Comprehensible Predictive Models for Business Processes

Dominic Breuker
Hitfox Group GmbH,
Rosa-Luxemburg-Str 2, 10178 Berlin, GERMANY
{dominic.breuker@hitfoxgroup.com}

Martin Matzner
European Research Center for Information Systems (ERCIS), University of Muenster,
Leonardo-Campus 3, 48149 Münster, GERMANY
{martin.matzner@ercis.uni-muenster.de}

Patrick Delfmann
Institute for IS Research, University of Koblenz-Landau,
Universitätsstraße 1, 56070 Koblenz, GERMANY
{delfmann@uni-koblenz.de}

Jörg Becker
European Research Center for Information Systems (ERCIS), University of Muenster,
Leonardo-Campus 3, 48149 Münster, GERMANY
{joerg.becker@ercis.uni-muenster.de}

This is an author-created version of a manuscript that is accepted for publication and forthcoming at Management Information Systems Quarterly, Special Issue on Transformational Issues of Big Data and Analytics in Networked Business.

Breuker, D., Matzner, M., Delfmann, P., and Becker, J., "Comprehensible Predictive Models for Business Processes", *MIS Quarterly*, forthcoming.

Comprehensible Predictive Models for Business Processes

Abstract: Predictive modeling approaches in business process management provide a way to streamline operational business processes. For instance, they can warn decision-makers about undesirable events that are likely to happen in the future, giving the decision-makers an opportunity to intervene. The topic is gaining momentum in process mining, a field of research that has traditionally developed tools to discover business process models from datasets of past process behavior. Predictive modeling techniques are built on top of process-discovery algorithms. As these algorithms describe business process behavior using models of formal languages (e.g., Petri nets), strong language biases are necessary in order to generate models with the limited amounts of data included in the dataset. Naturally, corresponding predictive modeling techniques reflect these biases. Based on theory from grammatical inference, a field of research that is concerned with inducing language models, we design a new predictive modeling technique based on weaker biases. Fitting a probabilistic model to a dataset of past behavior makes it possible to predict how currently running process instances will behave in the future. To clarify how this technique works and to facilitate its adoption, we also design a way to visualize the probabilistic models. We assess the technique's effectiveness in an experimental evaluation with synthetic and real-world data.

Keywords: Process Mining, Process Discovery, Business Process Intelligence, Grammatical Inference, Predictive Modeling

INTRODUCTION

Predictive analytics has evolved into one of the major topics in research, high on many organizations' agendas and considered a game-changer for the economy (Lund et al. 2013). Researchers in various disciplines, including information systems (IS), explore the opportunity of exploiting data in new, innovative ways (Shmueli and Koppius 2011), so predictive analytics is also gaining momentum in business process management (BPM), an established field in IS research. Process mining, which has emerged as an application area of data analysis in the BPM context (Chen and Storey 2012), refers to the development of tools and methods to generate insights based on event data collected during the execution of a business process (van der Aalst 2011).

Since work on process mining began in the mid-1990s, the goal has been primarily to support retrospective analysis by, for instance, constructing visual representations of business processes from data or verifying that processes are executed as intended (van der Aalst 2011). With the increasing popularity of predictive analytics, interest has arisen in using process mining to analyze not just the past but also the present and the future to gain comprehensive insight into a process (Grigori et al. 2004). Monitoring instances of business processes (Janiesch et al. 2012) and predicting their future behavior (Maggi et al. 2014) can enable managers to act proactively in anticipation of events.

The main contribution of this paper is a predictive modeling technique for business processes. Trained with event data collected during the execution of business processes, the technique can be applied to real-time event data to reason probabilistically how currently running process instances will behave in the future. In particular, it allows likely future behavior to be predicted. The technique facilitates a large number of analytical uses of "Big Data"—that is, datasets that traditional information technologies and computing approaches cannot perceive, acquire, manage, and process "in a tolerable time" (Chen et al. 2014, p. 173) —including early warning systems and anomaly detection (Chen and Storey 2012).

- *Early warning systems*: Decision-makers could monitor the likelihood of future negative events such that an alert is triggered when the likelihood of the event's occurring reaches a particular threshold. For instance, early warning systems could point managers to process instances that are in danger of violating service-level agreements, giving managers the opportunity to intervene before the organization suffers negative consequences.

- *Anomaly detection*: Decision makers could monitor the likelihood of currently running process instances to identify such instances as fraudulent behavior or those that require expert attention.

While we provide only a prototypical open source implementation, our technique is easily parallelizable. Thus, the elevated capabilities offered by Big Data computing platforms and technologies like Apache Hadoop are applicable to calculate the needed probabilistic models in a timely manner, which promotes the technique's practical applicability.

RESEARCH GOAL AND METHODOLOGY

The main goal of our research is to develop a *predictive process modeling technique* based on *process mining* and *grammatical inference* that *accurately* predicts future behavior of business processes and provides *comprehensible* results.

Predictive Modeling Based on Process Mining and Grammatical Inference

Predictive modeling is a novel topic of research in the area of process mining, and only few approaches have yet been designed. The mainstream design principle for such systems is to enhance a business process model (constructed with a process-mining technique) with additional information (van der Aalst 2013). However, we depart from established theory and pursue a different approach that is based on the relationship between process-mining techniques and grammatical-inference techniques. Process-mining techniques build formal models that describe a process as a set of all valid process instances (van der Aalst 2011), so these techniques are similar to traditional approaches in the field of grammatical inference. These grammatical-inference techniques build formal models that describe languages as sets of valid sentences (de la Higuera 2005). Language models are foundational to many modern technologies that deal with language, such as search engines and machine translation systems, but to get where it is now, grammatical inference had to depart from using formal, logical models and base its contemporary applications on probabilistic approaches that model a language as probability distributions over sentences instead of as a set of valid sentences (Chater and Manning 2006). Such models do a better job of accounting for the noisy data and uncertainty that arise from limited amounts of data and they facilitate the use of weak language biases, which are restrictions on the expressiveness of the language representable by the model, in practical applications (Norvig 2011, p. 5). Therefore, we explore the opportunities that arise from applying probabilistic grammatical-inference techniques to BPM.

Comprehensibility

Another issue related to predictive modeling is comprehensibility. In practical applications, predictive-modeling techniques often suffer from incomprehensibility (Provost and Fawcett 2013), as modern predictive models often rely on estimates of millions of parameters. As a result, explanations for the results are elusive, and managers can be reluctant to use systems based on these techniques for decision support (Provost and Fawcett 2013). The decision support systems literature long ago identified the need for comprehensible explanations (Lilien et al. 2004), as such explanations facilitate acceptance of decision-support software (Arnold et al. 2006; Kayande et al. 2009). Martens and Provost (2014) develop a theoretical framework that emphasizes the importance of comprehensibility for three types of user roles, stressing that managers must understand a predictive model to have the confidence in the system necessary to using it. Clients may also want explanations for decisions based on predictive models that affect them. Developers of a decision support system also benefit from understanding the inner workings of a predictive model, as such understanding helps them improve the system’s performance.

With a strong focus on improving how well business processes are understood by constructing business process models from data, process-mining techniques often provide results that are naturally comprehensible, so their level of adoption in practice is high (Turner et al. 2012). However, because moving to a grammatical-inference technique based on complex probabilistic models may hamper acceptance, our design goal is to choose the probabilistic model for our technique in a way that maintains comprehensibility. In particular, we adapt a technique from process mining that allows us to construct business process models as abstractions of fitted probabilistic models. Therefore, users who have little or no technical expertise can inspect abstractions of the probabilistic models using a visual notation that is native to the domain of BPM.

Research Goals

In summary, the two goals of our design are accurate prediction of business processes and comprehensibility.

- *Accurate prediction of business processes*: Design a probabilistic technique that, after being trained on historic data, can predict the future behavior of currently running process instances.
- *Comprehensibility*: Design a visualization technique for probabilistic models such that users who have little or no deep technical knowledge can interpret and understand the models.

Research Methodology

This research follows the design science paradigm, and the structure of this paper is based on the design science publication schema that Gregor and Hevner (2013) propose. Our primary purpose is to design a method for modeling business process event data probabilistically, a method to which we refer as the RegPFA artifact. The artifact consists of two components: the RegPFA Predictor and the RegPFA Analyzer. We developed an instantiation of the artifact, which is publicly available under a BSD license.[1]

In line with Gregor and Hevner (2013), this section defines the purpose and scope of the RegPFA artifact and emphasizes its practical relevance. We first provide justification for the effort by discussing process mining (van der Aalst 2011), which introduces the learning problem and the general setting, and grammatical inference (de la Higuera 2010), which introduces the methodological apparatus we use and lays the foundation for defining a suitable probabilistic model. Then we describe the artifact's design, presenting the RegPFA Predictor first by describing and justifying modifications to the probabilistic model identified from the literature and describing the technique we designed to fit it to event data, followed by the RegPFA Analyzer, which transforms fitted probabilistic models into business process models. Next, we assess the effectiveness of both components experimentally. Finally, we discuss our results and conclude.

[1] Available at https://github.com/DominicBreuker/RegPFA.

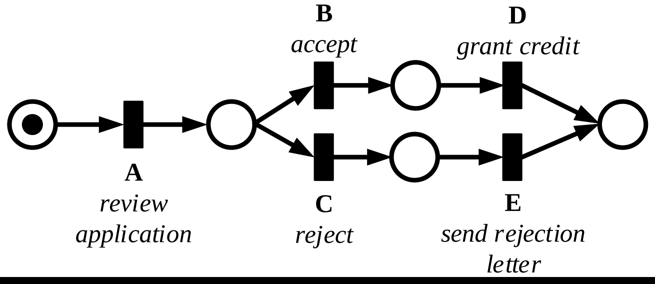
RESEARCH BACKGROUND

Business Process Mining

According to van der Aalst (2013, p. 1), BPM “is the discipline that combines knowledge from information technology and knowledge from management sciences and applies this to operational business processes.” BPM has traditionally revolved around the notion of business process models, which are abstract, visual representations of business processes that are usually designed manually in interviews with process experts. The abundance of data available today, which is the primary motivation for the research discipline of process mining, has driven calls for data-driven approaches to BPM (van der Aalst 2011).

Van der Aalst (2013, p. 22) defines process mining as a “discipline providing comprehensive sets of tools to provide fact-based insights and to support process improvements.” Using the results of a survey, van der Aalst (2013) observes that process mining’s share of the BPM literature increased from zero to 31.4 percent from 2000 to 2011, which makes it the most important key concern in BPM today.

Process mining starts with collecting sequential records of events. Events are of exactly one type, which refers to activities that have been performed, decisions that have been made, or another occurrence of interest. Each event belongs to a process instance and a point in time, so event sequences can be formed for each instance (van der Aalst 2011). The example in Fig. 1 illustrates a simple loan application process. The example, modeled in Petri net notation (Murata 1989), starts with reviewing the application, after which it is either accepted or rejected. If it is accepted, credit is granted to the applicant; if not, the applicant is notified accordingly. The event log shown in Fig. 1 could be created based on data from the organization’s IT systems to document the sequence of events for each case.



Event log

Process instance	Event sequence
C1	ABD
C2	ACE
C3	ABD
C4	ACE
C5	ACE

Fig. 1. A business process in Petri net notation, along with an exemplary event log

There are three classical types of process mining (van der Aalst 2011): process discovery, conformance checking, and enhancement. *Process discovery* refers to learning a process model inductively from data. In the example in Fig. 1, process-discovery techniques like the α-algorithm (van der Aalst et al. 2004) are applied to construct a Petri net using only the information contained in an event log. *Conformance checking* refers to comparing process models with data. If the Petri net in Fig. 1 is interpreted as a description of what the process should be, comparing the event log to the process model could indicate whether the process has been executed correctly. Finally, *enhancement* refers to extending or improving process models based on data. For instance, the Petri net in Fig. 1 could be annotated with frequencies that indicate how often applications have been accepted or rejected (e.g., 2/5 and 3/5).

Van der Aalst et al. (2010a, p. 38) observe that "process mining [analyzing historic data] is mainly used in an offline fashion and not for operational decision support," but interest in analyzing real-time data from currently running process instances has increased (Janiesch et al. 2012; van der Aalst et al. 2011a). Techniques that predict how long before running instances complete (van der Aalst et al. 2011b) or that recommend actions to improve performance (Kim et al. 2014) have been developed. These approaches use existing process models, possibly annotated with further information, in combination with real-time event data (van der Aalst 2013), so process-discovery techniques are important components since they construct these process models from historic event data. A discovered process model that is combined with sequences of events observed during the execution of process instances allows the process's current state to be characterized. Predictions of likely outcomes of decisions made about a process instance can be conditioned on the current state (van der Aalst 2013). Thus, the extent to which the future course of a process instance is predictable based on the events observed so far depends largely on how the current state of a process instance is characterized.

Many approaches to process mining that are designed to predict behavior or recommend actions characterize the state of the process according to the definitions used in the two-step process-discovery algorithm van der Aalst et al. (2010) present. At each point in time, this algorithm considers the last n events and interprets them as either a sequence, a multi-set, or a set. Thus, a unique state of the process is known at any time, and a corresponding automaton can be built. Based on this (or similar) characterization of states, it is possible to predict, for instance, a process instance's time of completion as the mean amount of time remaining for the process when it is in a particular state (van der Aalst et al. 2011b) and to apply more complex models, such as decision trees (Folino et al. 2012). Other process-discovery algorithms can also be the foundation for real-time analysis, such as Lakshmanan et al.'s (2015) approach to predicting the type of the next event in a running process instance, the first step of which is to apply a process-discovery algorithm. Therefore, process-discovery algorithms have an important role in predictive, real-time approaches, as they characterize the state of the process.

The literature contains a significant number of process-discovery techniques. (See Tiwari et al. (2008) or de Weerdt et al. (2012) for surveys.) First approaches, developed in the late 1990s (van der Aalst et al. 2003b), were not designed for processes with concurrent behavior, so the α-algorithm, which constructs Petri nets that can express concurrent behavior in a compact way, was developed (van der Aalst et al. 2004). Various extensions have been developed to mitigate the problem of the α-algorithm's ability to construct only models with simple routing constructs. For instance, the α+-algorithm (Li et al. 2007) supports various kinds of loops, and the α++-algorithm (Wen et al. 2007) learns dependencies between events that are not directly related in the event log.

Apart from supporting these complicated routing constructs, process-discovery techniques face challenges related to the quality of event logs, which challenges can be categorized broadly into noise and incompleteness (van der Aalst 2011). *Noise* refers to behavior that is in the data but that should not be represented in a discovered process, while *incompleteness* refers to event logs' being only a finite sample of possible process behaviors—a sample that is not necessarily representative and that may be missing important behavior.

Many process-discovery techniques, for instance the Heuristics Miner (Weijters et al. 2006) and the AGNEs miner (Goedertier et al. 2009), address noise by analyzing the frequency of patterns of behavior. A pattern is used to deduce the process model only if the pattern is found sufficiently often. Incompleteness is more difficult to address. One possibility is to design discovery techniques that deduce as much missing behavior as possible from the available data (van der Aalst 2011). However, in a practical application this solution still requires faith in the event log's completeness. Moreover, too much deduction can easily produce over-generalized models, a problem as severe as under-generalization. One solution is to enable the user of a process-discovery technique to manipulate the degree of generalization until a result is produced with which the user is comfortable (van der Aalst et al. 2010). Another solution involves switching to a different process representation, as most process-discovery algorithms use Petri nets (or similar notations), which formally specify the set of all acceptable event sequences for a process. However, other discovery techniques include the Fuzzy miner (Günther and van der Aalst 2007), which does not produce Petri nets but a visual representation of the importance of individual types of events and the correlations between them, and trace alignment (Bose and van der Aalst 2012), which clusters event sequences and visualizes representatives for each cluster.

