Complex Symbolic Sequence Encodings for Predictive Monitoring of Business Processes

Anna Leontjeva[1], Raffaele Conforti[2], Chiara Di Francescomarino[3], Marlon Dumas[1], and Fabrizio Maria Maggi[1]

[1] University of Tartu, Estonia.
{anna.leontjeva, marlon.dumas, f.m.maggi}@ut.ee
[2] Queensland University of Technology, Australia.
raffaele.conforti@qut.edu.au
[3] FBK-IRST, Italy.
dfmchiara@fbk.eu

Abstract. This paper addresses the problem of predicting the outcome of an ongoing case of a business process based on event logs. In this setting, the outcome of a case may refer for example to the achievement of a performance objective or the fulfillment of a compliance rule upon completion of the case. Given a log consisting of traces of completed cases, given a trace of an ongoing case, and given two or more possible outcomes (e.g., a positive and a negative outcome), the paper addresses the problem of determining the most likely outcome for the case in question. Previous approaches to this problem are largely based on simple symbolic sequence classification, meaning that they extract features from traces seen as sequences of event labels, and use these features to construct a classifier for runtime prediction. In doing so, these approaches ignore the data payload associated to each event. This paper approaches the problem from a different angle by treating traces as complex symbolic sequences, that is, sequences of events each carrying a data payload. In this context, the paper outlines different feature encodings of complex symbolic sequences and compares their predictive accuracy on real-life business process event logs.

Keywords: Process Mining, Predictive Monitoring, Complex Symbolic Sequence

1 Introduction

Process mining is a family of methods for analyzing business processes based on *event logs* consisting of *traces*, each representing one execution of the process (a.k.a. a *case*). A trace consists of a sequence of (possibly timestamped) events, each referring to an execution of an activity (a.k.a. an *event class*). Events in a trace may have a payload consisting of attributes such as the resource(s) involved in the execution of an activity or other data recorded with the event.

Predictive business process monitoring [14] is a category of process mining methods that aims at predicting at runtime and as early as possible the outcome of a case given its current (incomplete) trace. In this context, an outcome may be the fulfillment of a constraint on the cycle time of the case, the validity of a temporal logic constraint,

or any predicate over a completed case. For example, in a sales process, a possible outcome might be the placement of a purchase order by a potential customer, whereas in a medical treatment process, a possible outcome is the recovery of the patient upon completion of the treatment.

Existing approaches to predictive monitoring [14, 7] essentially map the problem to that of early sequence classification [24]. The idea is to train a classifier over the set of prefixes of historical traces. This classifier is used at runtime in order to predict the outcome of an ongoing case based on its current (incomplete) trace. A key step is to extract features from prefixes of historical traces. In this respect, existing approaches treat traces as simple symbolic sequences, meaning sequences of symbols, each representing an event but without its payload. When data is taken into account, only the latest payload of data attributes attached to the event at the end of each trace prefix is included in the feature vector of the classifier, but the evolution of data attributes as the case unfolds is ignored.

This paper investigates an alternative approach where traces are treated as complex symbolic sequences, that is, sequences of events each carrying a data payload consisting of attribute-value pairs. A crucial design choice in this approach is how to encode a complex symbolic sequence in terms of vectors of features. In this respect, the paper proposes two complex sequence encodings. The first encoding is based on indexes. This encoding specifies, for each position in the case, the event occurring in that position and the value of each data attribute in that position. The second encoding is obtained by combining the first one with an encoding based on Hidden Markov Models (HMMs), a well-known generative probabilistic technique. As this work deals with the problem of case classification, a discriminative HMM approach is adopted. In particular, separate HMMs are trained for each possible outcome (e.g., one HMM for positive cases and one for negative cases). Then, the likelihood of a trace prefix to belong to each of these two models is measured. The difference in likelihoods is expressed in terms of odds-ratios, which are then used as features to train the classifier. The proposed methods are evaluated in terms of their accuracy at different points in a trace based on two real life logs: (i) a patient treatment log provided for the BPI challenge 2011 [1] and (ii) an insurance claim process log from an insurance company [22].

The paper is structured as follows. Section 2 reviews previous work on predictive business process monitoring and introduces HMMs, which are used later in the paper. Section 3 presents the proposed methods while Section 4 discusses their evaluation. Finally, Section 5 draws conclusions and outlines future work.

2 Background and Related Work

This section provides an overview of existing predictive business process monitoring approaches (Section 2.1) and briefly introduce Hidden Markov Models (HMMs), which we use for complex symbolic sequence encoding (Section 2.2).

2.1 Predictive Monitoring: The Related Work

Existing techniques for predictive business process monitoring can be broadly classified based on the type of predicted outcome. In this respect, a first group of works concen-

trates on the time perspective. In [3, 2], the authors present a set of approaches in which annotated transition systems, containing time information extracted from event logs, are used to: (i) check time conformance while cases are being executed, (ii) predict the remaining processing time of incomplete cases, and (iii) recommend appropriate activities to end users working on these cases. In [10], an ad-hoc predictive clustering approach is presented, in which context-related execution scenarios are discovered and modeled through state-aware performance predictors. In [20], the authors use stochastic Petri nets for predicting the remaining execution time of a process.

