Predicting Unseen Process Behavior Based on Context Information from Compliance Constraints

Qian Chen[1(✉)], Karolin Winter[2], and Stefanie Rinderle-Ma[1]

[1] Technical University of Munich, TUM School of Computation, Information and Technology, Garching, Germany
{qian.chen,stefanie.rinderle-ma}@tum.de

[2] Eindhoven University of Technology, Department of Industrial Engineering and Innovation Sciences, Eindhoven, The Netherlands
k.m.winter@tue.nl

Abstract. Predictive process monitoring (PPM) offers multiple benefits for enterprises, e.g., the early planning of resources. The success of PPM-based actions depends on the prediction quality and the explainability of the prediction results. Both, prediction quality and explainability, can be influenced by unseen behavior, i.e., events that have not been observed in the training data so far. Unseen behavior can be caused by, for example, concept drift. Existing approaches are concerned with strategies on how to update the prediction model if unseen behavior occurs. What has not been investigated so far, is the question how unseen behavior itself can be predicted, comparable to approaches from machine learning such as zero-shot learning. Zero-shot learning predicts new classes in case of unavailable training data by exploiting context information. This work follows this idea and proposes an approach to predict unseen process behavior, i.e., unseen event labels, based on process event streams by exploiting compliance constraints as context information. This is reasonable as compliance constraints change frequently and are often the cause for concept drift. The approach employs state transition systems as prediction models in order to explain the effects of predicting unseen behavior. The approach also provides update strategies as the event stream evolves. All algorithms are prototypically implemented and tested on an artificial as well as real-world data set.

Keywords: Predictive Process Monitoring · Unseen Behavior · Context Information · Compliance Constraints

1 Introduction

Predictive Process Monitoring (PPM) aims at predicting relevant target values based on an event log such as the next event label to occur [16] as well as the remaining time [24] or the outcome [23] of process instances. In general, PPM provides great prospects for decision support in almost any application domain.

Especially in advanced application domains such as manufacturing, logistics, and healthcare, PPM is confronted with the problem of *unseen behavior*, i.e., behavior that has not been observed in the event log so far and hence is not available for the training phase of PPM. Unseen behavior might be caused by concept drift as well as by incomplete or infrequent process executions [13,14,20]. Unseen behavior in PPM results in two challenges: i) when and how to update the prediction model at the presence of unseen behavior [4,14,18,20], and ii) how to predict unseen behavior. ii) can be compared to zero-shot learning in Machine Learning (cf. [27]) where classes can be predicted also in case of unavailable training data by exploiting context information.

In this work, we follow up on the idea of zero-shot learning for unseen behavior in PPM, i.e., we investigate how context data can be exploited to enable the prediction of unseen behavior, i.e., unseen next event labels. The exploitation of context information for improving the prediction quality and the explainability of the prediction results has been investigated for different sources of context data such as text [25], expert feedback [5], and sensor data [21]. However, none of the existing approaches has exploited context data for predicting unseen behavior. The idea of exploiting context data is somehow similar to the idea of using a-priori knowledge to be *"leveraged for improving the predictive power of a predictive process monitoring technique"* [8]. The difference is that a-priori knowledge can be considered context data, but not necessarily vice versa. More precisely, context data is not necessarily available in an a-priori way, but emerges and changes during runtime.

In this work, we opt for exploiting one source of context data as a starting point, i.e., compliance constraints. The reason is that compliance constraints are prevalent in almost any application domain and are often themselves the cause for concept drift which, in turn, constitutes a source for unseen behavior. Compliance constraints usually emerge from regulatory documents and can be subject to frequent changes [10], i.e., have to be constantly monitored themselves. Consider a process for the transportation of delicate foods which needs to obey to multiple regulations. If a change in one of those regulations happens, new compliance constraints can come into effect, e.g., *If the temperature in the container exceeds 10° during transportation, these goods must be destroyed.* Since this is a newly imposed constraint, the associated event(s) corresponding to the destruction of goods have not been observed in the underlying event log and are hence not included in the training phase, causing *unseen behavior* in the test phase. The idea is to integrate information from the compliance constraint into the prediction model. This way we do not have to wait until we observe the new process behaviour caused by this constraint, e.g., the event(s) reflecting an activity related to goods destruction. Compliance constraints can therefore be seen as "promising" context to enable the prediction of unseen behavior, resulting in the overarching research question of this work:

How to exploit context information on compliance constraints to predict unseen process behavior, i.e., unseen next event labels?