Only a few techniques consider probabilistic models of business processes. Of the twenty-six discovery algorithms de Weerdt et al. (2012) review, only four are tagged as probabilistic; three of the four are more than a decade old (Cook and Wolf 1998; Herbst and Karagiannis 2004), and the most recent technique (Ferreira and Gillblad 2009) is designed to discover Markov chain models from event logs for which the process instance to which an event belongs is unknown.

Few probabilistic models of business processes build on concepts that are similar to those we use. For instance, Blum et al. (2008) use Hidden Markov Models (HMM) to mine a surgery workflow from a log containing ten process instances. Jeong et al. (2010) also use HMMs to analyze students' learning behavior. Weber et al. (2013) introduce a method with which to mine process models with Probabilistic Finite Automata (PFA) applied to the α-algorithm. Our approach is also based on PFA, but to avoid overfitting we introduce a modification to the PFA that is based on Bayesian regularization.

Probabilistic techniques are popular in many fields because of their ability to handle noisy, possibly incomplete data. In particular, probabilities play an important role in machine learning and data-mining techniques. (E.g., see Murphy (2012) or Aggarwal and Yu (2009).) A possible disadvantage of probabilistic techniques for process discovery is that they do not directly deliver a formal, executable model such as a Petri net, which may explain the probabilistic techniques' low popularity. However, since our primary interest lies in designing a predictive modeling technique for business processes based on event data, probabilistic techniques could be suitable. (We review them in the next section.) Our approach differs from the practice of first discovering a model structure and subsequently annotating probabilistic information, as is prevalent in the process-mining literature (van der Aalst 2013), as we design a probabilistic technique to model a probability distribution directly over the set of all conceivable event sequences.

Grammatical Inference

Grammatical inference is a field of research that is concerned with learning grammars from data. As de la Higuera (2005, p. 1332) explains, "In a broad sense a learner has access to some data which is sequential or structured (strings, words, trees, terms or limited forms of graphs) and is asked to return a grammar that should in some way explain these data." Grammatical inference research provides methodological support for this abstract learning problem.

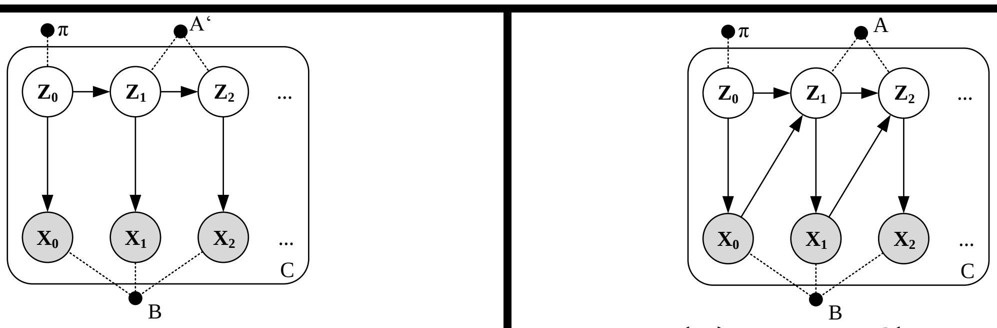
Models of grammars have traditionally been formal models of languages as they are used in computer science—that is, models that describe a set of strings that are part of a language (de la Higuera 2010). These models can be either grammars, which describe languages by means of recursive application of production rules, or automata, which describe languages by means of abstract machines that generate words (Hopcroft et al. 2009). Using automata to describe sets of strings that define languages is similar to process-mining approaches, which learn models that describe sets of event sequences that define processes.

When applied to real-world problems, these formal grammar models are often unsatisfying (de la Higuera 2005), as noise can render a learned model all but useless. For instance, as soon as an unlikely exception is part of the log, the exception will also be part of the model, while a usually likely event that might not be part of the log will not. As the data typically consists only of positive examples, deciding whether enough data has been used to justify trusting the learned models is a major challenge. Therefore, contemporary applications often apply probabilistic versions of the formal models discussed above (de la Higuera 2005), which had "a revolutionary impact", for instance, in computational linguistics (Chater and Manning 2006, p. 337).

Probabilistic models do not define languages as sets of strings but as distributions over strings (de la Higuera 2010). Accordingly, learning probabilistic languages requires approximating distributions rather than identifying sets of strings. Learning a language represented as a set of strings requires a strong language bias, as using a finite amount of data to select one of the infinite sets of strings that are consistent with the data is possible only if strong assumptions rule out most of the sets. However, considering an infinite set of models is not problematic in learning a probabilistic language, as the most likely explanation for the data can be chosen regardless of the number of explanatory candidates. Thus, a probabilistic setting also overcomes the problems associated with classical techniques' assumptions (Nowak et al. 2002).

Grammatical inference and process mining have substantial similarities. In particular, an event log consists of several sequences of process events, and each sequence can be interpreted as word of a grammar that prescribes the kinds of sequences we can build in principle. In the case of business processes, that grammar is the process model: it prescribes which sequences of events are possible. Hence, applying grammatical inference to event logs requires learning the language of the event logs, which means learning the process model. Hence, grammatical inference techniques can be applied to event logs to create a model that describes the corresponding business process, as the central learning problems of both fields are similar. BPM scholars recognize the challenges that motivate the use of probabilistic models in grammatical inference, yet probabilistic models are rarely used in BPM. This observation motivates an exploration of grammatical inference as a theoretical basis for the analysis of business process event data.

The most important probabilistic models in grammatical inference are the Hidden Markov Model (HMM) and the Probabilistic Finite Automaton (PFA) (Verwer et al. 2014), both of which are particular kinds of Bayesian network. These models are defined formally in Fig. 2 and are illustrated in directed graphical models notation, which are visual representations of probabilistic models (Koller and Friedman 2009). In graphical models, circles correspond to random variables and are shaded gray if observational data is available. Directed arcs describe the dependencies among variables. Model parameters take the form of black dots, which dashed lines connect to the variables whose distributions they define. Boxes (so-called *plates*) surround the parts of a probabilistic model that are repeated.



$$P(Z_0) \sim Categorical(\pi_0, \dots, \pi_K)$$
$$P(X_t | Z_t = k) \sim Categorical(b_{k0}, \dots, b_{kE})$$
$$P(Z_t | Z_{t-1} = k) \sim Categorical(a'_{k0}, \dots, a'_{kK})$$

$$P(Z_0) \sim Categorical(\pi_0, \dots, \pi_K)$$
$$P(X_t | Z_t = k) \sim Categorical(b_{k0}, \dots, b_{kE})$$
$$P(Z_t | Z_{t-1} = k, X_{t-1} = e) \sim Categorical(a_{ke0}, \dots, a_{keK})$$

Fig. 2. Comparison of probabilistic models – HMM (left) and PFA (right)

The left side of Fig. 2 illustrates the HMM (Rabiner 1989), whose initial state, Z_0, depends on no other variables and whose value is chosen randomly from a categorical distribution parameterized by the probability vector π. All other state variables $Z_1, Z_2, \dots$ depend on the preceding state variables, and their values are chosen randomly from a categorical distribution parameterized by probability vectors a'_k. The events observed during process execution depend only on the HMM's current state and are chosen randomly from categorical distributions parameterized by probability vectors b_k.

Similarities between business processes and the HMM's structure against the backdrop of the BPM domain are evident. In business process modeling notations like Petri nets, processes are conceptualized as systems with an underlying state that depends on a process instance's past behavior (van der Aalst 1998) and that is sufficient to describe which events could be observed next. For instance, knowing which of a Petri net's places currently have tokens allows one to determine which transitions are enabled. Analogously, the HMM's probabilities can describe the likelihood that a given event will be observed, depending on the state.

However, contrasting the HMM and BPM also reveals differences. In the HMM, the state transition probabilities depend only on the previous state; the event observed at time $t-1$ has no impact on the distribution of Z_t. In BPM, this model structure is counterintuitive, as we expect that the next state depends on the observed event. Consider again the example of Petri nets: given a current state, the state after firing one of the enabled transitions usually depends on the transition that is fired. These considerations suggest using a PFA (Vidal et al. 2005) instead of an HMM. The right side of Fig. 2 illustrates the PFA's model structure, which is similar to HMM but also considers events as influence factors.

The PFA's initial state, Z_0, depends on no other variables as its value is chosen randomly from a categorical distribution parameterized by the probability vector π. All other state variables $Z_1, Z_2, \dots$ depend on the preceding state variables *and* on the events observed during process execution $X_0, X_1, \dots$. The values of the state variables are chosen randomly from a categorical distribution parameterized by probability vectors a'_{ke}, considering both the preceding state and the preceding observed event. The events observed during process execution again depend only on the PFA's current state and are chosen randomly from categorical distributions parameterized by probability vectors b_k. As this model structure resonates well with the conceptualizations of business processes, we use it as a starting point in defining our probabilistic model.

In order to apply grammatical inference, we need a suitable probabilistic model but also a technique with which to fit it to data. The available techniques can be categorized broadly into three classes (Verwer et al. 2014): state merging, parameter estimation, and Bayesian inference.

- *State merging*: Algorithms in this class start with an automaton that initially consists of a large number of states. Then the algorithms iteratively merge states, reducing the complexity of the automaton. The procedure stops when no pair of states is worth merging anymore. Various algorithms apply various merging criteria, such as those that are based on Hoeffding bounds (Carrasco and Oncina 1994), posterior probability (Stolcke 1994), or distribution similarity (Ron et al. 1998).
- *Parameter estimation*: Unlike state-merging algorithms, parameter estimation techniques do not learn a model's states and how they are connected but start with a standard structure and estimate parameters by applying the maximum likelihood (ML) principle. The most popular example of the standard structure, usually an automaton with a given number of states and all possible transitions, is the Baum-Welch algorithm, which is a special case of the more general Expectation Maximization (EM) algorithm (Dempster et al. 1977) used for HMM learning (Baum et al. 1970).

- *Bayesian inference*: Unlike state merging and parameter estimation, Bayesian inference techniques do not learn a single model but average all possible models. A popular technique of this kind is Gibbs sampling, which generates a large number of samples from the posterior distribution, each of which can be used for prediction. The final prediction is computed as an average (Gao and Johnson 2008).

Since a design goal of our approach is that all probabilistic models be visually interpretable and understandable without the need for technical knowledge, Bayesian inference techniques, which do not generate a single PFA estimate so their results are not visualizable, are not suitable for our artifact. Therefore, state merging and parameter estimation techniques are left as possible candidates. Recently a grammatical inference challenge was offered in which state-of-the-art PFA and HMM learning techniques were evaluated in a grammatical-inference context based on a large number of datasets (Verwer et al. 2014). We acknowledge that process-mining specifics (especially concurrency) may limit Verwer et al.'s study's expressiveness for our problem. However, the authors indicate opportunities to identify promising techniques and, leaving out Bayesian inference techniques, Hulden's (2012) solution, which implements an EM algorithm for PFAs, performed best. That the state-merging techniques performed worse than the EM approach suggests that they should be discarded, so the challenge's results suggest using an EM parameter estimation algorithm for our purposes.

ARTIFACT DESIGN

Before introducing the design of the RegPFA artifact in detail, we present an overview of the components and their interplay. The RegPFA Predictor provides the predictive modeling functionality, while the RegPFA Analyzer provides visualization and analysis. The artifact is illustrated in Fig. 3 using a simplified Standardized Technical Architecture Modeling Component Diagram (SAP 2007).

In the RegPFA Predictor the workflow starts with application of a learning algorithm to an event log. The learning algorithm delivers a probabilistic model fitted to the event data that (one hopes) reflects the dynamics of the underlying business process. When this probabilistic model is applied to the real-time event data of currently running process instances, predictions about how these instances will behave in the future can be evaluated probabilistically.

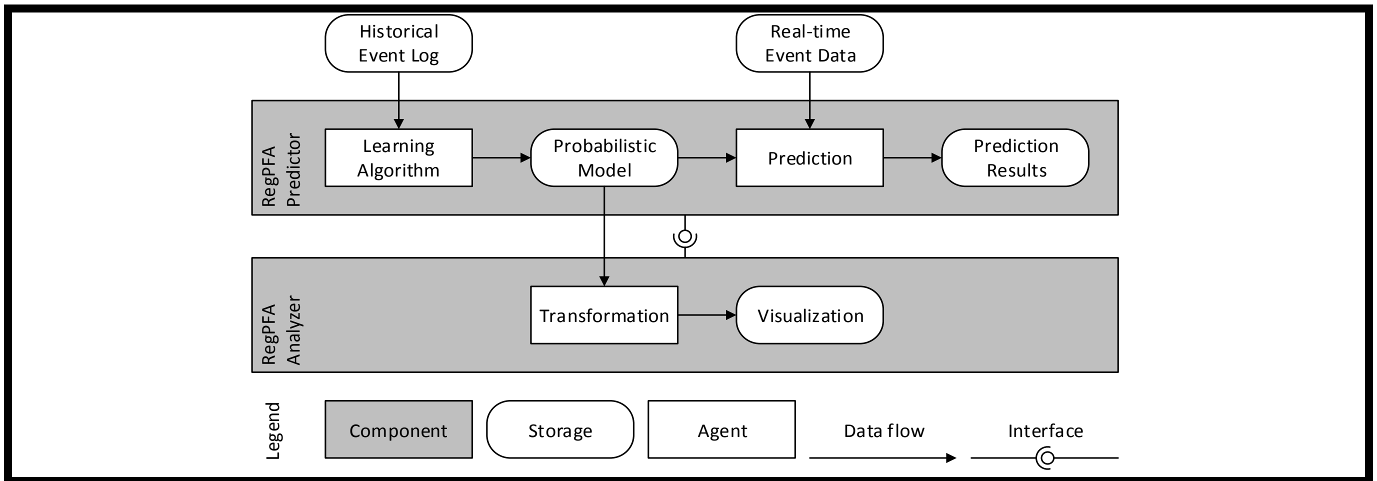


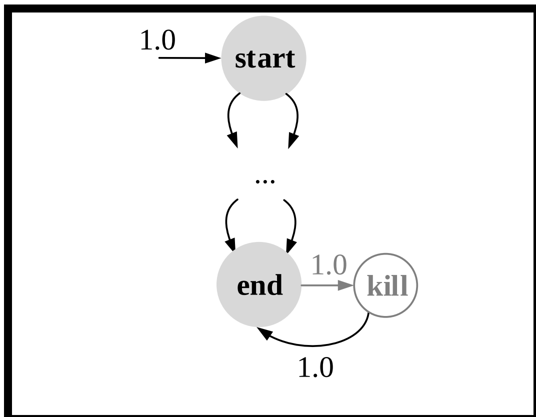
Fig. 3. Overview of the RegPFA Artifact

The other component, the RegPFA Analyzer, can be used only after the learning algorithm has produced a probabilistic model fitted to an event log. Using the probabilistic model as input, a transformation component generates a visualization that humans can understand. Using process modeling notations (Petri nets and automata) for the visualization ensures that domain experts who do not have technical knowledge about predictive modeling can understand the visualization, inspect predictive models, and decide whether the model's structure reflects or contradicts the experts' domain knowledge.

Design of the RegPFA Predictor

Probabilistic Model

We identified the PFA as the starting point for the RegPFA Predictor. This section introduces modifications with which to tailor the RegPFA Predictor to BPM—that is, structural constraints we impose in order to produce well-structured process models when fitting PFAs to event logs. These constraints are inspired by the definition of workflow nets, which have designated start and end points (the *source* and the *sink*) (van der Aalst 1998). Analogously, we define a special starting state by keeping π fixed at a value that allows the process to start only in that state. In the same way, we also define a special ending state at which only a designated termination event ("*kill*") can be emitted and where the process is forced to stay in that state. *Kill* is an event that occurs only in the ending state and that must be appended to each process instance in an event log to indicate termination. All states other than the ending state have zero probability of generating *kill*. The left part of Fig. 4 illustrates these structural assumptions, which can be implemented by keeping the corresponding parameters fixed at suitable values during parameter optimization.