A second group of works focuses on approaches that generate predictions and recommendations to reduce risks. For example, in [7], the authors present a technique to support process participants in making risk-informed decisions, with the aim of reducing the process risks. Risks are predicted by traversing decision trees generated from the logs of past process executions. In [16], the authors make predictions about time-related process risks, by identifying (using statistical principles) and exploiting indicators observable in event logs that highlight the possibility of transgressing deadlines. In [21], an approach for Root Cause Analysis through classification algorithms is presented. Decision trees are used to retrieve the causes of overtime faults on a log enriched with information about delays, resources and workload.

An approach for prediction of abnormal termination of business processes is presented in [12]. Here, a fault detection algorithm (local outlier factor) is used to estimate the probability of a fault to occur. Alarms are provided for early notification of probable abnormal terminations. In [6], Castellanos et al. present a business operation management platform equipped with time series forecasting functionalities. This platform allows for predictions of metric values on running process instances as well as for predictions of aggregated metric values of future instances (e.g., the number of orders that will be placed next Monday). Predictive monitoring focused on specific types of failures has also been applied to real case studies. For example, in [15, 8], the authors present a technique for predicting "late show" events in transportation processes. In particular, they apply standard statistical techniques to find correlations between "late show" events and external variables related to weather conditions or road traffic.

A key difference between these approaches and our technique is that they rely either on the control-flow or on the data perspective for making predictions at runtime, whereas we take both perspectives into consideration. The two perspectives have been considered together only in [14], where a framework for the predictive monitoring of constraint fulfillment and violation has been proposed. In this approach, however, only the payload of the last executed event is taken into account, while neglecting the evolution of data values throughout the execution traces. The present paper aims at addressing this latter limitation by treating the input traces as complex symbolic sequences.

2.2 Hidden Markov Models

Hidden Markov Models (HMMs) [18] are a class of well-studied models of sequential observations that have been widely applied in the context of sequence classification [24]. HMMs are probabilistic generative models, meaning that there is an assumption that an observed sequence is generated by some process that needs to be uncovered

via probabilistic reasoning. The idea behind HMM is that a sequence consists of observed events, generated by some hidden factors. Assume, for example, that two coins – a fair one and a biased one – are tossed in some unknown order. Only a sequence of heads and tails can be observed. Our goal is to figure out which parts of the sequence were produced by the fair and which by the biased coin. This process can be described by:

- observed events $O = \{O_1, O_2, ..., O_T\}$ - resulting sequence consisted of heads and tails;
- set of discrete symbols - the finite *alphabet size* $V = \{V_1, V_2, ..V_{|M|}\}$ - {head, tail} in our example;
- number of hidden states N, where each state is denoted as $S = \{S_1, S_2, ..S_{|N|}\}$ - represented by fair and biased coin in our example;
- vector of initial probabilities π - how often, in general, each coin is chosen;
- matrix of emission probabilities B - probabilities for each symbol to occur in a particular hidden state - for example, the probability of tails of the biased coin;
- matrix transition probabilities A - probability to move from one state to another or to stay in the same state - transition probabilities answer the question "how often the coins were switched".

The common HMM construction procedure is to specify parameters N and V and to train a model $\lambda = \{A, B, \pi\}$ using a maximum likelihood method such as the standard Baum-Welch algorithm [18].

3 Predictive Monitoring: The Proposed Approach

In this section, the proposed approach for predictive monitoring is described. In particular, in Section 3.1, an overview of the entire approach is given. In Section 3.2, the core part of the proposed approach is introduced, i.e., the encoding of log cases as complex symbolic sequences.

3.1 Overview

Fig. 1 shows an overview of the proposed approach. To predict the outcome of an ongoing case, its current (incomplete) trace (say of length n) is encoded using complex symbolic sequences. As explained in detail in Section 3.2, a complex symbolic sequence carries information about the control flow and the data flow of the trace.

In the approach, a log of historical (completed) cases is supposed to be available. From these cases, all the prefixes of length n are extracted and, in turn, encoded in the form of complex symbolic sequences. In addition, these sequences are labeled using a binary or categorical value according to their outcome. These "historical complex symbolic sequences" are used to train a classifier. The current ongoing trace is then used to query the classifier that returns the label that is the most probable outcome for the current case according to the information derived from the historical cases. In this work, we use random forest as classifier that belongs to the class of ensemble methods [5]. At the core of the method is the concept of decision tree. However, instead

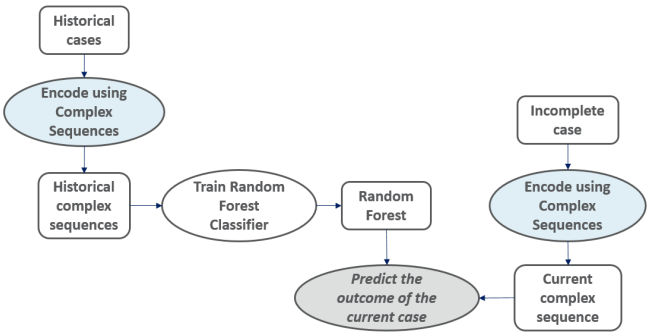

Fig. 1: Overview of the proposed approach.

of training a single tree on a dataset, it grows a pre-defined number of trees and let them vote for the most popular outcome. Random forest is easy to train as it requires less input parameters to tune compared to other classification algorithms.[4] Moreover, it has shown superior results over other well-known classification algorithms like support vector machines (SVM) and generalized boosted regression models (GBM) [23, 19] in several cases [9]. A comparison of the performances of these algorithms when applied to one of the datesets used in this paper is shown in Fig. 8.