The input of the approach comprises a process event log and a set of compliance constraints that apply to the underlying process. Furthermore, we consider that the process event log can be incomplete in terms of observations, i.e., not every event label that is conceivable based on the set of constraints has been observed so far. This corresponds to epistemic uncertainty as typically referenced in the machine learning community, cf., e.g., [11,26]. By incorporating knowledge on compliance constraints we do not have to wait until we observe the behavior enforced by them and update the prediction model when we have observed it. We can already anticipate unseen behavior based on, e.g., data attributes, e.g., in the food transportation example the temperature exceeding 10 degrees. After the unseen behavior has been observed our approach uses appropriate update strategies to further enhance the prediction quality.

The presented approach comprises an offline component that augments a state transition system with contextual information from compliance constraints. The augmented transition system is then used for prediction with an event stream in the online component. The approach is evaluated based on synthetic and real-world event logs without and with existing update strategies. Moreover, the augmented transition system is compared to deep learning methods with either a single $LSTM$ layer [14] or a Process Transformer [3].

The remainder of this paper is structured as follows. Section 2 outlines the problem statement, Sect. 3 presents the next event label prediction approach using constraint context information. The approach is evaluated in Sect. 4, related work is discussed in Sect. 5 before the paper concludes in Sect. 6.

2 Problem Statement and Preliminaries

Next event label prediction takes an event log (training data) and an event stream (test data) as input and predicts based on the event log the next event label for the ongoing process instance observed in the event stream. The event log contains events which origin from different cases, each case represented by a trace. An event can have multiple attributes, e.g., an event label, a timestamp, a life cycle transition, data values, or resources that were involved during the execution. In the following running example, we abstract from this representation by just depicting a trace as sequence of event labels and their number of occurrences.

Example: *Assume a scenario with event log* $L = [< A, B, E >^{100}, < A, C >^{100}]$ *as training data where the number in the superscript denotes the frequency of the trace occurrence, and event stream* S *as test data, i.e.,* $S =< A_1, A_2, B_1, C_2 >$ *where the subscript reflects the case id. For S, existing prediction models would result in predicting* E *for case 1 and no prediction for case 2.*

In this paper we aim to predict the next event label for an evolving event stream based on an event log and a set of compliance constraints as additional context information. A compliance constraint c is defined as a triplet (p, s, r) consisting of a non-empty predecessor event p, a possibly empty successor event s and r specifying the relation between p and s. This definition is deliberately kept independent from any formalism such as linear temporal logic, in order to show the general applicability of the approach.

Example (ctd.): *Assume that in the scenario described above, the following two constraints are imposed on the process execution due to, e.g., newly arising or updated regulations:*

c_1 : "D directly follows C"
c_2 : "Y eventually follows B"

Constraints c_1 *and* c_2 *can be formalized as* $c_1 = (\{C\}, \{D\}, \{directly\ follows\})$ *and* $c_2 = (\{B\}, \{Y\}, \{eventually\ follows\})$. *The directly follows relation means that whenever C occurs, D must occur next without other events in between. The eventually follows relation implies that whenever B occurs, Y must occur afterwards* [12].

For the running example, existing next event label prediction without considering the constraint information, cannot predict events D and Y since they have not been observed in L. Only approaches considering updates can incorporate this information if at some time it is observed in the stream. By integrating compliance constraints c_1 and c_2 into the prediction model, we gain knowledge about those additional events in advance allowing for their prediction.

Example (ctd.): *Including the additional knowledge contributed by constraints* c_1 *and* c_2 *into next event label prediction, we envision the prediction of* E *or* Y *for case 1 and the prediction of* D *for case 2.*

The presented approach will be capable of predicting unseen events that stem from constraints without requiring updating the prediction model. However, as soon as unseen behavior emerging from sources other than constraints is observed in the stream, this information is included via updates, as well.

Example (ctd.): *Assume that event stream S evolves in the following steps. Then existing approaches and the envisioned approach including constraints yield the prediction results summarized in Table 1 without updating the prediction models. We can see that existing approaches do not yield any predictions if the trace length exceeds the longest observed trace which is the case for any of the streams in Table 1. This might give the prediction with constraint information an edge, at least until prediction models are updated. We will investigate the "sweet spot" between the effort and gain of including constraint information into the prediction, in connection with update strategies, in Sect. 4.*

Table 1. Prediction results with evolving event stream; –: no prediction

Event stream	**Existing approaches**	**Including constraints**
$S =< A_1, A_2, B_1, C_2, E_1 >$	case 1: – ; case 2: –	case 1: Y; case 2: D
$S =< A_1, A_2, B_1, C_2, E_1, D_2 >$	case 1: – ; case 2: –	case 1: Y; case 2: –
$S =< A_1, A_2, B_1, C_2, E_1, D_2, Y_1 >$	case 1: – ; case 2: –	case 1: – ; case 2: –

In order to consider that unseen behavior is predicted, we aim at predicting not only the next event labels, but also how certain we are for the upcoming event label.