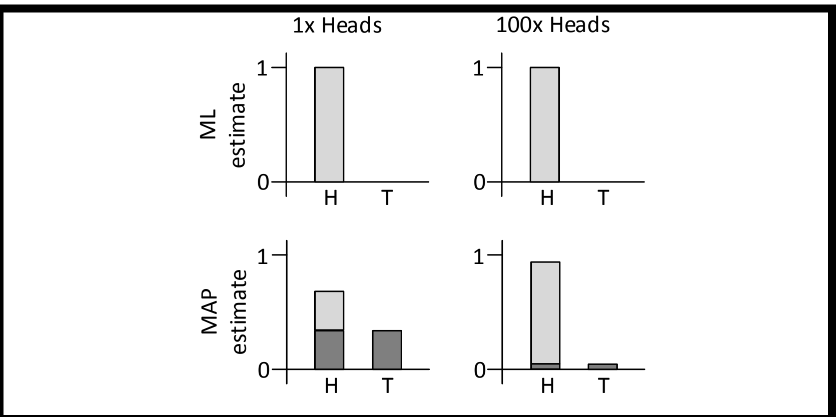


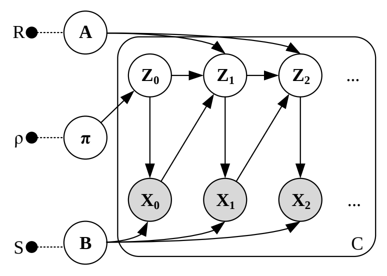
Fig. 4. Illustrations of the modified model structure (left) and parameter estimation (right)

Our discussion of grammatical inference also determined that parameter estimation is the family of techniques that is most suitable for our design endeavor. In particular, we identified an EM-based technique, an ML estimator, but a problem with ML is that it can easily produce overfitted estimations if datasets are too small (Hastie et al. 2009). Overfitting can be a severe problem for process data, so process miners consider incomplete data one of the major challenges in their discipline (van der Aalst et al. 2011a). We address this problem by introducing a modification to the PFA that is based on Bayesian regularization (Steck and Jaakkola 2002).

This technique is best illustrated with the example of flipping a coin (cf. Fig. 4, right). The goal is to estimate p, the probability of the coin's coming up heads. In the first scenario illustrated in Fig. 4, heads is observed only once, and in the second scenario heads is observed one hundred times with no tails observations. Maximum likelihood estimation delivers $p = 1$ in both scenarios, which appears to be good for the second scenario, while the first estimate appears to overfit a single observation. The first estimate contradicts the subjective belief that extreme probabilities close to 1 are unlikely.

One remedy to this problem is to move to a Bayesian framework, which can account for the belief that extreme probabilities are unlikely. In a Bayesian framework, p is not treated as a parameter but as an uncertain quantity in the same way that the numbers of heads and tails are uncertain; that is, p is treated as a random variable with a distribution called a *prior*, and its parameters are the *hyperparameters* of a probabilistic model. If a beta distribution (the *conjugate prior* to the binomial distribution (Barber 2011)) were used in the coin-flipping example, its two parameters would express the belief that p will have a particular value (e.g., 0.5) and they would express the strength of this belief in terms of *pseudo-observations*. For instance, if p is expected to be 0.5 and the strength of this expectation is set to two observations, one empirical observation of heads is stacked upon this prior belief in scenario 1, which delivers $p = \frac{1+1}{2+1} = \frac{2}{3}$. A hundred empirical observations deliver $p = \frac{1+100}{2+100} = \frac{101}{102}$, which is close to the ML estimate. In summary, Bayesian priors can be used to create estimators that deliver smooth estimates on small datasets and converge against ML with increasing amounts of data (Steck and Jaakkola 2002). This regularized approach to parameter estimation, called maximum a posterior (MAP) estimation, can be done in simple examples like coin-flipping, as well as in more complicated probabilistic models with many random and hidden variables.

To apply Bayesian regularization to the PFA, we redesign the model such that all parameters $A, B,$ and π are treated as random variables. We use Dirichlet distributions—defined by a parameter vector ρ, state-dependent parameter vectors s_k, and state- and event-dependent parameter vectors r_{ke}—since they are conjugate to categorical distributions (Barber 2011). Fig. 5 exhibits the final model, which we call the RegPFA.



$$
\begin{aligned}
P(Z_0) &\sim Categorical(\pi_0, \dots, \pi_K) \\
P(X_t | Z_t = k) &\sim Categorical(b_{k0}, \dots, b_{kE}), \\
&\quad \forall k \in \{1, \dots, K\} \\
P(Z_t | Z_{t-1} = k, X_{t-1} = e) &\sim Categorical(a_{ke0}, \dots, a_{keK}), \\
&\quad \forall k \in \{1, \dots, K\}, e \in \{1, \dots, E\} \\
P(\pi_1, \dots, \pi_K) &\sim Dirichlet(\rho_1, \dots, \rho_K) \\
P(b_{k1}, \dots, b_{kE}) &\sim Dirichlet(s_{k1}, \dots, s_{kE}), \forall k \in \{1, \dots, K\} \\
P(a_{ke1}, \dots, a_{keK}) &\sim Dirichlet(r_{ke1}, \dots, r_{keK}), \\
&\quad \forall k \in \{1, \dots, K\}, e \in \{1, \dots, E\}
\end{aligned}
$$

Fig. 5. Graphical models and formal description of the RegPFA model

Hyperparameters of Dirichlet distributions can be interpreted as pseudo-observations (Steck and Jaakkola 2002). For instance, when one estimates the probability b_{k0} of seeing event type 0 in state k, the Dirichlet prior adds $s_{k0} - 1$ pseudo-observations to the observations in the event log, making the strength of this prior equivalent to $\sum_{e=0}^{E} s_{ke} - E$. Combining all the priors yields a total pseudo-count of $pc = \sum_{k=1}^{K} \rho_k \sum_{e=1}^{E} s_{ke} \sum_{j=1}^{K} r_{kej} - K\big(1 + E(1 + K)\big)$.

To express the belief that everything is equally likely in the absence of data, we can use symmetric priors, setting all parameters of the Dirichlet priors to the same value: $\rho_k = s_{ke} = r_{kej} = hp$ and yielding a total pseudo-count of $pc = K\big(1 + E(1 + K)\big)(hp - 1)$. If we want to express a belief with strength equal to n observations, we must set $pc = n$, which yields the value to which we set all Dirichlet parameters: $hp = 1 + \frac{n}{K\left(1+E(1+K)\right)}$. This value is just a default setting from which users may want to deviate if, for instance, a user believes there is a state in which only one particular type of event can be observed, in which case the user could build this belief into the model by biasing one of the priors accordingly.

Parameter Estimation

The EM algorithm (Dempster et al. 1977) is the standard parameter estimation technique for probabilistic models with unobserved variables. Therefore, we derived an instantiation of the EM algorithm for the RegPFA model defined in Fig. 5 in order to do a MAP estimation of its parameters, as illustrated in Fig. 6. After initial parameters are defined, the algorithm iterates between the E-step and the M-step until it converges, which will happen because the log likelihood will improve in each iteration and cannot become infinitely large. Since the result is a local optimum that depends on the initial parameters, EM should be run multiple times with differing initial values (Moon 1996).

```
doEM(log, hp, K, E)
  params = initialize_params(K, E)
  do
    marginals  = E-step(log, params)
    params = M-step(marginals, hp)
  until convergence(params)
  return params
```

Fig. 6. EM algorithm as pseudo-code

The remainder of this section explains the four components of the algorithm and how we implemented them. We discuss the M-step first, followed by the E-step. For a more in-depth treatment of EM in general, we refer to textbooks like Bishop (2006).

Initial parameter selection: As the EM algorithm is an iterative optimization procedure that converges toward a local optimum, its result depends on the initial parameter values. In the absence of other information, the default is to choose initial values randomly. Researchers have studied how to find good initial values for the EM algorithm and have developed techniques that are superior to random initialization (e.g., Karlis and Xekalaki (2003)), but since these results cannot be transferred directly to new models, we implemented random generation and defer a discussion of other techniques to future work.

M-step: To implement the M-step's goal of improving parameter values, we must find updating equations for all parameters by starting with $\theta^{new} = \arg\max_{\theta}\left[\mathbb{E}_{Z|X,\theta^{old}} \ln P(X,Z|\theta) + \ln P(\theta)\right]$, the EM approach for MAP parameter estimation (Dempster et al. 1977), and applying Lagrangian multipliers to the optimization problem. Solving the optimization problem delivers the equations in Fig. 7 (bottom), which are defined using the terms also exhibited in Fig. 7 (top). Of these terms, the three *prior* terms are easily computable, as they depend only on the parameters of the Dirichlet distributions, which must be fixed prior to executing EM. However, the *data* terms require the computation of marginal distributions over the state variables given the data and the current parameters, which is what the E-step delivers.

First, we define *data* and *prior* terms representing the contributions from the event log and the assumptions of the user.

$$data_k = \sum_{c=1}^{C} P\left(Z_0^{(c)} = k \middle| X^{(c)}, \theta^{old}\right) \qquad prior_k = \rho_k - 1$$

$$data_{ke} = \sum_{c=1}^{C} \sum_{t \in T_e^{(c)}} P\left(Z_t^{(c)} = k \middle| X^{(c)}, \theta^{old}\right) \qquad prior_{ke} = s_{ke} - 1$$

$$data_{kej} = \sum_{c=1}^{C} \sum_{t \in T_e^{(c)}, t \neq 0} P\left(Z_{t-1}^{(c)} = k, Z_t^{(c)} = j \middle| X^{(c)}, \theta^{old}\right) \qquad prior_{kej} = r_{kej} - 1$$

In the equations above, $T_e^{(c)}$ denotes the set of points in time at which an event of type e is observed in instance $x^{(c)}$. $T^{(c)}$ denotes all points in time regardless of the event types. For instance, if $x^{(4)} = (2,2,1,3,2,4)$, then $T_2^{(4)} = \{0,1,4\}$, while $T^{(4)} = \{0,1,2,3,4,5\}$.

Using the terms above, the parameter updating equations are defined as follows:

$$\pi_k = \frac{data_k + prior_k}{\sum_{j=1}^{K}(data_k + prior_k)}; \; b_{ke} = \frac{data_{ke} + prior_{ke}}{\sum_{e=1}^{E}(data_{ke} + prior_{ke})}; \; a_{kej} = \frac{data_{kej} + prior_{kej}}{\sum_{j=1}^{K}(data_{kej} + prior_{kej})}$$

Fig. 7. Updating equations for the probabilistic model derived from the EM approach

E-step: The E-step's goal is to compute all marginal distributions required in the three *data* terms while treating all parameters θ as fixed at a certain value θ^{old}. The standard technique for this computation is *belief propagation* (Murphy 2012). For our model, the procedure is almost identical to the forward-backward algorithm used in the Baum-Welch algorithm for HMMs (Baum et al. 1970), which iterates all instances $x^{(c)}$ in the event log and passes messages forward and backward through the chain of hidden states Z to compute the marginal distributions. In terms of computational complexity, the E-step is the bottleneck in our version of EM. With belief propagation, its complexity is $O(|X| \times |K|^2)$ and thus linear in the total number of events in the event log, so it scales to large datasets. Details about belief propagation can be found in textbooks like Murphy (2012) and Bishop (2006). Murphy (2012) also discusses ways to speed up the E-step for cases in which the $|K|^2$ term becomes prohibitive. An evaluation of their applicability is deferred to future research.

Convergence criterion: Each iteration of EM improves $P(X|\theta)P(\theta)$, the quantity the algorithm optimizes, so convergence can be detected by monitoring the rate of improvement and stopping once that rate falls below a threshold δ, that is, $|\ln P(X|\theta^{new})P(\theta^{new}) - P(X|\theta^{old})P(\theta^{old})| < \delta$. Thresholds between 10^{-2} and 10^{-8} are common (Abbi et al. 2008).

Model Selection

The EM algorithm requires the user to specify a number of inputs (cf. Fig. 6). log is the data used for learning, which will always be known, as will E, the number of event types, since all events are recorded in the event log. However, the inputs K and hp require special treatment. K is unknown and not amenable to optimization, as it determines the size of the model's state space, while hp determines the strength of the regularization, but the appropriate value will be unclear. In the absence of additional information, we resort to a grid search over a range of values such that, for each combination of K and hp, a model is learned with EM on training data, and a separate set of validation data (not used for learning) allows one to decide which of the models fits best. We use cross entropy as a measure of model fit, as cross entropy is often used in grammatical inference to assess the quality of a language model (Rosenfeld 2000). The cross entropy of an EM result θ with respect to an event log log can be expressed as shown in Fig. 8 and interpreted as an estimate of the cross entropy of θ with respect to the true but unknown distribution of the data. The lower the cross entropy, the better the parameters θ are in assigning appropriate probabilities.

Cross entropy:

$$H(log, \theta) = -\frac{1}{|log|} \sum_{x^{(c)} \in log} \log_2 P\left(x^{(c)} \middle| \theta\right)$$

Akaike information criterion:

$$AIC(\theta) = -2 \ln P(X|\theta) + 2|\theta|$$

Heuristic information criterion:

$$HIC_t(\theta) = -2 \ln P(X|\theta) + 2|\theta|_t$$

Fig. 8. Model selection criteria

In addition to cross entropy, we can use penalized likelihood criteria, such as the Akaike information criterion (AIC) (Akaike 1998), to select optimal inputs. AIC, defined in Fig. 8, has been used extensively in the HMM learning literature (Celelux and Durant 2008). It penalizes the likelihood proportionate to the number of parameters, denoted by $|\theta|$, such that the lower the AIC score, the better the model. We also define a modified criterion based on our own intuition: For business processes, we expect large parts of the model parameters to be zero because, in most states, there will be only a few types of events possible to observe next, and after an event is observed, there will be only a few states to which the process can possibly move. To account for the expected sparsity, the modified criterion penalizes the number of parameters above a threshold, denoted as $|\theta|_t$, not the number of parameters $|\theta|$. We call this threshold the heuristic information criterion (HIC) and define it analogously to AIC in Fig. 8. All three methods of grid-search optimization discussed in this section are evaluated experimentally later in this paper.

The grid search procedure illustrated in Fig. 9 is called with arguments *grid_hp* and *grid_K*, which represent the sets of possible values for *hp* and K. The argument *modelScorer* is a function that can implement any of the equations' criteria discussed in this section.

```
doGridSerach(log, grid_hp, grid_K, E, modelScorer)
  bestModel = null
  for hp in grid_hp
    for K in grid_K
      model = doEM(log, hp, K, E)
      if modelScorer(model) < modelScorer(bestmodel)
        bestModel = model
  return bestModel
```

Fig. 9. Model selection procedure as pseudo-code

Design of the RegPFA Analyzer

A design goal of the RegPFA approach developed in the previous section is to ensure that the model structure is interpretable and understandable by domain experts, so we used probabilistic models of automata as a starting point. With RegPFA models that reflect the structure of automata, we can design the RegPFA Analyzer on top of the RegPFA Predictor.

The purpose of the RegPFA Analyzer is to provide visualizations of RegPFA models that domain experts who do not have expertise in probabilistic modeling can inspect and understand. A direct way to construct a visualization of a RegPFA is to build an automaton with K states and K^2E transitions. Each transition corresponds to a product of RegPFA parameters $b_{ke}a_{kej}$, the estimated probability of moving from state k to state j when an event of type e is observed in state k. This process delivers a fully connected automaton with transitions among all states and for all event types, but it also delivers unreadable and uninformative results unless it is applied to small toy examples.