3.2 Complex Symbolic Sequence Encodings

Each case of a log corresponds to a sequence σ_i of events describing its control flow. Each event is also associated with data in the form of attribute-value pairs. Moreover, each completed case is associated to an outcome - a *label*, which can assume binary or categorical values. We represent a case in the following form:

sequence(event{associated data},...,event{associated data}): label

As running example, we consider the log in Fig. 2 pertaining to a medical treatment process. Each case relates to a different patient and the corresponding sequence of events indicates the activities executed for a medical treatment of that patient. In the example, *consultation* is the first event of sequence σ_1. Its data payload "$\{33, \text{radiotherapy}\}$" corresponds to the data associated to attributes *age* and *department*. Note that the value of *age* is static: It is the same for all the events in a case, while the value of *department* is different for every event. In the payload of an event, always the entire set of attributes available in the log is considered. In case for some event the value for a specific attribute is not available, the value *unknown* is specified for it.

The goal of predictive business process monitoring is to build a classifier that learns from a set of historical cases L how to discriminate classes of cases and predict as early as possible the outcome of a new, unlabeled case. More specifically, we are interested in automatically deriving a function f that, given an ongoing sequence σ_x provides a

[4] Random forest requires two parameters: Number of trees to grow (ntrees) and number of features to use for each tree (mtry).

σ_1 (consultation{33, radiotherapy},...,ultrasound{33, nursing ward}):false
...
σ_k (order rate{56, general lab},..., payment{56, clinic}):true

Fig. 2: Running example.

label for it, i.e., $f : (L, \sigma_x) \rightarrow \{label_x\}$. To achieve this goal, a random forest classifier is trained on all sequence prefixes of the same length of σ_x derived from historical cases in L. In order to train the classifier, each (prefix) sequence σ_i, $i = 1...k$ has to be represented through a feature vector $g_i = (g_{i1}, g_{i2}, ...g_{ih})$.

In the most straightforward encodings, sequences are treated as simple symbolic sequences, while additional information related to data and data flow is neglected. This work combines and exploits both the control and the data flow dimension by considering the sequences as complex symbolic sequences. In particular, two different encodings (the *index-based* encoding and the *HMM-based* encoding) are taken into consideration. In the following sections, first, four classical baseline encodings are sketched and then the two new encodings are illustrated in detail.

	consultation	ultrasound	...	payment	label
σ_1	1	1	...	0	false
...					
σ_k	0	0	...	1	true

(a) *boolean* encoding.

	consultation	ultrasound	...	payment	label
σ_1	2	1	...	0	false
...					
σ_k	0	0	...	4	true

(b) *frequency-based* encoding.

	event_1	...	event_m	label
σ_1	consultation		ultrasound	false
...				
σ_k	order rate		payment	true

(c) *simple index* encoding.

	age	event_1	...	event_m	...	department_last	label
σ_1	33	consultation		ultrasound	...	nursing ward	false
...							
σ_k	56	order rate		payment	...	clinic	true

(d) *index latest payload* encoding.

Table 1: Baseline encodings for the example in Fig. 2.

3.3 Baselines

The first two approaches we use as baselines in our experiments describe sequences of events as feature vectors, where each feature corresponds to an event class (an activity) from the log. In particular, the *boolean* encoding represents a sequence σ_i through a feature vector $g_i = (g_{i1}, g_{i2}, ...g_{ih})$, where, if g_{ij} corresponds to the event class e, then:

$$g_{ij} = \begin{cases} 1 & \text{if } e \text{ is present in } \sigma_i \\ 0 & \text{if } e \text{ is not present in } \sigma_i \end{cases}$$

For instance, the encoding of the example reported in Fig. 2 with the *boolean* encoding is shown in Table 1a. The *frequency-based* encoding, instead of boolean values, represents the control flow in a case with the frequency of each event class in the case. Table 1b shows the *frequency-based* encoding for the example in Fig. 2.

Another way of encoding a sequence is by taking into account also information about the order in which events occur in the sequence, as in the *simple index* encoding. Here, each feature corresponds to a position in the sequence and the possible values for

each feature are the event classes. By using this type of encoding the example in Fig. 2 would be encoded as reported in Table 1c.

The fourth baseline encoding adds to the simple index baseline the data of the latest payload. Here, data attributes are treated as static features without taking into consideration their evolution over time. Table 1d shows this encoding for the example in Fig. 2.

	age	event_1	...	event_m	...	department_1	...	department_m	label
σ_1	33	consultation		ultrasound		radiotherapy		nursing ward	false
...									
σ_j	56	order rate		payment		general lab		clinic	true

(a) *index-based* encoding.

	age	event_1	...	event_m	...	department_1	...	department_m	LLR_event	...	LLR_department	label
σ_1	33	consultation		ultrasound		radiotherapy		nursing ward	0.12	...	0.56	false
...												
σ_j	56	order rate		payment		general lab		clinic	4.3	...	1.7	true

(b) *HMM-based* encoding.

Table 2: Encodings for the example in Fig. 2.

3.4 Index-Based Encoding

In the *index-based* encoding, the data associated with events in a sequence is divided into static and dynamic information. Static information is the same for all the events in the sequence (e.g., the information contained in case attributes), while dynamic information changes for different events (e.g., the information contained in event attributes). The resulting feature vector g_i, for a sequence σ_i, is:

$$g_i = (s_i^1, .., s_i^u, event_{i1}, event_{i2}, ..event_{im}, h_{i1}^1, h_{i2}^1 ... h_{im}^1, ..., h_{i1}^r, h_{i2}^r, ...h_{im}^r),$$

where each s_i is a static feature, each $event_{ij}$ is the event class at position j and each h_{ij} is a dynamic feature associated to an event. The example in Fig. 2 is transformed into the encoding shown in Table 2a.

