event processing and predictive analytics*, BCI, ACM, 2012, pp. 26–31.
-  Chiara Di Francescomarino, Chiara Ghidini, Fabrizio Maria Maggi, and Fredrik Milani, *Predictive process monitoring methods: Which one suits me best?*, BPM, Lecture Notes in Computer Science, vol. 11080, Springer, 2018, pp. 462–479.
-  James C Fu and WY Wendy Lou, *Distribution theory of runs and patterns and its applications: a finite markov chain imbedding approach*, World Scientific, 2003.

References VI

-  Nikos Giatrakos, Elias Alevizos, Alexander Artikis, Antonios Deligiannakis, and Minos N. Garofalakis, *Complex event recognition in the big data era: a survey*, VLDB J. **29** (2020), no. 1, 313–352.
-  Zhongyang Li, Xiao Ding, and Ting Liu, *Constructing narrative event evolutionary graph for script event prediction*, IJCAI, ijcai.org, 2018, pp. 4201–4207.
-  Yan Li, Tingjian Ge, and Cindy X. Chen, *Data stream event prediction based on timing knowledge and state transitions*, Proc. VLDB Endow. **13** (2020), no. 10, 1779–1792.
-  Srivatsan Laxman, Vikram Tankasali, and Ryen W. White, *Stream prediction using a generative model based on frequent episodes in event sequences*, KDD, ACM, 2008, pp. 453–461.

References VII

-  Douglas C Montgomery, Cheryl L Jennings, and Murat Kulahci, *Introduction to time series analysis and forecasting*, John Wiley & Sons, 2015.
-  Vinod Muthusamy, Haifeng Liu, and Hans-Arno Jacobsen, *Predictive publish/subscribe matching*, DEBS, ACM, 2010, pp. 14–25.
-  Alfonso Eduardo Márquez-Chamorro, Manuel Resinas, and Antonio Ruiz-Cortés, *Predictive monitoring of business processes: A survey*, IEEE Trans. Services Computing **11** (2018), no. 6, 962–977.

References VIII



Emmanouil Ntoulas, Elias Alevizos, Alexander Artikis, Charilaos Akasiadis, and Athanasios Koumparos, *Online trajectory analysis with scalable event recognition*, Geoinformatica (2022).






Emmanouil Ntoulas, Elias Alevizos, Alexander Artikis, and Athanasios Koumparos, *Online trajectory analysis with scalable event recognition*, EDBT/ICDT Workshops, CEUR Workshop Proceedings, vol. 2841, CEUR-WS.org, 2021.



Suraj Pandey, Surya Nepal, and Shiping Chen, *A test-bed for the evaluation of business process prediction techniques*, CollaborateCom, ICST / IEEE, 2011, pp. 382–391.

References IX

-  Ehab Qadah, Michael Mock, Elias Alevizos, and Georg Fuchs, *A distributed online learning approach for pattern prediction over movement event streams with apache flink*, EDBT/ICDT Workshops, CEUR Workshop Proceedings, vol. 2083, CEUR-WS.org, 2018, pp. 109–116.
-  Dana Ron, Yoram Singer, and Naftali Tishby, *The power of amnesia*, NIPS, Morgan Kaufmann, 1993, pp. 176–183.
-  ———, *The power of amnesia: Learning probabilistic automata with variable memory length*, Machine Learning **25** (1996), no. 2-3, 117–149.

References X



Marios Voudas, Konstantina Bereta, Dimitris Kladis, Dimitris Zissis, Elias Alevizos, Emmanouil Ntoulas, Alexander Artikis, Antonios Deligiannakis, Antonios Kontaxakis, Nikos Giatrakos, David Arnu, Edwin Yaqub, Fabian Temme, Mate Torok, and Ralf Klinkenberg, *Online distributed maritime event detection and forecasting over big vessel tracking data*, IEEE BigData, IEEE Computer Society, 2021.



Wil Van Der Aalst, *Process mining: discovery, conformance and enhancement of business processes*, Springer, 2011.



Ricardo Vilalta and Sheng Ma, *Predicting rare events in temporal domains*, ICDM, IEEE Computer Society, 2002, pp. 474–481.

References XI



George A. Vouros, Akrivi Vlachou, Georgios M. Santipantakis, Christos Doulkeridis, Nikos Pelekis, Harris V. Georgiou, Yannis Theodoridis, Kostas Patroumpas, Elias Alevizos, Alexander Artikis, Christophe Claramunt, Cyril Ray, David Scarlatti, Georg Fuchs, Gennady L. Andrienko, Natalia V. Andrienko, Michael Mock, Elena Camossi, Anne-Laure Joussetme, and Jose Manuel Cordero Garcia, *Big data analytics for time critical mobility forecasting: Recent progress and research challenges*, EDBT, OpenProceedings.org, 2018, pp. 612–623.

References XII



George A. Vouros, Akrivi Vlachou, Georgios M. Santipantakis, Christos Doulkeridis, Nikos Pelekis, Harris V. Georgiou, Yannis Theodoridis, Kostas Patroumpas, Elias Alevizos, Alexander Artikis, Georg Fuchs, Michael Mock, Gennady L. Andrienko, Natalia V. Andrienko, Christophe Claramunt, Cyril Ray, Elena Camossi, and Anne-Laure Joussetme, *Increasing maritime situation awareness via trajectory detection, enrichment and recognition of events*, W2GIS, Lecture Notes in Computer Science, vol. 10819, Springer, 2018, pp. 130–140.



Frans M. J. Willems, Yuri M. Shtarkov, and Tjalling J. Tjalkens, *The context-tree weighting method: basic properties*, IEEE Trans. Information Theory **41** (1995), no. 3, 653–664.

References XIII



Cheng Zhou, Boris Cule, and Bart Goethals, *A pattern based predictor for event streams*, Expert Syst. Appl. **42** (2015), no. 23, 9294–9306.