Transition probabilities can be exploited to simplify the visualization. Clearly, all transitions with zero probability should be excluded, which removes all transitions that lead out of the final state, as there can be no exit from the final state. However, apart from those that are fixed because of the PFA modifications inspired by workflow nets, transition probabilities will usually not be zero because of the Bayesian regularization that is introduced to combat overfitting. If pseudo-observations are distributed symmetrically, as happens by default, probabilities may be small but will be positive unless technical subtleties like limited precision of floating point numbers produce zero values.

This discussion demonstrates that providing users with all of the information contained in a RegPFA model is not practical; some loss of information is necessary to ensure comprehensibility, which requirement motivates a threshold-based abstraction of a RegPFA model. We define a threshold, ε, adjustable by the user, and cut out all transitions with probabilities lower than the threshold. Then a user can change ε interactively until the user finds a visualization with the desired degree of detail.

Fig. 10 shows an exemplary visualization of a RegPFA model fitted to a real-world event log of a problem-management process that will be used later in the evaluation. The visualization of the RegPFA corresponding to $\varepsilon = 0.1$ (left) has a unique starting state of *s1* indicated by a dangling, unlabeled transition entering it, and a unique ending state of *s12* indicated by the thick line surrounding it. Transitions are labeled with their event type and probability. Fig. 10 (right) shows the effect of raising the threshold to $\varepsilon = 0.2$. The right part is cut out as it is entered through a transition with probability 0.16, which is too low to be included in the more abstract visualization.

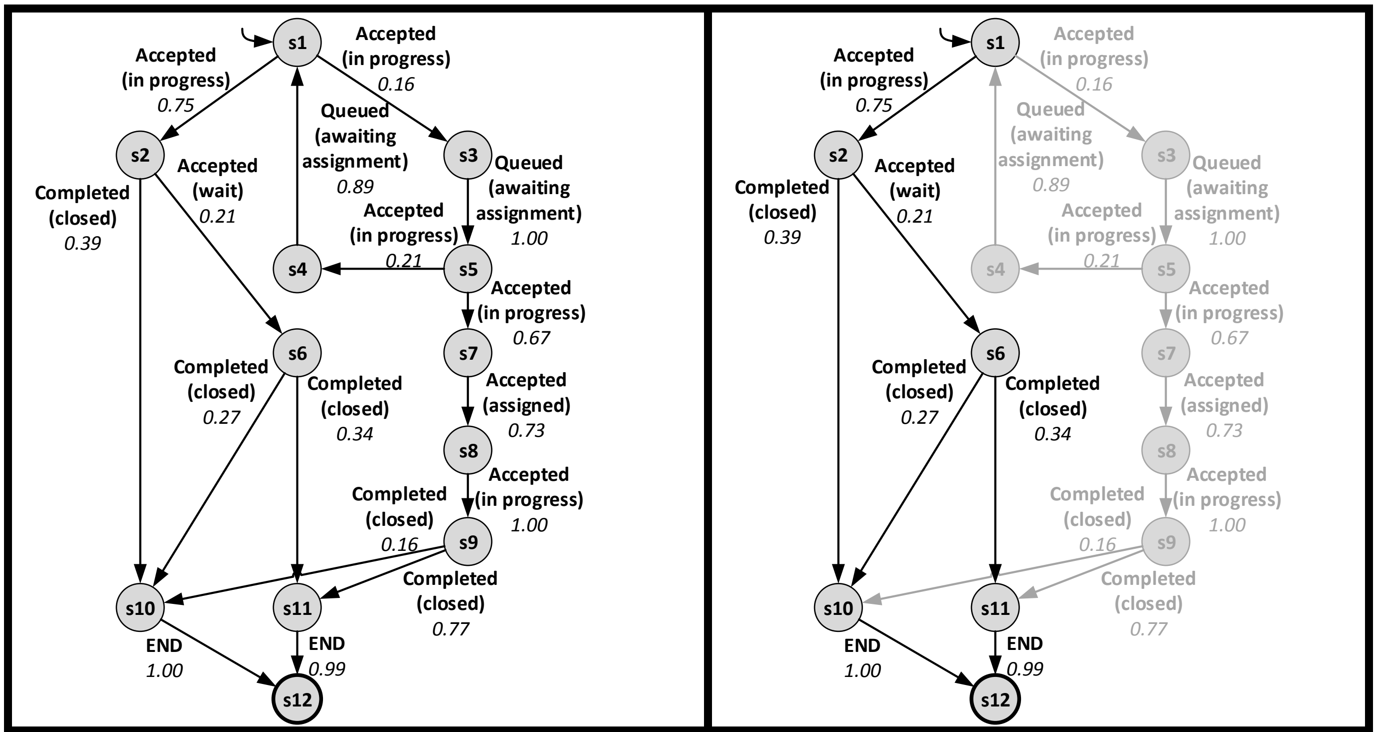


Fig. 10. RegPFA model visualized with abstraction levels of $\boldsymbol{\varepsilon = 0.1}$ (left) and ($\boldsymbol{\varepsilon = 0.2}$) (right)

To guide the choice of ε, we derive a reasonable default value. Assume a state k in the predictive model that provides no information as to which event type will be observed or to which state the process will move. This situation can be expressed by setting all probabilities b_{ke} of seeing an event type e in state k to the same value, $b_{ke} = 1/\, E$. Similarly, all probabilities a_{kej} of moving to any state j after seeing s are set to $a_{kej} = 1/K$. As a result, all transitions have probability $b_{ke}a_{kej} = 1/(KE)$. As no transition is more likely than any other, we cannot select a subset to discard. In real applications, the probabilities will never be equal but may all be close. If some of the probabilities increase, the probabilities of others must drop so the probability distribution sums to 1, creating motivation for formulating the threshold relative to $\frac{1}{KE}$, which we call the pruning ratio pr. This ratio translates to a threshold according to the formula $\varepsilon =$

$1/pr \cdot 1/(KE)$, such that transitions that are pr times as unlikely as would be expected if the state contained no information are discarded.

While the automaton constructed by means of this pruning approach may already be interpretable by humans, we add a second step to improve comprehensibility. Automata with high degrees of concurrent behavior may be extremely large, although representing them as equivalent Petri nets can lead to more compact models (Murata 1989) and facilitate understanding. Being able to handle concurrency is one of the main goals in process mining, so many discovery algorithms are based on Petri net representations (van der Aalst et al. 2011a). One discovery algorithm, the *region miner* (van der Aalst et al. 2010), applies a two-step procedure to discover a Petri net from an event log: After creating an automaton in the first step, the second step uses a Petri net synthesis algorithm (Cortadella et al. 1997) to construct a Petri net that is equivalent to the automaton. By integrating the second step into our approach, we can construct a Petri net representation of a predictive model (pruned according to ε).

EVALUATION

To determine whether the RegPFA Predictor and the RegPFA Analyzer meet their goals, we conducted experiments with synthetic data and real-world data. We generated synthetic data by randomly executing models of business processes and recording the events observed during the course of execution, so we could use numerous process models and generate large quantities of data. We used real-world data collected from organizations to achieve external validity. Our experiments pursue five goals:

***G1: Evaluation of the model-selection criteria*:** When designing the RegPFA Predictor, we defined criteria that would facilitate our ability to choose the best one from a set of candidate probabilistic models. We identified two criteria from the literature and defined another based on our own insight. *The evaluation goal is to determine how well the model selection criteria work and to assess each criterion's performance relative to that of the others.*

***G2: Evaluation of regularization effectiveness*:** When designing the RegPFA Predictor, we modified the PFA by adding Bayesian regularization in order to mitigate problems related to small, unrepresentative samples. *The evaluation goal is to quantify to what extent overfitting to small samples can be avoided by means of Bayesian regularization*.

***G3: Evaluation of performance in the presence of challenging constructs*:** The literature of process mining discusses a number of control-flow constructs with which some process-discovery algorithms have problems dealing when the algorithm's language bias is too limited to accommodate the problematic behavior (van der Aalst 2011). As the RegPFA Predictor is based on an automata-like model, it can generally account for challenging constructs, but the EM algorithm could fail. *The evaluation goal is to determine whether challenging constructs affect the RegPFA Predictor's performance*.

***G4: Evaluation of performance in predictive modeling problems*:** The RegPFA Predictor should perform well when it is applied to predictive modeling problems in the context of business processes. *The evaluation goal is to compare the RegPFA Predictor's predictive performance to suitable benchmarks*.

***G5: Evaluation of process discovery with the RegPFA Analyzer*:** Visualizing predictive models with the RegPFA Analyzer facilitates process discovery. *The evaluation goal is to compare the RegPFA Analyzer's performance with state-of-the-art process-discovery techniques.*

We conducted three experiments to address these five goals: The first experiment addresses goals G1, G2, and G3; the second experiment addresses goals G1, G2, and G4; and the third experiment addresses goal G5. We explain these experiments after presenting the data used in the experiments.

Data

We used thirty-nine Petri nets used in previous studies to evaluate process-discovery algorithms for the experiments with synthetic data (de Medeiros 2006; de Weerdt et al. 2012). Appendix A lists these models and their properties. We embedded the models into simulation software to generate event logs of varying sizes, expressed in terms of the number of process instances. The simulation procedure began by giving all models a dedicated starting state, after which we executed the Petri net until we ran into a deadlock, indicating process completion. At each step in the execution, the transition to fire was chosen at random from all enabled transitions. Upon firing, we recorded the transition's label and added it to the event log. After termination, we ran a new simulation until we generated a sufficient number of process instances. De Weerdt et al. (2012) use an additional Petri net called *driverslicensel*, a variation of the *driverslicense* model found in de Medeiros (2006). When evaluating G5, we also report results for logs generated from this model in order to ensure comparability to de Weerdt et al.'s (2012) results.

We took our real-world event logs from the 2012 and 2013 BPI challenges (van Dongen 2012, 2013). From the 2012 challenge, we gathered anonymized data from a Dutch financial institute's loan or overdraft application process—262,000 events in 13,087 instances that stem from three sub-processes: one that tracks the states of the application, another that tracks the states of the offer, and a third that tracks the states of work items associated with the application. Events from the third sub-process are classified further into type *schedule* (work items scheduled for execution), type *start* (work items on which work has started), or type *complete* (work items on which work has been completed). We filtered the event log to retain only events of type *complete*.

From the 2013 challenges we gathered two logs from Volvo IT Belgium that describe incident- and problem-management processes. The incident-management log contains 3,777 instances with 36,730 events of 11 types, and the problem-management log contains 744 instances with 4,045 events of 5 types.

Evaluation Results

Experiment 1

Our first experiment, designed to address goals G1, G2, and G3, was based on synthetic event logs generated from the 39 Petri nets. A large sample (1,000 instances) and a small sample (50 instances) were generated for each, and all were split into a training set log_{train} (70%) and a validation set log_{val} (30%). The EM algorithm was applied to the training set, while the validation set was used to calculate $H(log_{val}, \theta)$, the score of a model defined by θ on the validation set. Finally, a third event log, log_{test}, with 10,000 instances was generated to evaluate $H(log_{test}, \theta)$, the performance of an EM result θ on yet unseen data.

We used several parameters for the grid search: For the number of states, we used the grid $K_{grid} = [2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 25, 30, 40, 50, 75]$ as a set of candidate values. The strength of regularization, expressed as a relative fraction of the total number of events in the logs, was optimized over the grid $[0.00, 0.05, 0.10, 0.15, 0.20, \ldots, 0.75]$ and ranged from no regularization to severe regularization. We set the convergence threshold for EM to $\delta = 10^{-4}$. For each combination of K and hp, we performed EM five times and retained only the best result to avoid bad local optima. We configured the HIC criterion to include only parameters larger than 0.05.

After applying EM to the $2 \cdot 39$ event logs in both the large sample and the small sample, we applied our three grid search optimization methods to select an optimal model. This process delivered $2 \cdot 39 \cdot 3$ optimal models, chosen with respect to the three methods. For each triple, we calculated $H(log_{test}, \theta)$ for all three models and recorded the lowest cross entropy, determining that one method was better than the others if it chose the models with the lowest cross entropy. We counted how often each method made the best choice and reported the results in Table I. (The counts did not sum to thirty-nine per row, as two or more methods can agree.) Detailed results on the overall performance and the performance of the model selection criteria are reported in Appendix A.

Table I. Number of times a model selection method chose the best model

	Validation set	**AIC**	**HIC$_{0.05}$**
Large sample	16	11	24
Small sample	14	11	20

As Table I shows, the HIC criterion performed best, regardless of the sample size, so it is better than the AIC criterion, as expected. It is also better than evaluating cross entropy on held-out validation data, probably because the estimates are not good at these sample sizes.

We tried various degrees of regularization in the grid search and chose the best by means of model selection criteria. To determine how well the grid search avoids overfitting, we defined overfitting as choosing from all models under consideration one that assigns zero probability to any one of the 10,000 process instances of our large event log, log_{test}. In these cases, $H(log_{test}, \theta)$ is infinite. We counted how often this happened for each of the selection methods (cf. Table II) and how often the model that performed best on log_{test} overfitted.

Table II. Number of times a model selection method chose a model that assigned zero probability to possible behavior

	Validation set	**AIC**	**HIC$_{0.05}$**	**Optimal**
Large sample	7	7	4	0
Small sample	14	5	5	0

We found that counts for the optimal choice are zero, regardless of the sample size, so there was always a sufficiently regularized model that did not overfit. However, none of the model selection criteria consistently avoided selecting an overfitted model. In line with the results of Table I, HIC performed best on both samples, as it chose the fewest overfitting models; the AIC criterion was slightly worse but comparable to HIC; and using the validation set delivered comparable results only when it was applied to the large sample. These results are in line with our expectations, as evaluating cross entropies of very small validation sets (15 process instances for the small sample) is likely to produce bad estimates. The results, shown in Tables I and Table II, together indicate that the HIC criterion is effective in choosing models that fit the data well and that generalize to yet unseen process instances. The increase in the AIC's number of overfitted models could be due to a disadvantage when AIC is applied to probabilistic models, as we expect that most transitions have very low likelihood.

Next, we addressed G3 by investigating the effects of complex control flow constructs. To avoid confusing EM performance with model-selection performance, we considered only the models that scored highest with respect to $H(log_{test}, \theta)$—that is, the model θ with the best generalization performance. De Medeiros (2006) designs the thirty-nine Petri nets to contain models that exhibit various kinds of properties, but not all process-discovery algorithms can deal with these properties because of limitations in the underlying representational bias. One type of these constructs is *loops* (particularly of small length), that is, a process that can move to a state in which it has been before. Another is *non-local choice*, where a process has states in which the type of the next event depends on events seen long before, not on directly related events. *Concurrency* is present if more than one type of event is executed in parallel, that is, without any constraints on their order. Finally, *duplicates* are conceptually different kinds of events that cannot be distinguished in the event log, as they appear to be of the same event type. We flagged the models accordingly (cf. Appendix A) based on de Medeiros' (2006) characterization.

While the underlying probabilistic representation we use in our approach can handle all of the challenging constructs, learning the constructs with the EM optimization method may still create issues, so we identified differences in performance. For each challenging construct, we grouped the models with respect to the presence or absence of the construct and used Python's SciPy package to apply the Kruskal-Wallis test, which determines whether two or more samples are independent (Kruskal and Wallis 1952). In our case, the test reveals whether the challenging constructs influence performance; if so, corresponding p-values should comply with a significance level. Since none of the p-values (cf. Table III) is significant even at the 0.1 level, we cannot conclude that these constructs affect the performance of the EM approach.