3.5 HMM-Based Encoding

The core idea of HMMs is to provide an abstraction of the information contained in a sequence. However, in general, HMMs are used to *describe* sequential data, not to *classify* it. Moreover, they usually deal only with simple symbolic sequences. The aim of the proposed approach, in contrast, is to be able to *discriminate* between *complex symbolic sequences* with respect to their outcome and make predictions for new, unlabeled sequences.

In order to overcome these limitations of HMMs, we propose some extensions. In order to shift from generative (descriptive) to discriminative models, we take an approach similar to the one presented in [13, 11]. Here, the main idea is to use *discriminative HMMs* to represent a sequence through a measure that captures in some way the relation of the sequence with its outcome. To deal with complex symbolic sequences, the data associated to events is separated into static and dynamic information and the

evolution of each dynamic feature (and the sequence of event classes) is expressed as a simple symbolic sequence. In addition, to encode a case with *HMM-based* encoding, a training set is needed to train the HMMs. In particular, the following steps need to be performed:

- the sequences of event classes and sequences related to each dynamic feature of both the case to be encoded and to the ones in the training set are transformed into simple symbolic sequences;
- the simple symbolic sequences of each dynamic feature (or event class) from the training set are partitioned according to the labels of the cases they belong to. For example, in the binary case one subset corresponds to all sequences that have a positive label and another subset to the sequences with a negative label;
- for each subset of simple symbolic sequences corresponding to a dynamic feature (or event class), a HMM is trained. For example, in the binary case two different HMMs, $\text{HMM}_{positive}$ and $\text{HMM}_{negative}$, are generated;
- for each simple symbolic sequence derived from the case to be encoded, the log-likelihood ratio (LLR) is computed. LLR expresses the likelihood of the sequence to belong to one of the trained models. In the binary case, it shows the likelihood of the sequence to belong to the model describing the positive sequences ($\text{HMM}_{positive}$) over the likelihood to belong to the HMM of the negative ones ($\text{HMM}_{negative}$). Intuitively, the greater the value of LLR is, the greater is the chance that the sequence belongs to a case with a positive outcome. For a case σ_i, and for a given dynamic feature (or event class) h_j, the corresponding log-ratio is defined as:

$$LLR(\sigma_i^{h_j}) = log(\frac{HMM(\sigma_i^{h_j})_{positive}}{HMM(\sigma_i^{h_j})_{negative}}),$$

  where $\sigma_i^{h_j}$ is the simple symbolic sequence extracted from σ_i related to h_j. The information contained in a simple symbolic sequence is, hence, condensed into one number, expressing the relationship of the sequence with a given label value.

The result of applying this procedure to all the information that can be considered as a simple symbolic sequence in a case (sequences of event classes and dynamic data) is a set of LLR values, which are added to the feature vector obtained with the *index-based* encoding. In particular, the input vector for the classifier is, in this case:

$$g_j = (s_j^1, .., s_j^u, event_{j1}, event_{j2}, ..event_{jm}, h_{j1}^1, ...h_{jm}^1, ..., h_{j1}^r, ...h_{jm}^r, LLR_j^1, ..LLR_j^r),$$

where each s_i is a static feature, each $event_{ij}$ is the event class at position j and each h_{ij} is a dynamic feature associated to an event. Each LLR_j^i is the log-likelihood ratio computed based on the simple symbolic sequence corresponding to an event class or a dynamic feature of the original case. Table 2b shows an encoding for the example in Fig. 2 obtained by using log-likelihood ratio values.

Log	# Cases	# Events	# Event Classes
$dataset_1$	1,143	150,291	624
$dataset_2$	1,065	16,869	9

Table 3: Case study datasets.

4 Evaluation

In this section, we provide a description of the carried out experimentation. In particular, our evaluation focuses on the following research questions:

RQ1. Do the proposed encodings provide *reliable* results in terms of predictions?
RQ2. Do the proposed encodings provide reliable predictions at *early* stages of the running case?
RQ3. Are the proposed encodings *stable* with respect to the quality of the results provided at different stages of the running case?

The three questions focus on three intertwined aspects. The first one relates to the quality of the results (in terms of prediction correctness) provided by the proposed encodings. The second one investigates how early the encodings are able to provide reliable results. The third one focuses on the stability of the quality of the results when computed at different stages of an ongoing case. In the following, we describe the experiments carried out to answer these research questions.

4.1 Datasets

We conducted the experiments by using two real-life logs: The BPI challenge 2011 [1] log (herein called $dataset_1$) and an event log (herein called $dataset_2$) of an Australian insurer. The former log pertains to a healthcare process and describes the executions of a process related to the treatment of patients diagnosed with cancer in a large Dutch academic hospital. Each case refers to the treatment of a different patient. The event log contains domain specific attributes that are both case attributes and event attributes in addition to the standard XES attributes.[5] For example, *Age*, *Diagnosis*, and *Treatment code* are case attributes (that we consider as static features) and *Activity code*, *Number of executions*, *Specialism code*, and *Group* are event attributes (that we consider as dynamic features). The second log relates to an insurance claims handling process and covers about one year of completed cases. The insurance claims log includes only event attributes like *Claim type*, *Claim reason*, and *Amount*. Table 3 summarizes the characteristics of the two logs (number of cases, number of events, and number of event classes).