In the paper, we assume that we are in a violation free setting, meaning that the event log L never violated the set of compliance constraints C we consider with unseen behavior, but violations of other compliance constraints are in principle possible. In the case of the event stream, we only include cases that do not contain violations of compliance constraints when updating the prediction model. Moreover, we consider that we have observed the predecessor of a constraint. That in turn means that a chaining of constraints, i.e., "B follows A" and "C follows B", may not occur as we would lack the predecessor event B. Furthermore, the assumption is made that there are no overlaps in control-flow constraints, i.e., "B directly/eventually follows A" and at the same time "X directly/eventually follows A" cannot occur without further knowledge. Further knowledge means in this case that certain conditions on data attributes need to hold, e.g., "If the credit amount is greater than 10.000€, "perform detailed check" must happen after "request received", otherwise "perform normal check" must happen after "request received". In this work, we only consider control-flow constraints without any further information (e.g., data attributes), but it will be taken into account in future work.

3 Next Event Label Prediction Approach

In order to address the problem as outlined in Sect. 2, we follow the basic idea of predictive process monitoring approaches by initially training a prediction model in an offline component and carrying out the prediction and update of the prediction model in an online component. For the offline component the input consists of an event log L and a set of compliance constraints C. The output is a prediction model trained based on information from L combined with the external knowledge based on C. The online component takes an event stream S and the set of compliance constraints C as input and delivers the prediction of the next event label with corresponding probability. In order to foster explainability the user is informed whether the prediction was made solely based on L or based on which compliance constraint in C. Moreover, several update strategies allowing to cope with concept drifts induced by changes in the underlying process models are integrated into the online component.

As prediction model we opt for transition systems as introduced in [1,19]. Both papers focus on remaining time prediction and the latter incorporates regression models based on data attributes and allows for activity sequence prediction as well. Transition systems are selected because we are facing the challenges of i) frequent changes causing unseen behavior and ii) explainability. Considering i) compared to, e.g., deep-learning models, transition systems with appropriate abstractions can be constructed at low computational costs allowing for incorporating changes as soon as they occur without waiting hours or days for retraining the prediction model. For ii) transition systems are a white box model and allow for different explainability options, i.e., we can easily convey prediction results to users and, e.g., distinguish between predictions that are made solely based on the given event log or predictions that were made based on a particular constraint in combination with the probabilities attached to the prediction result.

3.1 Creating the Prediction Model – Offline Component

For the offline component, at first, an annotated transition system based on event log L is constructed analogously to [1,19] and later on augmented with constraint information. A *transition system* TS constructed based on event log L consists of a set of states S, a set of event labels E and a set of transitions T. A transition t is a triplet (s_1, e, s_2) determining how one state s_1 is conveyed into another state s_2 via an event e. Events and states can be represented through different representation functions. An example for an event representation function is the function that maps an event onto its event label. An example for a state representation function is the function that maps a partial trace onto its sequence of event labels.

Figure 1 depicts the transition system TS for the running example as introduced in Sect. 2 where the parts of TS generated based on the event log L are depicted in black. The partial traces created from the traces in L are $< A >, < A, B >, < A, B, E >, < A, C >$. As illustrated in Fig. 1 in the case of i) representing states as sequences of partial traces, i.e., mapping each partial trace onto the event labels while taking the order into account, the set of states S consists of exactly those traces. In the case of ii) representation as last event, i.e., mapping each partial trace onto the label of the last event, the states are given as A, B, C and E. Note that artificial empty start states <> and ().