Table III. p-values of the Kruskal Wallis test

	Loop	**Non-local choice**	**Concurrency**	**Duplicates**
Large sample	0.40	0.46	0.11	0.14
Small sample	0.19	0.27	0.93	0.27

Experiment 2

Experiment 2 primarily addresses the performance of the RegPFA Predictor on prediction problems, so the experiment is based on the five real-world event logs. As the actual business processes for these event logs are not known, performance cannot be measured in absolute terms, so we need suitable benchmark prediction techniques to serve as baseline predictors in a comparative analysis. Analogous to the grammatical-inference competition discussed earlier, we implemented a simple table look-up approach and a prediction approach based on n-grams (Verwer et al. 2014). The table look-up approach, which we call *History*, uses as input the entire sequence of events seen so far, selects from the training set all process instances that begin with this sequence and estimates the probability that the next event will be of a given event type by counting how many of the selected sequences continue with an event of the given type and dividing this count by the number of process instances that begin with the selected sequences; that is, the prediction is based on the relative frequency with which the given event type occurs after a particular sequence. The event type with the highest probability is predicted to be the next one. n-gram predictors work in a similar way, but they take into account only a sliding window of size n instead of the entire sequence. To estimate the probability of the next event's being a given event type, n-grams use as input the last $n-1$ events seen so far and return the relative frequency with which event sequences in the training set continue with the given event type. A detailed description of these benchmark prediction techniques is shown in Appendix B. Our experiments evaluated n-gram predictors for $n \in \{2,3,4,5,6\}$. As in experiment 1, we split the event logs into training (50%), validation (25%), and test sets (25%). Settings for the EM algorithm are also the same as in experiment 1.

We evaluated the predictors based on four performance metrics, all of which we computed with the test set. One metric is the *cross entropy* of the predictor with respect to the test set. We defined cross entropy in Fig. 8 for the RegPFA model described in Fig. 5. This definition applies to any other model that defines a probability distribution over sequences, and our benchmark predictors clearly define such a distribution. The other three metrics are based on two prediction problems: the problem of predicting what the next event type will be and the problem of predicting whether the next event will be of a given type. Iterating the events in the test set and counting how often a predictor predicts the correct event type gives rise to the metric of prediction *accuracy*, which is the fraction of correct predictions. As for the problem of predicting whether the next event will be of a given type, we measure *sensitivity* (the rate of true positives) and *specificity* (the rate of true negatives). Since there is one such classifier for each type of event, the final performance metrics are averages over all event types, weighted by the frequencies of the event types.

Table IV shows the results of the experiment, including the PFA-based predictors, the history-based predictor, and the n-gram-based predictors for each event log. We considered that a predictor was working best when it produced the highest values for accuracy, sensitivity, and specificity and the lowest value for cross entropy.

Table IV. Performance statistics of the five prediction approaches

Event log	Predictor	Accuracy	ØSensitivity	ØSpecitivity	H
BPI2012_W	EM_HIC	0.685	0.558	0.950	12.810
	EM_AIC	0.719	0.578	0.955	11.183
	EM_Val	0.718	0.582	0.955	10.385
	History	0.615	0.458	0.938	Infinity
	2-gram	0.721	0.589	0.956	9.438

	3-gram	0.726	0.553	0.956	9.328
	4-gram	0.727	0.587	0.957	Infinity
	5-gram	0.728	0.588	0.957	Infinity
	6-gram	0.726	0.584	0.957	Infinity
BPI2012_A	EM_HIC	0.801	0.723	0.980	3.093
	EM_AIC	0.769	0.658	0.977	3.393
	EM_Val	0.796	0.720	0.980	3.014
	History	0.801	0.723	0.980	2.839
	2-gram	0.751	0.629	0.975	3.472
	3-gram	0.778	0.681	0.978	3.146
	4-gram	0.801	0.723	0.980	2.839
	5-gram	0.801	0.723	0.980	2.839
	6-gram	0.801	0.723	0.980	2.839
BPI2012_O	EM_HIC	0.809	0.646	0.973	4.588
	EM_AIC	0.811	0.647	0.973	4.513
	EM_Val	0.811	0.647	0.973	4.513
	History	0.687	0.514	0.957	Infinity
	2-gram	0.750	0.591	0.964	5.356
	3-gram	0.811	0.647	0.973	4.180
	4-gram	0.811	0.647	0.973	4.146
	5-gram	0.811	0.647	0.973	4.129
	6-gram	0.811	0.649	0.974	Infinity
BPI2013_Incidents	EM_HIC	0.586	0.268	0.958	15.910
	EM_AIC	0.709	0.377	0.973	12.501
	EM_Val	0.714	0.383	0.974	12.041
	History	0.569	0.308	0.961	Infinity
	2-gram	0.621	0.346	0.965	Infinity
	3-gram	0.633	0.368	0.966	Infinity
	4-gram	0.635	0.377	0.967	Infinity
	5-gram	0.631	0.378	0.966	Infinity
	6-gram	0.624	0.375	0.966	Infinity
BPI2013_Problems	EM_HIC	0.627	0.497	0.934	9.399
	EM_AIC	0.690	0.521	0.945	7.231
	EM_Val	0.686	0.519	0.944	7.265
	History	0.585	0.467	0.927	Infinity
	2-gram	0.690	0.521	0.945	7.133
	3-gram	0.699	0.564	0.948	Infinity
	4-gram	0.686	0.553	0.946	Infinity
	5-gram	0.680	0.543	0.944	Infinity
	6-gram	0.665	0.530	0.942	Infinity

Experiment 1 revealed that the HIC criterion works best, but experiment 2 refuted this result, as with respect to the cross entropy metric, model selection with the validation set outperformed HIC. With respect to the metrics derived from exemplary prediction problems, HIC delivered the best result in only one of the five cases, for event log *BPI2012_A*, while using AIC and cross entropy evaluation delivered the best result in two of the five cases. History performs worst in almost all cases; only on event log *BPI2012_A* competitively. This result confirms expectations since the entire event sequence seen so far can easily produce overfitted probability distributions. History's good performance on *BPI2012_A* can be explained by *BPI2012_A*'s simple process structure, as most of the 13,087 process instances in this event log are identical; there are only sixteen unique process instances.

The RegPFA-based predictors effectively avoid overfitting, as shown in the finite cross entropy values in all five cases. By contrast, the history-based predictor almost always overfits (except for *BPI2012_A*). n-grams also produce infinite cross entropies if n is large enough (except for *BPI2012_A*), so overfitting can be avoided by keeping n low. The performance metrics suggest that neither the RegPFA-based predictors nor n-grams consistently outperform the other, as both methods perform similarly on the event logs *BPI2012_A* and *BPI2012_O*, n-grams work best on *BPI2012_W* and *BPI2013_Problems*, and the PFA-based predictor scores best on *BPI2013_Incidents*. This result suggests that the techniques that work best depend on the event log and the underlying business process.

To determine why some techniques work better than others on specific event logs, we performed a qualitative in-depth evaluation based on two of the five event logs. First, we used the RegPFA Analyzer—that is, our visualization approach applied to the RegPFA-based predictive model—to analyze *BPI2013_Incidents*, for which the RegPFA-based approach outperforms n-grams. After applying a pruning ratio of $pr = 0.1$, we generated the model shown in Fig. 11, for which we used Petri net synthesis for the visualization. The low pruning ratio removed all state transitions with probabilities smaller than about 10 percent and generated an abstract view of this complex process.

This process handles incidents at an IT service desk, and the Petri net suggests that the process usually starts with a sequence of two events of type "*Accepted In progress*," which moves the token from place *p1* to place *p3*, indicating productive work on the customer's problem. The incident may then be completed directly in a call ("*Completed In call*") or marked as resolved ("*Completed Resolved*", token to *p4*). In the latter case, the process waits for the customer's approval before the incident is closed ("*Completed Closed*"). If a support team working on the incident is unable to resolve it—which can happen at the beginning of the process (token in *p1*) or after working on the incident (token in *p3*)—an event of type "*Queued Awaiting assignment*" follows, which indicates that a support team wants to transfer the incident to another team, and moves the token to *p5*. The new support team then works on the incident ("*Accepted*

In progress”, token to *p6*) and resolves it (“*Completed Resolved*”, token to *p4*), transfers it to yet another team (“*Queued Awaiting assignment*”, token back to *p5*), puts it on hold (“*Accepted Assigned*”, token back to *p5* or to *p7*), or waits for user input (“*Accepted Wait - user*”, token to *p8*). With the token either in place *p7* or place *p8*, complex iterations of work on the incident, interrupted by waits for one or more user inputs, can follow. Either this process resolves the incident (“*Completed Resolved*”, token from *p8* to *p4*) or a new support team is put in charge (“*Queued Awaiting assignment*”, token from *p8* to *p5*).

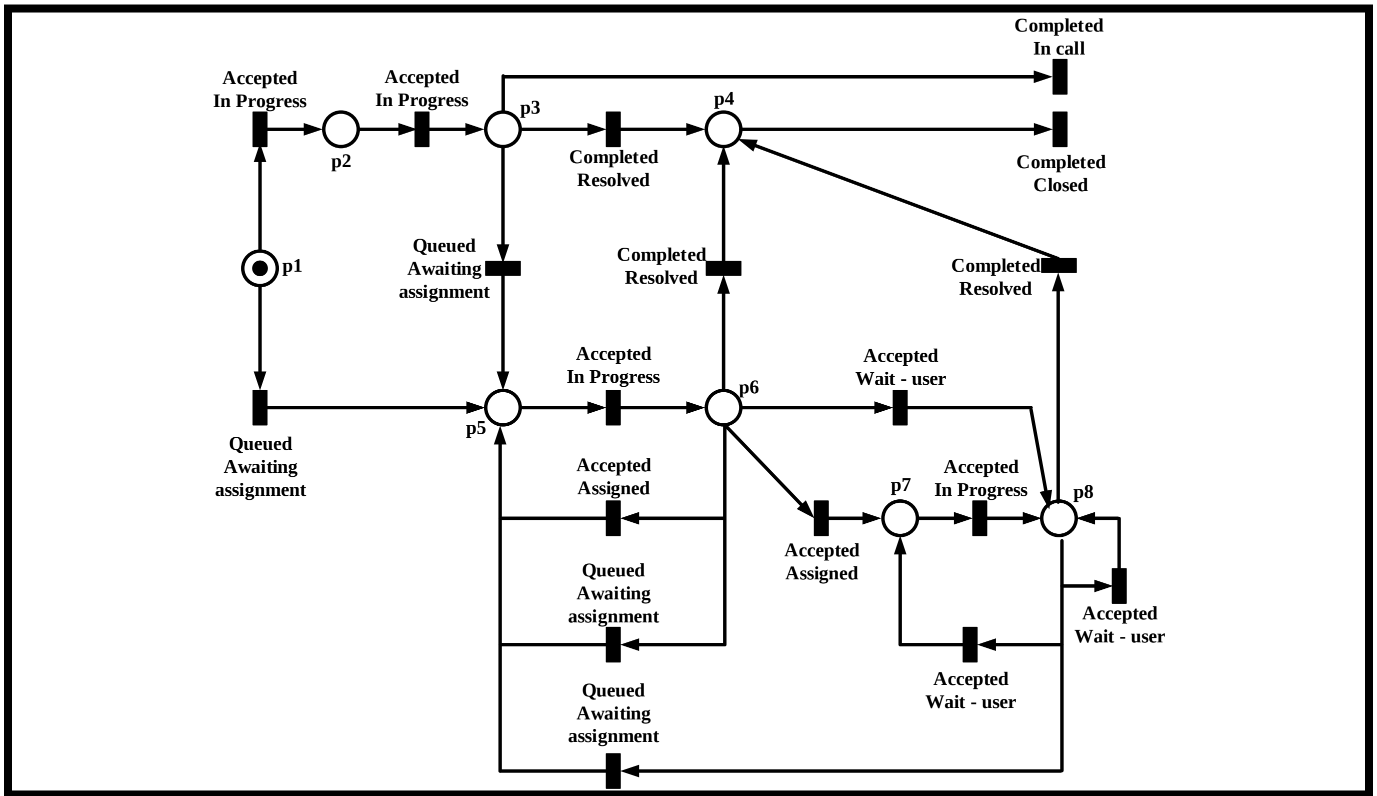


Fig. 11. Petri net visualization of a predictive model for “BPI2013_Incidents”

To understand why PFA-based predictive models can outperform n-grams on this process, consider a process in the state in which there is a token in *p5*. As Fig. 11 shows, the only significant type of event observable in this state is “*Accepted In progress.*” It is the predictive model’s goal to estimate the probability, so a 3-gram model, for instance, estimates the probability depending on the types of the last two events seen before the process entered the state. In this Petri net, four such pairs of events might be observed: “*Accepted In progress*” and “*Queued Awaiting assignment*”; no event and “*Queued Awaiting assignment*”; “*Accepted In progress*” and “*Accepted Assigned*”; and “*Accepted Wait - user*” and “*Queued*

Awaiting assignment." Thus, the 3-gram divides the available data into four parts and estimates four probabilities, one for each way of entering the state, and so fails to identify the state. Since each of the estimates considers only part of the data, the prediction is likely worse than had a single state been identified.

The RegPFA Predictor imposes no predefined structure on the states. If it converges to a good local optimum, the E-step of the EM algorithm identifies all process instances that move through a state, regardless of how they entered it. Thus, the PFA-based predictor can use all evidence in the data to estimate a single probability, which will result in a more reliable estimate than that of the 3-gram model, as the estimate is based on a larger part of the data.

To compare the insights from this discussion with results from *BPI2012_A*, a process that describes a loan application, we use the RegPFA Analyzer to generate the Petri net shown in Fig. 12. Since we use the standard pruning ratio of 1.5, resulting in the threshold of 0.3 percent, this Petri net is not as abstract as that in Fig. 11, yet it is sufficiently simple to be easily readable because of the high degree of structure and the low level of noise. On this event log, RegPFA did not outperform the n-grams.

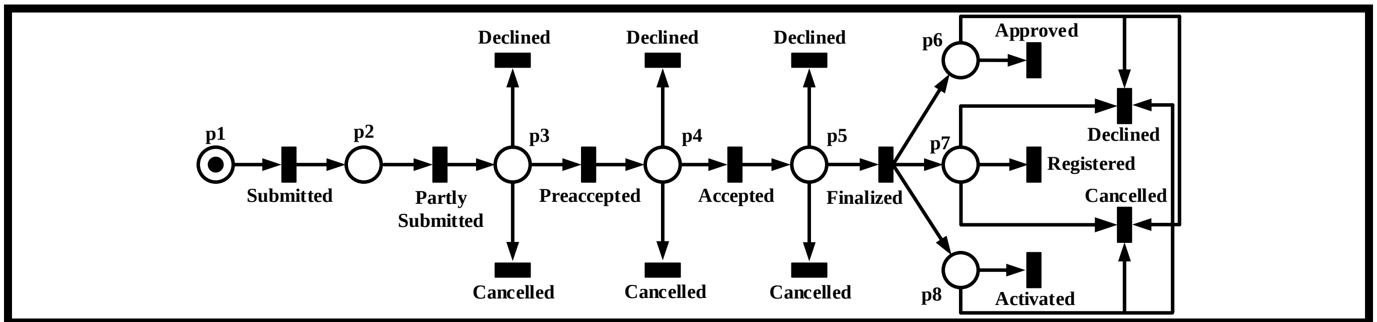


Fig. 12. Petri net visualization of a predictive model for event log "BPI2012_A"

The process usually begins with submission of an application ("*Submitted*"), after which the process goes through a number of checks, each updating the application's status, represented by the events "*Partly Submitted*," "*Preaccepted*," "*Accepted*," and "*Finalized*." The process is linear and may terminate at any time once the application is partly submitted, either with cancellation by the applicant ("*Cancelled*") or rejection by the bank ("*Declined*"). Only after finalizing the application ("*Finalized*", tokens into *p6, p7*, and *p8*) may the three events "*Approved*," "*Registered*," and "*Activated*" be observed in any order.