[5] XES (eXtensible Event Stream) is an XML-based standard for event logs proposed by the IEEE Task Force on Process Mining (`www.xes-standard.org`).

4.2 Evaluation Measures

In order to assess the goodness-of-fit for the trained classifiers, we used the Area Under the ROC Curve (AUC) measure [4]. A ROC curve is defined starting from a standard notion of confusion matrix, i.e., the matrix in which each column represents the predicted outcomes of a set of cases, while each row represents the actual outcomes and cells represent:

- true-positive (T_P: cases with positive outcomes predicted correctly);
- false-positive (F_P: cases with negative outcomes predicted as positive);
- true-negative (T_N: cases with negative outcomes predicted correctly);
- false-negative (F_N: cases with positive outcomes predicted as negative).

To draw a ROC curve, two derivatives of the confusion matrix should be defined, i.e., the true positive rate (TPR), represented on the y-axis, and the false positive rate (FPR), represented on the x-axis of the ROC curve. The TPR (or recall), $\frac{TP}{(TP+FN)}$, defines how many positive outcomes are correctly predicted among all positive outcomes available. On the other hand, the FPR, $\frac{FP}{(FP+TN)}$, defines how many negative outcomes are predicted as positive among all negative outcomes available. AUC condenses the information provided by a ROC curve into a single measure of performance. A classifier of the random guess, expressed as a ROC curve, is represented by a diagonal line with AUC of 0.5, while the perfect classifier would score AUC of 1 and is represented by the ROC curve crossing the coordinates $(0, 1)$ - where $\mathit{FPR} = 0$ and $\mathit{TPR} = 1$.

The measure we use to evaluate the earliness of a prediction is based on the number of events that are needed to achieve a minimum value for AUC. Finally, we use standard deviation to evaluate the stability of the results computed at different stages of an ongoing case.

4.3 Evaluation Procedure

In our experimentation, first, we have ordered the cases in the logs based on the time at which the first event of each case has occurred. Then, we have split the logs in two parts. We have used the first part (80% of the cases) as training set, i.e., we have used these cases as historical data. Note that the training set was used differently in the experiments based on the different encodings. For most of them, the entire training set was used to train the random forest classifier. The only exception is the HMM-based encoding that uses 75% of the training set for training the HMMs and 25% for training the random forest. We have used the remaining cases (remaining 20% of the whole log) as a test set (used as ongoing cases).

Next, we have defined 4 temporal constraints corresponding to the following linear temporal logic rules [17] over event classes in dataset$_1$:

- $\varphi_1 = \mathbf{F}(\text{“}tumor\ marker\ CA-19.9\text{”}) \vee \mathbf{F}(\text{“}ca-125\ using\ meia\text{”})$,
- $\varphi_2 = \mathbf{G}(\text{“}CEA-tumor\ marker\ using\ meia\text{”} \rightarrow \mathbf{F}(\text{“}squamous\ cell\ carcinoma\ using\ eia\text{”}))$,
- $\varphi_3 = (\neg\text{“}histological\ examination-biopsies\ nno\text{”})\mathbf{U}(\text{“}squamous\ cell\ carcinoma\ using\ eia\text{”})$,
- $\varphi_4 = \mathbf{F}(\text{“}histological\ examination-big\ resectiep\text{”})$.

LTL	# Positive cases	# Negative cases
φ_1	459	684
φ_2	894	249
φ_3	260	883
φ_4	320	823
γ_1	788	277

Table 4: Distribution of labels in the datasets.

and we have used them to label cases in the training set from dataset$_1$ as compliant or non-compliant (one labeling for each rule). This set of (realistic) rules encompasses all the main linear temporal logic operators. Cases in the training set of dataset$_2$ have been labeled with respect to a constraint corresponding to a rule γ_1 formalizing a regulation internal to the insurance company. This rule requires a claimant to be informed with a certain frequency about the status of his or her claim. The distribution of labels in the datasets is shown in Table 4.

In our experiments, a few input parameters had to be chosen. For random forest classifier, the number of trees was fixed to 500 and the optimal number of features to use for each tree (mtry) was estimated separately using 5-fold cross-validation on the training set. The optimal number of hidden states for HMMs was estimated in a similar way. In particular, the original training set was split, in turn, into training and testing cases and, using these cases, different parameter configurations were tested. The optimal ones – with highest AUC, were chosen for the experiments.

In order to measure the ability of the models to make accurate predictions at an early stage, we computed the AUC values using prefixes ranging from 2 to 20. This choice is justified by the observation that for the defined formulas, encodings based on the sole control flow are able to provide correct predictions after about 20 events.

4.4 Results and Discussion

Figures 3-6 show the trend of the AUC values when predicting the compliance of cases in the test set from dataset$_1$, with respect to φ_1-φ_4. In particular, each plot shows the evolution of the AUC values for the encodings under examination when using the first 20 prefixes of each case in the test set. In Fig. 3, we plot the AUC trend for predictions over the fulfillment of φ_1. For very early predictions the baseline based on the latest data payload gives an AUC that is comparable to the one obtained with complex symbolic sequences. However, for longer prefixes, when more data is available referring to the trend of the attribute values attached to events, this information is exploited by the encodings based on complex symbolic sequences that diverge from the baseline that remains approximately constant. Note that starting from prefixes of length 7 the AUC for both the encodings based on complex symbolic sequences is above 0.9.