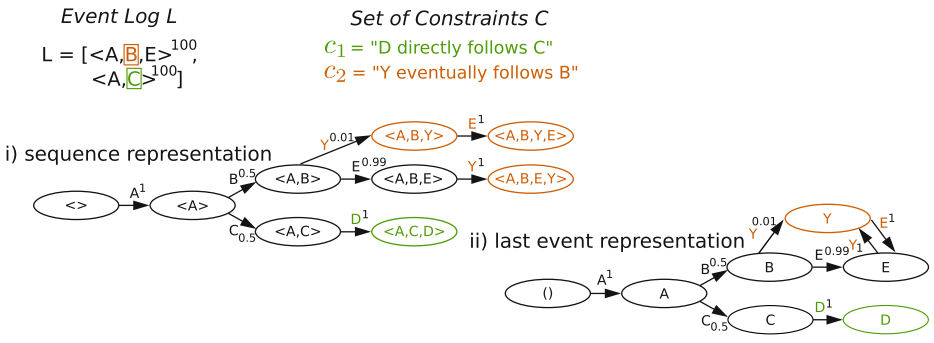

Fig. 1. Augmented Annotated Transition Systems for the Running Example

States can be further annotated with all possible next states, and each of them is associated with a probability from the prior state to them. The probability of each outgoing state is calculated using a measurement function. An *annotated* *transition system* ATS is then a transition system TS together with an event and state representation as well as a measurement function [1]. As measurement function for the TS constructed based on event log L we consider the function calculating the probability based on relative occurrences of the transitions within the log file. For instance, the probability from state $< A >$ to state $< A, B >$ in the running example in Fig. 1 is the quotient of the number of visits from $< A >$ to $< A, B >$ and the number of total visits from state $< A >$ to its all possible next states. So in this case, the probability from state $< A >$ to state $< A, B >$ is 0.5, as $< A, B >$ has been observed 100 times in the event log and $< A, C >$ occurs in 100 cases.

For further details on transition systems, state and event representations, as well as measurement functions, we refer the reader to [1,19].

In order to incorporate constraint information, the basic annotated transition system ATS calculated based on event log L is augmented based on information from the set of compliance constraints C. The result is denoted as *Augmented Annotated Transition System* $AATS$ and is constructed based on Algorithm 1. The algorithm works independently from the chosen state and event representations.

Algorithm 1. Annotated Transition System Augmentation Algorithm

Input: ATS = annotated transition system based on an event log, C = set of constraints, one constraint c corresponds to a triple (predecessor, successor, relation)

Output: $AATS$ = augmented annotated transition system

```
1: for each eventually follows constraint c in $C$ do
2:     if c.predecessor and c.successor have been observed in $ATS$ then c was already observed,
       do no augmentation for constraint
3:     end if
4:     if only c.predecessor has been observed in $ATS$ then add c.successor to every state which
       contains c.predecessor; extend beyond constraints if the state with further events
5:     end if
6:     if c.predecessor has not been observed but c.successor has been observed in $ATS$ then there
       must be an error or violation in the log for constraint
7:     end if
8: end for
9: for each directly follows constraint c in $C$ do
10:    if c.predecessor and c.successor have been observed in $ATS$ then c was already observed,
       do no augmentation for constraint
11:    end if
12:    if only c.predecessor has been observed in $ATS$ then add c.successor to every state which
       ends with c.predecessor; extend beyond constraints if the state with further events
13:    end if
14:    if c.predecessor has not been observed but c.successor has been observed in $ATS$ then there
       must be an error or violation in the log for constraint
15:    end if
16: end for
```

In Algorithm 1, we take as input the basic annotated transition system ATS and a set of compliance constraints C. In order to avoid violation after augmentation, the basic ATS is augmented with eventually follows constraints first (see line 1). For example, if an additional constraint c_3 : "F directly follows E" is introduced for the running example in Sect. 2, then $< A, B, E >$ is extended with F. However, the eventually follows constraint c_2 : “Y eventually follows B” will generate a violated trace $< A, B, E, Y, F >$ on top of $< A, B, E, F >$. Next, the basic ATS is augmented with the eventually follows constraint based on three scenarios. For scenario 1 (line 2 and line 3), if both events are observed in ATS then no augmentation is conducted. The main scenario this paper focuses on is described from line 4 to line 5, in which the predecessor has been observed in ATS while the successor is unseen at the moment. For eventually follows augmentation, the successor should be added to the end of each state in which the predecessor is included. We represent this augmented state as a constraint state as it is directly constructed from a constraint. When an additional state is added to the basic transition system, the count of the corresponding transition from the initial state to the additional state is increased by 1. If the initial state which contains the predecessor of the constraint has further events, then those events need to be added after the constraint state. Line 6 and line 7 indicate a violation scenario if the successor is observed before the predecessor in the log. The augmentation process for directly follows constraints is similar to the eventually follows one except for the second scenario from line 12 to line 13. In this case, only states which end up with the predecessor of the constraint are augmented with the successor of the constraint. We again extend the constraint state with following events if the initial state has further events.