This process's structure helps to clarify why the RegPFA Predictor cannot beat n-grams: the type of event observed last identifies almost all states of the Petri net. For instance, knowing that an event of type

"*Preaccepted*" was observed last is sufficient to know that there is a token in *p4*, so an n-gram can use all available data to estimate probabilities in this state. The only situation in which the type of the last event is insufficient is when the process ends with "*Approved*," "*Registered*," and "*Activated*" in any order. For instance, to identify process termination, the three last events must be known in order to know whether one is missing, so the performance metrics of the n-grams increase with n, but only until $n = 4$. Analogous to the previous example, the 4-gram has to estimate the probability of termination six times since there are six possible orders of the three final events. Therefore, although the PFA-based approach could have an advantage, in this example the two approaches' performances are equal.

Two observations arise from this qualitative analysis:

- The RegPFA Predictor can estimate probabilities more exactly than n-grams can if the process has states that can be entered in more than one way—that is, if different event sequences can lead to the same state—as the RegPFA Predictor can make more efficient use of limited data.
- However, the RegPFA Predictor does not *necessarily* outperform n-grams when the process enters a state in more than one way, as sufficient data may be available to compensate for the n-grams' representational shortcoming, or the EM algorithm may converge to a bad local optimum and fail to identify these states.

Experiment 3

We conducted experiment 3 to address G5's goal of evaluating the RegPFA Analyzer. We used the results from de Weerdt et al. (2012), which provides a comprehensive evaluation of state-of-the-art process-discovery algorithms, as a reference by which to compare our approach. De Weerdt et al. (2012) apply process-discovery algorithms to event logs with 300 process instances generated from nineteen of the thirty-nine models we have used so far and use an event log with 350 process instances generated from the *driverslicensel* model. They apply to twenty event logs metrics developed in process mining to measure the quality of each algorithm's result. The measures can be categorized into the dimensions *recall* and *precision*, where recall reflects "how much behavior present in the log is captured by the model," and *precision* measures “whether a mined process model does not underfit the behavior present in the log” (de Weerdt et al. 2012, p. 660).

Fitness, one of the recall measures de Weerdt et al. (2012) use, is based on replaying an event log with a discovered Petri net (Rozinat and van der Aalst 2008), which means firing transitions so the event log is generated. If doing so is possible, the Petri net has fitness 1, and if there are excess tokens or tokens are missing, fitness is reduced as low as 0. One of the precision measures de Weerdt et al. (2012) use is *advanced behavioral appropriateness* (Rozinat and van der Aalst 2008), which searches for variability in both the event log and the Petri net, defined as pairs of event types that sometimes follow or precede each other. *Advanced behavioral appropriateness,* which measures to what degree the variability found in the discovered Petri net is also found in the event log, like *fitness,* ranges from 0 to 1. Since both measures are implemented in ProM (Rozinat and van der Aalst 2008), we used them for our evaluation.

For each Petri net, we generated ten event logs of the same size as those de Weerdt et al. (2012) use in order to ensure that our results did not differ from those of de Weerdt et al. (2012), despite our using of a different random sample. Then we applied the RegPFA Analyzer to discover a Petri net and measured *fitness* and *advanced behavioral appropriateness* using ProM’s conformance checker plugin. We always measured the same scores for the ten runs, except for the fitness score of *hFig6p19*, for which values ranged between 0.965 and 0.970. These observations indicate that the results are stable and can be compared to the values de Weerdt et al. (2012) report.

The results indicate that the RegPFA Analyzer can likely compete with state-of-the-art process-discovery techniques. Its *fitness* score of 0.998 equals to that of the genetic miner and is bested only by the ILP miner’s 1.0 (which was expected, as the ILP miner generates Petri nets with perfect fitness by design (van der Werf et al. 2009)). The RegPFA Analyzer scores better (0.908) than all the techniques evaluated by de Weerdt et al. (2012) in *advanced behavioral appropriateness*, where α++ performed best (0.879). The comparison and detailed results on each model’s performance are reported in Appendix A.

DISCUSSION

Contributions

We proposed five goals for the evaluation and three experiments, each of which addressed one or more of the five goals. Table V provides a summary of the findings. Experiment 2 compared the RegPFA Predictor to n-grams, which were used because of their popularity as baseline models in grammatical inference and their relationship to prediction techniques in process mining. While several other predictive approaches in business process mining define the current state of a process based on the last n events (e.g., van der Aalst et al.'s (2011b) time-prediction technique) and allow abstractions like interpreting the last n events not as a sequence but as a multi-set or set, they are similar to n-grams. Therefore, the RegPFA Predictor constitutes a contribution to the process-mining literature by demonstrating that an EM-based approach to estimating automata can be a useful alternative to the existing n-gram-based techniques. Experiment 2's qualitative analysis sheds light on the conditions under which the one or the other approach is advantageous.

Table V. Summary of the findings from the experiments

Goal	EXP	Findings
G1	E1, E2	The results are contradictory, as HIC performed best on synthetic data, AIC performed worst, and using a validation set performed in the middle. The experiments with real-world data do not confirm these results but suggest that using the validation set is best. This result reinforces the results of de Weerdt et al. (2012), who also identified differences between real and synthetic event logs.
G2	E1, E2	The results indicate that overfitting is effectively avoided, as infinite cross entropies are rare on synthetic data if the HIC or the AIC criterion is applied and the sample size is large enough; the same is true for using a validation set. No overfitting was observed on real-world data.
G3	E1	The results confirm the expectation that control flow constructs that are challenging do not affect performance.
G4	E2	The results show that the RegPFA Predictor *can* outperform the approaches against which it was benchmarked, but it does not always do so. Therefore, a general recommendation cannot be given, and the best approach must be chosen based on the concrete event log.
G5	E3	The results show that the RegPFA Analyzer can compete with state-of-the-art process-discovery techniques. In a comparative benchmark on synthetic data, the RegPFA Analyzer scored second best in terms of *fitness* and best in terms of *advanced behavioral appropriateness*.

In the same way, the RegPFA Analyzer contributes to the body of knowledge about process mining by providing a novel technique for process discovery. Pruning a fitted RegPFA based on a user-defined threshold allows one to make seamless adjustments to the desired level of abstraction like zooming in and out of a geographical map, a rare but desirable functionality in process mining (van der Aalst 2009). Consequently, the RegPFA is applicable to even highly unstructured processes whose event logs are far from complete. An example of this capability is the Petri net in Fig. 11, which illustrates an easily comprehensible model on a high level of abstraction. Most importantly, the language bias of the RegPFA Predictor is comparatively weak, as it assumes that the process is representable by any kind of automaton. Since the probabilistic learning algorithm aims to identify the most probable of an infinite number of possible process models, there is no need for a strong language bias that narrows the explanations of the data to a single process model. Experiment 3 showed that the RegPFA Analyzer performs similar to state-of-the-art process-discovery algorithms on standard benchmark processes, even though its language bias is weaker.

A third contribution to process mining is that the RegPFA Predictor's generalization performance can be evaluated easily on a part of the event log that is held out, as demonstrated in this paper. In process mining, such evaluations are rare (Goedertier et al. 2009) since they cannot be applied directly in a non-probabilistic setting. While only positive examples are given, which favors overly generalized models if fitness on the held-out data is optimized (van der Aalst 2011), this problem vanishes with a probabilistic technique since cross entropy optimization does not suffer from this deficiency.

Our experiments used entropy-based measures to evaluate and compare models, an approach that is in line with the grammatical inference literature (de la Higuera 2010; Rosenfeld 2000). Entropy-based measures are popular since they can be used with a wide range of probabilistic models and since they are a natural performance metric. While improving these measures is typically accompanied by better performance in predictive modeling applications, the literature holds many counterexamples (Rosenfeld 2000), and experiment 2 adds some counterexamples to the list. The lowest cross entropy does not always imply best performance on the exemplary prediction problems, so we contribute to the grammatical inference literature by confirming the importance of using application-specific performance metrics.

Limitations

Our evaluation of grammatical inference techniques is limited to an EM-based estimation of automata, so future work should focus on evaluating other techniques and comparing the results to those presented in this paper. We considered only basic n-gram models in our evaluation, although there are many variations of this technique. For instance, a context-dependent variation of n-gram length (Kepler et al. 2012) might improve the results, as could the application of regularization approaches (Zhai and Lafferty 2004).

Similarly, the literature offers some extensions to the EM algorithm that may prove useful for use on BPM data if sophisticated strategies for choosing initial parameter values could be developed—for example, Karlis and Xekalaki (2003) compare strategies for Gaussian and Poisson mixture models—so future research could investigate how existing process-discovery algorithms can be used to construct initial values. There are also sophisticated strategies for avoiding local optima, such as Ueda and Nakano (1998), who use a technique they call “deterministic annealing” instead of randomly restarting EM multiple times, so how to transfer this technique to the RegPFA Predictor could be the subject of future research. Some prediction techniques can also outperform EM-based approaches under certain conditions, such as using collapsed Gibbs sampling (Shibata and Yoshinaka 2012), a technique from the Bayesian inference class. We did not use these techniques, as they do not support visualization, but they may be useful for prediction.

So far we did not investigate how the RegPFA technique could be combined with other state-of-the-art data analysis techniques in order to increase their overall predictive performance. For instance, classification techniques can investigate business data that resides in the context of a business process instance’s execution. Thus, a stage like an applicant’s class of credit worthiness could be predicted from case-related business data like a participant’s age and then inform a process-related prediction next to the process history.

So far the question concerning to what extent we can build process models that comply with the workflow patterns van der Aalst et al. (2003a) introduce is unresolved. In general, the RegPFA model can represent any behavior expressible with state machines having designated start and end states, as we specified the model with the notion of workflow nets in mind. Consequently, we expect most workflow patterns to be expressible. However, some workflow patterns, e.g., the cancellation pattern, describe behavior that corresponds to rather complex state spaces that will quickly outgrow the capabilities of the EM estimation technique even if implemented on Big Data platforms. A thorough evaluation of workflow pattern mining feasibility with EM-based RegPFA estimation is deferred to future research.

Implications for Research on Big Data Analytics

The presented technique provides an illustrative example of new data-driven approaches to BPM that will be facilitated through Big Data. The total runtime of our experiments was about two months on a conventional computer and without parallelization, because they required for almost 100,000 runs of the EM algorithm. However, the advances in distributed and parallel computing technologies that go by the name "Big Data" will make it possible to calculate the probabilistic models suggested by our approach more quickly because the computational tasks the technique requires can be executed in parallel. Despite the time required for the current research, at least with the stop threshold we chose for the EM Algorithm, RefPFA performs satisfactory. As its runtime is linear in the size of the event log, it scales to even larger process logs. In practical applications, mining can be executed periodically, not every day, so runtimes of minutes or even hours could be acceptable. The prediction itself—the question of "what event will be next"—can be computed in milliseconds, a satisfactory time even for real-time applications.

The results offered in this article have further implications for research related to the generalities of Big Data analytics:

Improving comprehensibility: The extant literature emphasizes the importance of predictive models' comprehensibility so managers will have confidence in and use them (e.g., Martens and Provost 2014). This research included comprehensibility as a design goal, and a large portion of the research activities has sought probabilistic models that users without deep technical knowledge can interpret and understand. Therefore, our work may serve researchers in the field of Big Data as an example of how to approach the goal of comprehensibility.

Increasing the comprehensiveness of prediction approaches: Our technique could be one component of an overarching, comprehensive approach to predictive business process analytics, so future research related to Big Data might seek to improve and integrate it with other data analysis techniques. Dissemination of our solution is encouraged, as the RegPFA artifact's implementation is publicly available.

Providing an example of how to adopt the methodological apparatus from machine learning: The machine learning field provides mature instruments that can facilitate the construction of innovative techniques that are beneficial in various application contexts. The present research demonstrates the relevance and power of IS as an intersection discipline by fitting existing instruments into a new context (BPM) and shows that Big Data requires joint efforts of basic and applied research across disciplines.

Novel applications and use cases: Our technique will facilitate applications that support novel BPM use cases in which human process workers and responsible or computational agents receive information on the likely future behavior of single process instances at an early stage of process execution. These cases are starting points for future Big Data research. Researchers could explore new ways to exploit prediction in certain business contexts and focus on the challenges related to privacy, protection of individual rights, and other ethical issues that are associated with the use of the suggested technique in organizations.

CONCLUSION

In this paper we designed a predictive modeling approach for business process event data that is based on established research from the field of grammatical inference. The application of these techniques is novel in the field of business process mining. Using experiments with synthetic and real-world data, we showed that the RegPFA Predictor can be effective, and we investigated the circumstances under which it outperforms suitable benchmarks. We also showed that probabilistic models can be adopted without scarifying process-discovery capabilities and that the RegPFA Analyzer, despite being based on a weak language bias, can compete with state-of-the-art techniques from process mining. We hope that, by making a case for probabilistic modeling in process mining, we achieved a solution that is relevant and long-lasting and that addresses what Chen and Storey (2012) describe as "the challenge that our discipline continues to face" (p. 1186). We trust that future research will explore its advantages further.

BIBLIOGRAPHY

Abbi, R., El-Darzi, E., Vasilakis, C., and Millard, P. 2008. “Analysis of Stopping Criteria for the EM Algorithm in the Context of Patient Grouping According to Length of Stay,” in *Proceedings of the 4th International IEEE Conference on Intelligent Systems (IS’08)*, Varna, Bulgaria, pp. 9–14.

Aggarwal, C. C., and Yu, P. S. 2009. “A Survey of Uncertain Data Algorithms and Applications,” *IEEE Transactions on Knowledge and Data Engineering* (21:5), pp. 609–623.

Akaike, H. 1998. “Information Theory and an Extension of the Maximum Likelihood Principle,” in Parzen, E., Tanabe, K., Kitagawa, G. (eds.), *Selected Papers of Hirotugu Akaike*, New York, NY, USA: Springer, pp. 199–213.

Arnold, V., Clark, N., Collier, P. A., Leech, S. A., and Sutton, S. G. 2006. “The Differential Use and Effect of Knowledge-Based System Explanations in Novice and Expert Judgement Decisions,” *MIS Quarterly* (30:1), pp. 79–97.

Barber, D. 2011. *Bayesian Reasoning and Machine Learning*, Cambridge, UK: Cambridge University Press.

Baum, L. E., Petrie, T., Soules, G., and Weiss, N. 1970. “A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains,” *The Annals of Mathematical Statistics* (41:1), pp. 164–171.

Bishop, C. M. 2006. *Pattern Recognition and Machine Learning*, New York, NY, USA: Springer.

Bose, R. P. J. C., and van der Aalst, W. M. P. 2012. “Process Diagnostics using Trace Alignment: Opportunities, Issues, and Challenges,” *Information Systems* (37:2), pp. 117–141.

Blum, T., Padoy, N., Feußner, H., and Navab, N. 2008. “Workflow Mining for Visualization and Analysis of Surgeries,” *International Journal of Computer Assisted Radiology and Surgery* (3:5), pp. 379-386.

Carrasco, R. C., and Oncina, J. 1994. “Learning Stochastic Regular Grammars by Means of a State Merging Method,” in Carrasco, R. C., Oncina, J. (eds.), *Proceedings of the 2nd International ICGI Colloquium on Grammatical Interference and Applications (ICGI-94)*, Lecture Notes in Computer Science (Vol. 862), Berlin, Germany: Springer, pp. 139–152.