Similar trends can be observed in Figures 4-5 referring to the case labeling based on the compliance with respect to φ_2 and φ_3. In the last plot, in Fig. 6, referring to the case labeling based on the compliance with respect to φ_4, the divergence of the

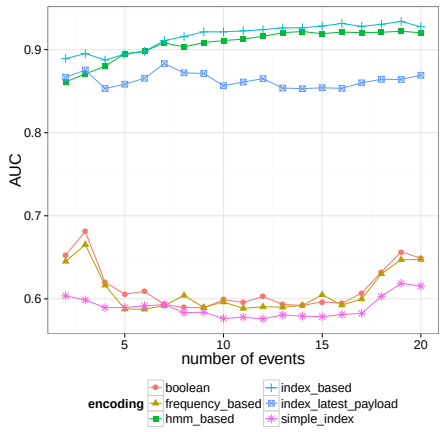

Fig. 3: AUC values using prefixes of different lengths. Labeling based on compliance with respect to φ_1.

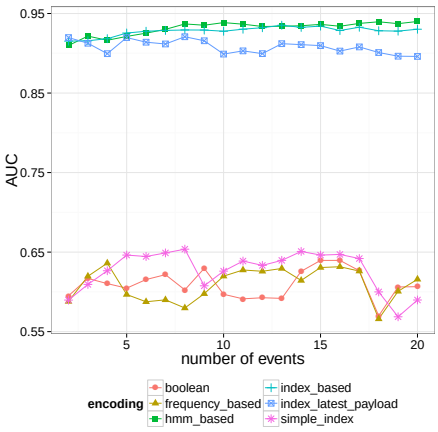

Fig. 4: AUC values using prefixes of different lengths. Labeling based on compliance with respect to φ_2.

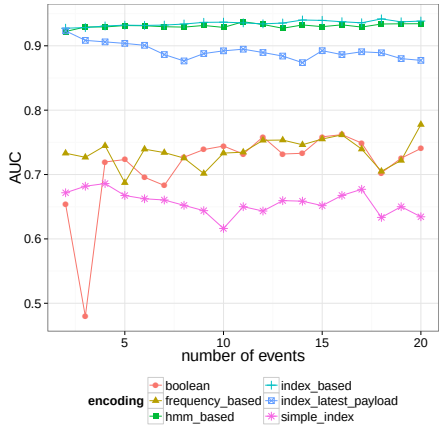

Fig. 5: AUC values using prefixes of different lengths. Labeling based on compliance with respect to φ_3.

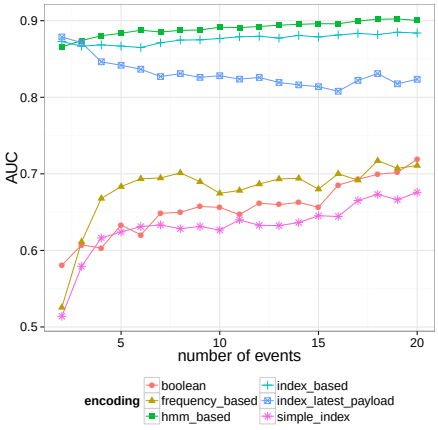

Fig. 6: AUC values using prefixes of different lengths. Labeling based on compliance with respect to φ_4.

encodings based on complex symbolic sequences with respect to the one that considers only the latest data payload is more evident. Here, the HMM-based encoding slightly outperforms the one that considers only indexes.

Fig. 7 shows the AUC trend obtained for the case labeling based on the compliance with respect to γ_1 of cases in dataset$_2$. We can observe that also for this dataset, for early predictions the baseline encoding based on the latest data payload gives a good AUC, while the other baselines have a lower AUC. For slightly longer prefixes (between

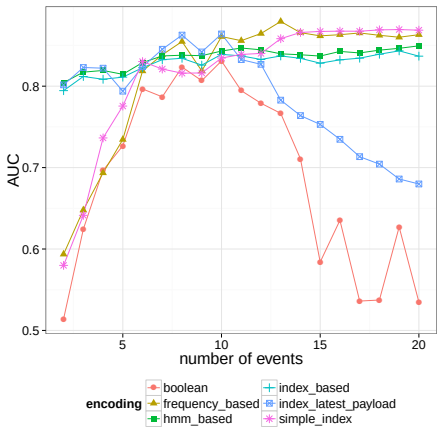

Fig. 7: AUC values using prefixes of different lengths. Labeling based on compliance with respect to γ_1.

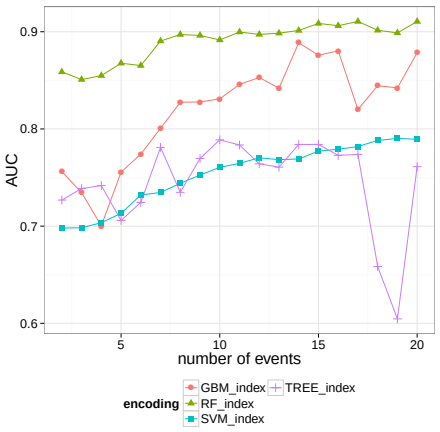

Fig. 8: AUC values using different classification algorithms. Labeling based on compliance with respect to φ_1.

6 and 13), the AUC values of all the baseline encodings is comparable with the one of the encodings based on complex symbolic sequences. From prefixes of length 11 the AUC values for the boolean encoding and for the one based on the latest data payload decrease again. This case study shows that, although baseline encodings can perform very well for certain prefix lengths, their performance is not stable. On the other hand, encodings based on complex symbolic sequences are able to provide a reasonable AUC (around 0.8 in this case) even for short prefixes and to keep it constant or slightly improve it for longer prefixes.