Celelux, G., and Durant, J. 2008. “Selecting Hidden Markov Model State Number with Cross-Validated Likelihood,” *Computational Statistics* (23:4), pp. 541–564.

Chater, N., and Manning, C. D. 2006. “Probabilistic Models of Language Processing and Acquisition,” *Trends in Cognitive Sciences* (10:7), pp. 335–344.

Chen, H., and Storey, V. C. 2012. “Business Intelligence and Analytics: From Big Data to Big Impact,” *MIS Quarterly* (36:4), pp. 1165–1188.

Chen, M., Shiwen, M., and Liu, Y. 2014. “Big Data: A Survey,” *Mobile Networks and Applications* (19:2), pp. 171-209.

Cook, J. E., and Wolf, A. L. 1998. “Discovering Models of Software Processes from Event-based Data,” *ACM Transactions on Software Engineering and Methodology (TOSEM)* (7:3), pp. 215–249.

Cortadella, J., Kishinevsky, M., Kondratyev, A., Lavagno, L., and Yakovlev, A. 1997. "Petrify: A Tool for Manipulating Concurrent Specifications and Synthesis of Asynchronous Controllers," *IEICE Transactions on Information and Systems* (E80-D:3), pp. 315–325.

de la Higuera, C. 2005. "A Bibliographical Study of Grammatical Inference," *Pattern Recognition* (38:9), pp. 1332–1348.

de la Higuera, C. 2010. *Grammatical Inference. Learning Automata and Grammar*, Cambridge, UK: Cambridge University Press.

de Medeiros, A. K. 2006. *Genetic Process Mining*, Ph.D. Thesis. Eindhoven, The Netherlands: Technische Universiteit Eindhoven.

de Weerdt, J., de Backer, M., Vanthienen, J., and Baesens, B. 2012. "A Multi-Dimensional Quality Assessment of State-of-the-Art Process Discovery Algorithms Using Real-Life Event Logs," *Information Systems* (37:7), pp. 654–676.

Dempster, A., Laird, N., and Rubin, D. 1977. "Maximum Likelihood from Incomplete Data via the EM Algorithm," *Journal of the Royal Statistical Society. Series B (Methodological)* (39:1), pp. 1–22.

Ferreira, D. R., and Gillblad, D. 2009. "Discovering Process Models from Unlabelled Event Logs," in Dayal, U., Eder, J., Koehler, J., Reijers, H. A. (eds.), *Proceedings of the 7th International Conference on Business Process Management (BPM 2009)*, Lecture Notes in Computer Science (Vol. 5701), Berlin, Germany: Springer, pp. 143–158.

Folino, F., Guarascio, M., and Pontieri, L. 2012. "Discovering Context-Aware Models for Predicting Business Process Performances," in Meersman, R., Panetto, H., Dillon, T., Rinderle-Ma, S., Dadam, P., Zhou, X., Pearson, S., Ferscha, A., Bergamaschi, S., Cruz, I. F. (eds.), *On the Move to Meaningful Internet Systems: OTM 2012*, Lecture Notes in Computer Science (Vol. 7565), Berlin, Germany: Springer, pp. 287–304.

Gao, J., and Johnson, M. 2008. "A Comparison of Bayesian Estimators for Unsupervised Hidden Markov Model POS Taggers," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 08)*, Waikiki, Honolulu, Hawaii, Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 344-352.

Goedertier, S., Martens, D., Vanthienen, J., and Baesens, B. 2009. "Robust Process Discovery with Artificial Negative Events," *Journal of Machine Learning Research* (10), pp. 1305–1340.

Gregor, S., and Hevner, A. R. 2013. "Positioning and Presenting Design Science Research for Maximum Impact," *MIS Quarterly* (37:2), pp. 337–355.

Grigori, D., Casati, F., Castellanos, M., and Dayal, U. 2004. "Business Process Intelligence," *Computers in Industry* (53:3), pp. 321–343.

Günther, C. W., and van der Aalst, W. M. P. 2007. "Fuzzy Mining – Adaptive Process Simplification Based on Multi-Perspective Metrics," in Alonso, G., Dadam, P., Rosemann, M. (eds.), *Proceedings of the 5th International Conference on Business Process Management (BPM 2007)*, Lecture Notes in Computer Science (Vol. 4714), Berlin, Germany: Springer, pp. 328–343.

Hastie, T., Tibshirani, R., and Friedman, J. 2009. *The Elements of Statistical Learning - Data Mining, Inference, and Prediction*, (2nd ed.) New York, NY, USA: Springer.

Herbst, J., and Karagiannis, D. 2004. “Workflow Mining with InWoLvE,” *Computers in Industry* (53:3), pp. 245–264.

Hopcroft, J. E., Motwani, R., and Ullman, J. D. 2009. *Introduction to Automata Theory, Languages, and Computation* (3rd ed.), Cambridge, UK: Pearson.

Hulden, M. 2012. “Treba: Efficient Numerically Stable EM for PFA*,”* in *Proceedings of the 11th International Conference on Grammatical Inference*, September 5-8, 2012, University of Maryland, College Park, USA, Journal of Machine Learning Research - Workshop and Conference Proceedings (Vol. 21), pp. 249–253.

Janiesch, C., Matzner, M., and Müller, O. 2012. “Beyond Process Monitoring: A Proof-of-Concept of Event-Driven Business Activity Management,” *Business Process Management Journal* (18:4), pp. 625–643.

Jeong, H., Biswas, G., Johnson, J., and Howard, L. 2010. "Analysis of Productive Learning Behaviors in a Structured Inquiry Cycle Using Hidden Markov Models." In Baker, R.S.j.d., Merceron, A., and Pavlik Jr., P.I. (Eds.), *Proceedings of the 3rd International Conference on Educational Data Mining*, Pittsburgh, PA, USA, pp. 81-90.

Karlis, D., and Xekalaki, E. 2003. “Choosing Initial Values for the EM Algorithm for Finite Mixtures,” *Computational Statistics & Data Analysis* (41:2), pp. 577–590.

Kayande, U., de Bruyn, A., Lilien, G. L., Rangaswamy, A., and van Bruggen, G. H. 2009. “How Incorporating Feedback Mechanisms in a DSS Affects DSS Evaluations,” *Information Systems Research* (20:4), pp. 527–546.

Kepler, F. N., Mergen, S. L. S., and Billa, C. Z. 2012. “Simple Variable Length N-grams for Probabilistic Automata Learning*,”* in *Proceedings of the 11th International Conference on Grammatical Inference*, September 5-8, 2012, University of Maryland, College Park, USA, Journal of Machine Learning Research - Workshop and Conference Proceedings (Vol. 21), pp. 254–258.

Kim, A., Obregon, J., and Jung, J. 2014. “Constructing Decision Trees from Process Logs for Performer Recommendation,” in Lohmann, N., Song, M., Wohed, P. (eds.), *Business Process Management Workshops*, Lecture Notes in Business Information Processing (Vol. 171), Berlin, Germany: Springer, pp. 224–236.

Koller, D., and Friedman, N. 2009. *Probabilistic Graphical Models: Principles and Techniques, Foundations, Adaptive Computation and Machine Learning*. Cambridge, MA: The MIT Press.

Kruskal, W.H., Wallis, W.A. 1952. "Use of Ranks in One-Criterion Variance Analysis," *Journal of the American Statistical Association* (47:260), pp. 583-621.

Lakshmanan, G. T., Shamsi, D., Doganata, Y. N., Unuvar, M., and Khalaf, R. 2015. “A Markov Prediction Model for Data-Driven Semi-Structured Business Processes,” *Knowledge and Information Systems (42:1)*, pp. 97-126.

Li, J., Liu, D., and Yang, B. 2007. “Process Mining: Extending α-Algorithm to Mine Duplicate Tasks in Process Logs,” in Chang, K. C.-C., Wang, W., Chen, L., Ellis, C. A., Hsu, C.-H., Tsoi, A. C., Wang, H. (eds.), *Advances in Web and Network Technologies, and Information Management*, Lecture Notes in Computer Science (Vol. 4537), Berlin, Germany: Springer, pp. 396–407.

Lilien, G. L., Rangaswamy, A., van Bruggen, G. H., and Starke, K. 2004. "DSS Effectiveness in Marketing Resource Allocation Decisions: Reality vs. Perception," *Information Systems Research* (15:3), pp. 216–235.

Lund, S., Manyika, J., Nyquist, S., Mendonca, L., and Ramaswamy, S. 2013. "Game Changers: Five Opportunities for US Growth and Renewal," McKinsey Global Institute Report, (available at http://www.mckinsey.com/~/media/McKinsey/dotcom/Insights/Growth/US%20game%20changers/MGI_Game_changers_US_growth_and_renewal_Full_report.ashx).

Maggi, F.M., di Francescomarino, C., Dumas, M., and Ghidini, C. 2014. "Predictive Monitoring of Business Processes." In Jarke, M., Mylopoulos, J., Quix, C., Rolland, C., Manolopoulos, Y., Mouratidis, H., Horkoff, J. (eds.), *Advanced Information Systems Engineering*. Lecture Notes in Computer Science (Vol. 8484), Berlin, Germany: Springer, pp. 457–472.

Martens, D., and Provost, F. 2014. "Explaining Data-Driven Document Classifications," *MIS Quarterly* (38:1), pp. 73–100.

Moon, T. K. 1996. "The Expectation-Maximization Algorithm," *IEEE Signal Processing Magazine* (13:6), pp. 47–60

Murata, T. 1989. "Petri Nets: Properties, Analysis and Applications," *Proceedings of the IEEE* (77:4), pp. 541–580.

Murphy, K. 2012. *Machine Learning: A Probabilistic Perspective*, Cambridge, MA, USA: The MIT Press.

Norvig, P. 2011. "On Chomsky and the Two Cultures of Statistical Learning," (available at http://norvig.com/chomsky.html).

Nowak, M., Komarova, N. L., and Niyogi, P. 2002. "Computational and Evolutionary Aspects of Language," *Nature* (417), pp. 611–617.

Provost, F., and Fawcett, T. 2013. *Data Science for Business: What you need to know about data mining and data-analytic thinking*, Sebastopol, CA, USA: O'Reilly and Associates.

Rabiner, L. R. 1989. "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *Proceedings of the IEEE* (77:2), pp. 257–286.

Ron, D., Singer, Y., and Tishby, N. 1998. "On the Learnability and Usage of Acyclic Probabilistic Finite Automata," *Journal of Computer and System Sciences* (56:2), pp. 133–152.

Rosenfeld, R. 2000. "Two Decades of Statistical Language Modeling: Where Do We Go From Here?," *Proceedings of the IEEE* (88:8), pp. 1270–1278.

Rozinat, A., and van der Aalst, W. M. P. 2008. "Conformance Checking of Processes Based on Monitoring Real Behavior," *Information Systems* (33:1), pp. 64–95.

SAP 2007. "Standardized Technical Architecture Modeling Conceptual and Design Level," (available at http://www.fmc-modeling.org/download/fmc-and-tam/SAP-TAM_Standard.pdf).

Shibata, C., and Yoshinaka, R. 2012. "Marginalizing Out Transition Probabilities for Several Subclasses of PFAs," in Heinz, J., de la Higuera, C., and Oates, T. (eds.) *Proceedings of the 11th International Conference on Grammatical Inference*, September 5-8, 2012, University of Maryland, College Park, USA, Journal of Machine Learning Research - Workshop and Conference Proceedings (Vol. 21), pp. 259–263.

Shmueli, G., and Koppius, O. R. 2011. “Predictive Analytics in Information Systems Research,” *MIS Quarterly* (35:3), pp. 553–572.

Steck, H., and Jaakkola, T. 2002. “On the Dirichlet Prior and Bayesian Regularization,” in *Advances in Neural Information Processing Systems* (15), pp. 1441–1448.

Stolcke, A. 1994. *Bayesian Learning of Probabilistic Language Models*, Ph.D. Thesis. Berkeley, CA, USA: University of California.

Tiwari, A., Turner, C. J., and Majeed, B. 2008. “A Review of Business Process Mining: State-of-the-Art and Future Trends,” *Business Process Management Journal* (14:1), pp. 5–22.

Turner, C. J., Tiwari, A., Olaiya, R., and Xu, Y. 2012. “Business Process Mining: From Theory to Practice,” *Business Process Management Journal* (18:3), pp. 493–512.

Ueda, N., and Nakano, R. 1998. “Deterministic Annealing EM Algorithm,” *Neural Networks* (11:2), pp. 271–282.

van der Aalst, W. M. P. 1998. “The Application of Petri Nets to Workflow Management,” *Journal of Circuits, Systems and Computers* (8:1), pp. 21–66.

van der Aalst, W. M. P. 2009. “TomTom for Business Process Management (TomTom4BPM),” in van Eck, P., Gordijn, J., Wieringa, R. (eds.), *Advanced Information Systems Engineering*, Lecture Notes in Computer Science (vol. 5565), Berlin, Germany: Springer, pp. 2–5.

van der Aalst, W. M. P. 2011. *Process Mining: Discovery, Conformance and Enhancement of Business Processes*, Berlin, Germany: Springer.

van der Aalst, W. M. P. 2013. “Business Process Management: A Comprehensive Survey,” ISRN Software Engineering (2013), pp. 1–37.

van der Aalst, W. M. P., Adriansyah, A., Alves de Medeiros, A. K., Arcieri, F., and et al. 2011a. “Process Mining Manifesto,” in Daniel, F., Barkaoui, K., Dustdar, S. (eds.), *Business Process Management Workshops*, Lecture Notes in Business Information Processing (Vol. 99), Berlin, Germany: Springer, pp. 169–194.

van der Aalst, W. M. P., Pesic, M., and Song, M. 2010b. “Beyond Process Mining: From the Past to Present and Future,” in Pernici, B. (ed.), *Advanced Information Systems Engineering*, Lecture Notes in Computer Sciences (Vol. 6051), Berlin, Germany: Springer, pp. 38–52.

van der Aalst, W. M. P., Rubin, V., Verbeek, H. M. W., van Dongen, B. F., Kindler, E., and Günther, C.W. 2010. “Process Mining: A Two-Step Approach to Balance Between Underfitting and Overfitting,” *Software and Systems Modeling* (9:1), pp. 87–111.

van der Aalst, W. M. P., Schonenberg, M. H., and Song, M. 2011b. “Time Prediction Based on Process Mining,” *Information Systems Journal* (36:2), pp. 450–475.

van der Aalst, W. M. P., ter Hofstede, A.H.M., Kiepuszewski, B., and Barros, A.P. 2003a. “Workflow Patterns,” *Distributed and Parallel Databases* (14:3), pp. 5–51.

van der Aalst, W. M. P., van Dongen, B., Herbst, J., Maruster, L., Schimm, G., and Weijters, A. 2003b. "Workflow Mining: A Survey of Issues and Approaches," *Data & Knowledge Engineering* (47:2), pp. 237–267.

van der Aalst, W. M. P., Weijters, T., and Maruster, L. 2004. "Workflow Mining: Discovering Process Models from Event Logs," *IEEE Transactions on Knowledge and Data Engineering* (16:9), pp. 1128–1142.

van der Werf, J. M. E. M., van Dongen, B., and Hurkens, C. 2009. "Process Discovery Using Integer Linear Programming," *Fundamenta Informaticae* (94:3-4), pp. 387–412.

van Dongen, B. F. 2012. "BPI Challenge 2012 – Event Log of a Loan Application Process," Eindhoven University of Technology (doi: 10.4121/uuid:3926db30-f712-4394-aebc-75976070e91f).

van Dongen, B. F. 2013. "BPI Challenge 2013 – Logs of Volvo IT incident and problem management," Eindhoven University of Technology (doi: 10.4121/uuid:a7ce5c55-03a7-4583-b855-98b86e1a2b07).