Summing up, the case studies show that the baseline based on the latest data payload and the encodings based on complex symbolic sequences provide, in general, reliable predictions. Table 5, reporting the average AUC values for all the encodings under examination, confirms these results. However, while the baseline encoding is not always able to reach an average AUC value of 0.8, the two encodings based on complex symbolic sequences have an average AUC that is always higher than 0.82. Based on these results, we can, hence, positively answer **RQ1**.

Our experimentation also highlights that some of the presented encodings are able to provide reliable predictions at a very early stage of an ongoing case. As shown in Table 6 (left), the baseline based on the latest data payload and the encodings based on complex symbolic sequences are able to provide an AUC higher than 0.8 in all the cases under examination at a very early stage of an ongoing case (starting from prefixes of length 2 in most of the cases). This is not the case for the other baseline encodings. The encodings based on complex symbolic sequences are also able in most of the cases to reach an AUC higher than 0.9, though not always and at a very early stage of an ongoing case. In fact, both these encodings require 7 events for predicting the fulfillment of φ_1. The HMM-based encoding is the only one able to predict the fulfillment of φ_4 with

	mean across prefixes					st. deviation across prefixes				
encoding	φ_1	φ_2	φ_3	φ_4	γ_1	φ_1	φ_2	φ_3	φ_4	γ_1
boolean	0.614	0.610	0.714	0.655	0.690	0.027	**0.018**	0.063	0.036	0.111
frequency-based	0.609	0.610	0.735	0.679	**0.816**	0.025	0.021	0.022	0.043	0.084
simple index	0.590	0.627	0.656	0.631	**0.814**	**0.013**	0.025	**0.018**	0.036	0.080
index latest payload	**0.863**	**0.908**	**0.892**	**0.831**	0.787	**0.009**	**0.008**	**0.012**	**0.018**	0.060
index-based	**0.917**	**0.928**	**0.935**	**0.876**	**0.828**	**0.016**	**0.006**	**0.004**	**0.006**	**0.013**
HMM-based	**0.907**	**0.932**	**0.931**	**0.890**	**0.835**	**0.018**	**0.009**	**0.003**	**0.010**	**0.013**

Table 5: AUC trends. Bold values show the highest average AUC values (higher than 0.8) and the lowest AUC standard deviation values (lower than 0.02).

	min(prefix) for $AUC = 0.8$					**min(prefix)** for $AUC = 0.9$				
encoding	φ_1	φ_2	φ_3	φ_4	γ_1	φ_1	φ_2	φ_3	φ_4	γ_1
boolean					8					
frequency-based					6					
simple index					6					
index latest payload	2	2	2	2	2		2	2		
index-based	2	2	2	2	3	7	2	2		
HMM-based	2	2	2	2	2	7	2	2	18	

Table 6: Min. number of events needed for an AUC > 0.8 (left) and > 0.9 (right).

an AUC of 0.9 (after 18 events). Starting from these observations, we can positively answer **RQ2**.

Finally, the experiments highlight that some of the encodings have a trend that is more stable than others when making predictions at different stages of the ongoing cases. Table 5 shows that the encodings based on complex symbolic sequences have the most stable AUC trends (the standard deviation for AUC is lower than 0.02 in all the cases). This is not always true for the baseline encodings. We can then provide a positive answer to **RQ3**.

Execution Times All experiments were conducted using R version 3.0.3 on a laptop with processor 2,6 GHz Intel Core i5 and 8 GB of RAM. Table 7 shows the average execution time (in seconds) and the standard deviation (with respect to the time needed to predict the fulfilment for each of the investigated rules) required by the index-based and the HMM-based methods for different prefix lengths. The execution times for constructing the classifiers (off-line) is between 1.08 seconds and 186.41 seconds across all the experiments for the index-based encoding and between 0.99 and 186.41 seconds for the HMM-based encoding. Note that, in addition, the HMM-based encoding also requires time for training the HMMs, ranging from 23.14 to 83.51 seconds. At runtime, the process time for making a prediction on a given prefix of a case is in the order of milliseconds for the runtime prediction on short cases (in the order of seconds for longer cases).

	HMM Training					RF Training					Predictions				
	2	5	10	15	20	2	5	10	15	20	2	5	10	15	20
index-based avg						1.08	5.05	26.29	79.20	176.65	0.23	1.43	6.46	13.37	24.21
index-based s.d.						0.09	0.22	2.46	5.54	12.28	0.05	0.13	0.57	0.78	1.72
HMM-based avg	23.14	34.11	49.03	65.95	83.51	0.99	4.88	26.55	81.74	186.41	0.24	1.45	6.34	13.69	26.40
HMM-based s.d.	1.24	2.53	4.02	4.75	8.23	0.20	0.55	1.18	6.25	11.22	0.05	0.14	0.56	0.92	2.96

Table 7: Execution times per prefix length in seconds.

5 Conclusion

The paper has put forward some potential benefits of approaching the problem of predictive business process monitoring using complex symbolic sequence encodings. The empirical evaluation has shown that an index-based encoding achieves higher reliability when making early predictions, relative to pure control-flow encodings or control-flow encodings with only the last snapshot of attribute values. The evaluation has also shown that encodings based on HMMs may add in some cases an additional margin of accuracy and reliability to the predictions, but not in a significant nor systematic manner.