Verwer, S., Eyraud, R., and de la Higuera, C. 2014. "PAutomaC: A Probabilistic Automata and Hidden Markov Models Learning Competition," *Machine Learning* (96:1-2), pp. 129–154.

Vidal, E., Thollard, F., de la Higuera, C., Casacuberta, F., and Carrasco, R. C. 2005. "Probabilistic Finite-State Machines - Part I," *IEEE Transactions on Pattern Analysis and Machine Intelligence* (27:7), pp. 1013–1025.

Weber, P., Bordbar, B., and Tino, P. 2013. "A Principled Approach to Mining from Noisy Logs Using Heuristics Miner," in *Proceedings of the 2013 IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*, 16-19 April 2013, Singapore, pp. 119-126.

Weijters, A., van der Aalst, W. M. P., and de Medeiros, A. K. 2006. "Process Mining with the Heuristics Miner Algorithm," No. 166, *BETA Working Paper Series*, Eindhoven, The Netherlands, pp. 1–34. (available at http://wwwis.win.tue.nl/~wvdaalst/publications/p314.pdf).

Wen, L., van der Aalst, W. M. P., Wang, J., and Sun, J. 2007. "Mining Process Models with Non-Free-Choice Constructs," *Data Mining and Knowledge Discovery* (15:2), pp. 145–180.

Zhai, C., and Lafferty, J. 2004. "A Study of Smoothing Methods for Language Models Applied to Information Retrieval," *ACM Transactions on Information Systems (TOIS)* (22:2), pp. 179–214.

Appendix A

SYNTHETIC MODELS AND THEIR CHARACTERISTICS

Table VI summarizes the various characteristics of the synthetic models used in the experiments, including the number of event types, the size of the state space, whether a challenging construct is contained (loops, duplicates, non-local choice, and concurrency), and the entropy of the process defined by the model (estimated based on a sample of size 10,000). The original models may contain either duplicate tasks (two conceptually different transitions with the same label) or invisible tasks (transitions that have no label, as their firing is not recorded in the event log). We transformed all invisible transitions to duplicates such that, when there was an invisible task *i* in the original model, we added duplicates for all transitions *t* that, when fired, enable the invisible transition. These duplicates emulate the combined firing of *t* and *i*. Since we do not distinguish between duplicates and invisible tasks, we combined this category.

Table VI. Petri net models and their characteristics

Model	Events	States	Loops	Non-local choice	Concurrency	Duplicates	Est. entropy
1	4	3	1	0	0	0	4.74
1Skip	4	6	1	0	1	1	5.97
2	4	4	1	0	0	0	1.97
2Optional	4	4	1	0	0	0	1.99
2Skip	4	5	1	0	0	1	2.02
a1	7	7	1	0	1	0	6.02
a10Skip	10	11	0	0	1	1	2.58
a12	12	13	0	0	1	0	2.25
a2	11	14	1	0	1	0	8.07
a5	5	6	1	0	1	0	2.35
a6nfc	5	7	0	1	1	0	1.50
a7	7	10	0	0	1	0	3.56
a8	8	8	0	0	1	0	1.92
betaSimplified	11	18	0	1	0	1	2.00

bn1	41	40	0	0	0	0	2.00
bn2	41	40	1	0	0	1	4.00
bn3	41	40	1	0	0	1	9.02
Choice	10	7	0	0	0	0	4.00
driversLicense	7	8	0	1	0	0	1.00
flightCar	6	8	0	0	1	1	1.92
herbstFig3p4	10	11	1	0	1	0	4.45
herbstFig5p19	3	6	0	0	1	1	2.51
herbstFig5p1AND	3	4	0	0	0	1	1.00
herbstFig5p1OR	6	8	0	0	1	1	1.00
herbstFig6p10	9	13	1	0	1	1	3.63
herbstFig6p18	5	5	1	0	0	1	6.77
herbstFig6p25	19	19	1	0	0	1	6.20
herbstFig6p31	7	7	0	0	0	1	2.00
herbstFig6p33	8	8	0	0	0	1	1.92
herbstFig6p34	10	15	1	0	1	1	6.44
herbstFig6p36	10	16	0	1	0	0	1.00
herbstFig6p37	14	51	0	0	1	0	9.25
herbstFig6p38	5	11	0	0	1	1	2.16
herbstFig6p39	5	11	0	0	1	1	3.42
herbstFig6p41	14	18	0	0	1	0	3.50
herbstFig6p42	12	20	0	0	1	1	3.95
herbstFig6p45	6	14	0	0	1	0	3.45
herbstFig6p9	5	7	0	0	0	1	2.00
parallel5	7	34	0	0	1	0	6.91

DETAILED RESULTS OF THE EXPERIMENTS WITH SYNTHETIC DATA

Tables VII and VIII document in detail the results of the experiments with synthetic data. As discussed in the section on experiment 2, we fitted a RegPFA to the training set (70%) of each of the event logs and measured the result's quality by computing the cross entropy with respect to the large test event log (10,000 process instances). The tables show the increase in cross entropy relative to the entropy of the actual entropy listed for each event log in Table VI. Therefore, the entries in Tables VII and VIII represent the increase in entropy when the fitted model is used instead of the true model that generated the data. We report the performance with respect to each model's selection criterion and the "optimal" performance, that is, the performance that could have been achieved had the model selection delivered the best of all candidate models.

Table VII lists results for the large event logs (700 process instances in the training set and 300 process instances in the validation set).

Table VII. Experiments on synthetic data with large event logs

Model	Validation Set	AIC	$HIC_{0.05}$	Optimal
1	0.01	0.01	0.67	0.01
1Skip	1.02	1.88	2.71	1.00
2	0.00	0.00	0.00	0.00
2Optional	0.00	0.00	0.00	0.00
2Skip	0.00	0.10	0.00	0.00
a1	27.13	inf	27.01	20.76
a10skip	inf	34.45	28.23	27.04
a12	27.50	404.19	inf	19.70
a2	691.59	47.60	47.60	43.29
a5	0.01	0.00	0.00	0.00
a6nfc	12.51	42.62	5.87	3.76
a7	27.10	19.35	25.20	17.38
a8	inf	inf	18.30	9.27
betaSimplified	45.40	inf	43.11	36.63
bn1	0.00	26.27	0.00	0.00
bn2	inf	84.52	74.12	72.45

bn3	inf	145.69	111.74	107.00
Choice	24.83	inf	23.00	15.51
driversLicense	0.00	0.00	0.00	0.00
flightCar	138.49	26.68	26.68	21.91
hFig3p4	41.86	82.51	inf	38.48
hFig5p1AND	1.02	1.17	1.17	1.02
hFig5p1OR	0.00	0.00	0.00	0.00
hFig5p19	0.00	0.00	0.00	0.00
hFig6p10	40.68	inf	38.33	31.17
hFig6p18	inf	inf	15.11	14.39
hFig6p25	22.36	36.56	27.70	20.06
hFig6p31	421.27	421.27	16.50	11.12
hFig6p33	inf	inf	inf	13.76
hFig6p34	55.33	94.33	46.34	38.65
hFig6p36	1.00	1.00	1.00	1.00
hFig6p37	56.11	39.69	52.52	35.34
hFig6p38	98.71	362.66	inf	7.10
hFig6p39	22.67	25.92	20.51	17.58
hFig6p41	10.01	34.44	18.08	8.38
hFig6p42	34.54	33.59	33.27	25.41
hFig6p45	inf	16.21	15.40	12.06
hFig6p9	4.10	4.07	3.99	3.87
parallel5	0.10	3.21	0.25	0.10
# best choice	**16**	**11**	**24**	
# inf	**7**	**7**	**4**	**0**

Table VIII lists results for the small event logs (35 process instances in the training set and 15 process instances in the validation set).

Table VIII. Experiments on synthetic data with small event logs

Model	Validation Set	AIC	HIC$_{0.05}$	Optimal
1	inf	inf	10.84	3.57
1Skip	0.91	1.86	2.46	0.91
2	0.13	0.01	0.01	0.01
2Optional	0.32	6.07	0.01	0.01
2Skip	0.00	0.76	0.20	0.00

a1	25.16	29.00	24.08	13.59
a10skip	inf	27.55	25.42	22.50
a12	inf	19.33	7.14	5.18
a2	inf	51.33	36.88	31.09
a5	0.06	0.06	0.06	0.06
a6nfc	0.21	1.59	0.31	0.21
a7	25.80	9.09	1.74	1.14
a8	0.11	3.31	0.11	0.11
betaSimplified	inf	38.29	inf	28.02
bn1	inf	132.57	69.24	66.80
bn2	inf	198.52	73.48	69.74
bn3	inf	216.81	132.54	129.03
Choice	inf	15.77	inf	13.35
driversLicense	0.00	1.00	0.00	0.00
flightCar	0.07	7.89	0.07	0.07
hFig3p4	inf	36.64	inf	28.81
hFig5p1AND	17.66	12.30	16.37	11.89
hFig5p1OR	0.01	0.01	0.01	0.01
hFig5p19	0.00	5.12	0.00	0.00
hFig6p10	39.29	inf	34.31	31.51
hFig6p18	inf	107.95	14.93	11.98
hFig6p25	65.92	67.37	inf	59.24
hFig6p31	0.04	7.04	0.04	0.04
hFig6p33	inf	inf	22.87	18.66
hFig6p34	inf	52.81	inf	40.79
hFig6p36	1.03	7.03	1.04	1.03
hFig6p37	57.12	52.10	52.75	41.00
hFig6p38	18.96	80.53	22.53	6.20
hFig6p39	18.90	9.16	16.66	9.16
hFig6p41	0.23	11.75	0.24	0.23
hFig6p42	15.87	23.21	14.17	10.62
hFig6p45	5.94	inf	5.26	4.84
hFig6p9	inf	inf	15.65	11.56
parallel5	8.61	12.74	11.42	6.66
# best choice	**14**	**10**	**20**	
# inf	**14**	**5**	**5**	**0**

Table IX shows the fitness and advanced behavioral appropriateness scores for all event logs used to evaluate the RegPFA Analyzer.

Table IX. Experiments on process discovery

Model	**Fitness**	**Advanced Behavioral Appropriateness**
|2|	1.00	0.69
|2|Optional	1.00	1.00
|2|Skip	1.00	0.59
a10skip	1.00	1.00
a12	1.00	1.00
a5	1.00	1.00
a6nfc	1.00	1.00
a7	1.00	1.00
a8	1.00	1.00
betaSimplified	1.00	0.65
Choice	1.00	1.00
driversLicense	1.00	1.00
driversLicensel	1.00	0.88
hFig3p4	1.00	0.74
hFig5p19	0.97	1.00
hFig6p18	1.00	0.81
hFig6p31	1.00	1.00
hFig6p36	1.00	0.80
hFig6p38	1.00	1.00
hFig6p41	1.00	1.00
Ø	**0.998**	**0.908**

Table X compares the experiment's results to the other algorithms' scores that Weerdt et al. (2012) report.

Table X. Comparison with Fitness and Advanced Behavioral Appropriateness Scores reported in de Weerdt et al. (2012)

Algorithm	**Fitness**	**Advanced behavioral appropriateness**
ProbabilisticMiner	0.998	**0.908**
AGNES-Miner	0.995	0.813
α+	0.969	0.873
α++	0.984	0.879
DT Genetic Miner	0.996	0.778
Genetic Miner	0.998	0.737
HeuristicsMiner	0.973	0.809
ILP Miner	**1.000**	0.786

Appendix B

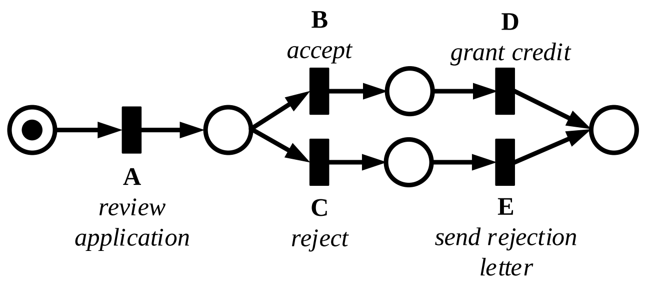
DESCRIPTION OF THE BASELINE PREDICTORS USED IN EXPERIMENT 2

We applied N-gram models to business process event data in experiment 2. N-gram models, popular techniques for language modeling, distribute the event sequences of business processes by means of several conditional probability tables. For each sequence of up to n-1 events, a probability table is maintained that specifies the distribution over the next event. The distribution is modeled formally as follows:

$$P\left(X_0^{(c)}, \dots, X_{T_c}^{(c)}\right) = \prod_{i=0}^{T_c} P\left(X_i^{(c)} \middle| X_{i-1}^{(c)}, \dots, X_0^{(c)}\right) \approx \prod_{i=0}^{T_c} P\left(X_i^{(c)} \middle| X_{i-1}^{(c)}, \dots, X_{i-n+1}^{(c)}\right) .$$

The conditional probability tables $P\left(X_i^{(c)} \middle| X_{i-1}^{(c)}, \dots, X_{i-n+1}^{(c)}\right)$ can be estimated by processing the event log to search for substrings that match the values of the $X_{i-1}^{(c)}, \dots, X_{i-n+1}^{(c)}$ variables and counting how often each event type follows the sequence. The counts allow probabilities to be estimated as relative frequencies.

As an example, Fig. B1 shows the same business process and event log that Fig. 1 shows, but it also contains conditional probability tables for a 3-gram estimated from the five process instances in the event log. For instance, these tables predict that, after an event sequence *AB*, an event of type *D* will follow with probability 1.0, and after an event sequence *BD*, an event of type *kill* will follow with probability 1.0. Event *kill* is the artificial event that indicates process termination.



Event log

Process instance	Event sequence
C1	ABD
C2	ACE
C3	ABD
C4	ACE
C5	ACE

Sequence = (--)

Event type	Probability
A	1.0
B	0.0
C	0.0
D	0.0
E	0.0
kill	0.0

Sequence = (-A)

Event type	Probability
A	0.0
B	0.4
C	0.6
D	0.0
E	0.0
kill	0.0

Sequence = (AB)

Event type	Probability
A	0.0
B	0.0
C	0.0
D	1.0
E	0.0
kill	0.0

Sequence = (BD)

Event type	Probability
A	0.0
B	0.0
C	0.0
D	0.0
E	0.0
kill	1.0

Sequence = (AC)

Event type	Probability
A	0.0
B	0.0
C	0.0
D	0.0
E	1.0
kill	0.0

Sequence = (CE)

Event type	Probability
A	0.0
B	0.0
C	0.0
D	0.0
E	0.0
kill	1.0

Fig. B1. Estimating a 3-gram for an exemplary event log

We show only a subset of all possible conditional probability tables. For instance, there is no table for event sequence *AD* because the tables are constructed from the relative frequencies with which certain types of events follow on the event sequence in the event log, and there is no occurrence of *AD* in the event log.

We maintain tables not only for event sequences of length $n - 1$, but also for shorter event sequences. In the example in Fig. B1, we maintain a table for the empty sequence (--) and for the sequence that contains only an event of type *A* (-A). The empty sequence is needed in order to model the probabilities of seeing a given type of event at the beginning of the process, while the sequence of only an event of type *A* (-A) is needed since *A* was observed in the event log with no event before it. In two out of five processes' instances, an event of type *B* follows after seeing only *A*. Three out of five instances proceed with a *C*, so the corresponding probabilities in the table are 0.4 and 0.6.

The *History* predictor, which we used in experiment 2 in addition to the n-gram, can be interpreted as a special type of n-gram. Since the *History* predictor is not limited in terms of the length of the event sequence it considers, it is an n-gram of unbounded length. Given a particular event log in which the longest process instance is of length $T_{max} = \max_{c} T_c$, the *History* predictor is a T_{max}-gram.