A threat to validity is that the evaluation is based on two logs only. Although the logs are representative of real-life scenarios, the results may not generalize to other logs. In particular, the accuracy may be affected by the definition of *positive outcome*. For logs different from the ones used here and other notions of outcome, it is conceivable that the predictive power may be lower. A direction for future work is to evaluate the methods on a wider set of logs so as to better understand their limitations.

The methods considered in this paper are focused on the problem of intra-case predictive monitoring, where the aim is to predict the outcome of one individual ongoing case seen in isolation from others. A macro-level version of this problem is the *inter-case predictive monitoring*, where the goal is to make predictions on the entire set of ongoing cases of a process, like for example predicting what percentage of ongoing cases will be delayed or end up in a negative outcome. Initial work on inter-case predictive monitoring [7] has approached the problem using control-flow encodings plus the last snapshot of attribute values. An avenue for future work is to investigate the use of complex symbolic sequence encodings in this context.

Acknowledgments

This research is partly funded by the ARC Discovery Project "Risk-aware Business Process Management" (DP110100091) by the EU FP7 Programme under grant agreement 609190 - "Subject-Orientation for People-Centred Production", and by the Estonian Research Council and by ERDF via the Software Technology and Applications Competence Centre – STACC.

References

1. 3TU Data Center: BPI Challenge 2011 Event Log (2011), doi:10.4121/uuid:d9769f3d-0ab0-4fb8-803b-0d1120ffcf54

2. van der Aalst, W.M.P., Pesic, M., Song, M.: Beyond process mining: From the past to present and future. In: Proc. of CAiSE. pp. 38–52 (2010)
3. van der Aalst, W.M.P., Schonenberg, M.H., Song, M.: Time prediction based on process mining. Inf. Syst. 36(2), 450–475 (2011)
4. Bradley, A.P.: The use of the area under the roc curve in the evaluation of machine learning algorithms. Pattern recognition 30(7), 1145–1159 (1997)
5. Breiman, L.: Random forests. Machine learning 45(1), 5–32 (2001)
6. Castellanos, M., Salazar, N., Casati, F., Dayal, U., Shan, M.C.: Predictive business operations management. In: Proc. of DNIS. pp. 1–14. Springer (2005)
7. Conforti, R., de Leoni, M., Rosa, M.L., van der Aalst, W.M.P., ter Hofstede, A.H.M.: A recommendation system for predicting risks across multiple business process instances. Decision Support Systems 69, 1–19 (2015)
8. Feldman, Z., Fournier, F., Franklin, R., Metzger, A.: Proactive event processing in action: a case study on the proactive management of transport processes. In: Proc. of DEBS. pp. 97–106. ACM (2013)
9. Fernández-Delgado, M., Cernadas, E., Barro, S., Amorim, D.: Do we need hundreds of classifiers to solve real world classification problems? The Journal of Machine Learning Research 15(1), 3133–3181 (2014)
10. Folino, F., Guarascio, M., Pontieri, L.: Discovering context-aware models for predicting business process performances. In: Proc. of OTM Confederated Conferences, pp. 287–304. Springer (2012)
11. Goldszmidt, M.: Finding soon-to-fail disks in a haystack. In: Proc. of HotStorage. USENIX (2012)
12. Kang, B., Kim, D., Kang, S.H.: Real-time business process monitoring method for prediction of abnormal termination using knni-based lof prediction. Expert Syst. Appl. (2012)
13. Leontjeva, A., Goldszmidt, M., Xie, Y., Yu, F., Abadi, M.: Early security classification of skype users via machine learning. In: Proc. of AISec. pp. 35–44. ACM (2013)
14. Maggi, F.M., Di Francescomarino, C., Dumas, M., Ghidini, C.: Predictive monitoring of business processes. In: Proc. of CAiSE. pp. 457–472. Springer (2014)
15. Metzger, A., Franklin, R., Engel, Y.: Predictive monitoring of heterogeneous service-oriented business networks: The transport and logistics case. In: Proc. of SRII Global Conference. IEEE (2012)
16. Pika, A., Aalst, W., Fidge, C., Hofstede, A., Wynn, M.: Predicting deadline transgressions using event logs. In: Proc. of BPM Workshops, vol. 132, pp. 211–216. Springer (2013)
17. Pnueli, A.: The temporal logic of programs. In: Proc. of FOCS. pp. 46–57. IEEE (1977)
18. Rabiner, L.: A tutorial on hidden markov models and selected applications in speech recognition. Proceedings of the IEEE 77(2), 257–286 (1989)
19. Ridgeway, G.: Generalized boosted models: A guide to the gbm package. Update 1(1) (2007)
20. Rogge-Solti, A., Weske, M.: Prediction of remaining service execution time using stochastic petri nets with arbitrary firing delays. In: Proc. of ICSOC. pp. 389–403. Springer (2013)
21. Suriadi, S., Ouyang, C., Aalst, W., Hofstede, A.: Root cause analysis with enriched process logs. In: Proc. of BPM Workshops, pp. 174–186. Springer (2013)
22. Suriadi, S., Wynn, M.T., Ouyang, C., ter Hofstede, A.H.M., van Dijk, N.J.: Understanding process behaviours in a large insurance company in Australia: A case study. In: Proc. of CAiSE. pp. 449–464. Springer (2013)
23. Suykens, J.A., Vandewalle, J.: Least squares support vector machine classifiers. Neural processing letters 9(3), 293–300 (1999)
24. Xing, Z., Pei, J., Keogh, E.J.: A brief survey on sequence classification. SIGKDD Explorations 12(1), 40–48 (2010)