After constraint augmentation as described in Algorithm 1, additional states and transitions are appended on top of the basic ATS. This leads to an update of annotations constructed in ATS before. Here we use the same measurement function as for ATS to calculate the probability of relative occurrences for states from the basic transition system and additional states from the augmented transition system. For example, the probability from state $< A, C >$ to state $< A, C, D >$ in the running example as depicted in Fig. 1 is 1 since there is only one transition from $< A, C >$ to $< A, C, D >$ which is augmented from the constraint.

In Fig. 1, the transitions and states that are inserted based on constraint information are depicted in green and orange depending on which constraint was used to create the state.

3.2 Next Event Label Prediction – Online Component

The online component serves two purposes, i.e., predicting the next event label and constantly updating and improving the existing prediction model based on event stream S. In the phase of online prediction, the $AATS$ constructed from the offline phase is applied to incoming traces from the event stream to predict the next event labels with their associated probabilities. The next event label with the highest probability is selected from the set of possible events. Note here, if the next event label is predicted based on information from compliance constraints, then the corresponding constraint information is provided, as well. This enables our approach to explain whether the prediction is based on the event log or the set of constraints. There could also be multiple next event labels possible for a partial trace with the same probability. In this case, the $AATS$ will predict all possible next event labels with again the probabilities and constraints if available.

As we use occurrence frequencies as measurement function for calculating probabilities from one state to another, consequently, updating the prediction model constantly as event stream evolves is needed. Here, the updating mechanism can be either triggered based on i) the event stream S or by ii) changes in the constraint set C. For i), the following scenarios are possible: a) only labels that were already observed in the log and constraint set are in the stream, i.e., no new/additional states and transitions need to be created and only the probabilities need to be recalculated; b) new labels are observed resulting in the need for a full retraining of the model, i.e., both ATS creation and constraint augmentation need to be conducted again. That means the initial ATS is updated based on new states and transitions, then the constraint augmentation according to Algorithm 1 is applied to the updated ATS. For ii) we need to recalculate only the augmentation part based on the initial ATS that is constructed from the event log L, i.e., perform Algorithm 1 with the newly updated set of compliance constraints. Combinations of i) and ii) are also conceivable.

4 Evaluation

The approach is prototypically implemented and available at https://www.cs.cit.tum.de/bpm/software/. To evaluate the approach a synthetic event log[1] generated using the Cloud Process Execution Engine[2] (CPEE) [15] and a well-established real-life event log, the Helpdesk event log[3], are used. As to the best of our knowledge, this approach is the first to aim at predicting unseen behavior resulting from constraints, we consider the following four comparisons, i) ATS vs. AATS without updates, ii) ATS vs. AATS with updates, iii) AATS vs. a deep-learning model without updates, and iv) AATS vs. a deep-learning model with updates. The first comparison aims at illustrating the advantage of taking constraint information into account. The second comparison shall demonstrate that the updating strategy for the AATS was chosen correctly, and the third comparison shall provide insights on how well a basic technique like the AATS performs against sophisticated prediction models. The last comparison seeks to highlight the benefits of our proposed approach against existing approaches in dealing with unseen behavior, i.e., updating prediction models when considering unseen event labels or new sequences. Given that the prediction of the next event label is a multi-class classification problem, we choose *accuracy*, *precision*, *recall*, and the *f1-score* as metrics. The *accuracy* measures the proportion of correct predictions to the total number of predictions made. The *f1-score* is the harmonic mean of *precision* (or positive predictive value) and *recall* (or sensitivity), where *precision* determines the exactness of the model, and *recall* measures the model's completeness [3].

[1] https://www.cs.cit.tum.de/bpm/data/.
[2] https://cpee.org, last access: 2023-03-21.
[3] https://data.4tu.nl/articles/_/12675977/1.

4.1 Data Sets

All data sets consist of an event log L containing only the predecessor events of the compliance constraints, i.e., the successor events are still unseen, the set of compliance constraints C and a test set representing the event stream S. The latter contains full information, i.e., also the unseen event labels for the constraints in set C.

Synthetic Data. In order to meet our violation-free assumption (cf. Sect. 2), we generate a process model with decision and parallel gateways as shown in Fig. 2. The training set contains 200 cases among which 4 different activities A, B, C, D are executed. Then one directly follows constraint: $c_1 = (\{D\}, \{E\}, \{directly\ follows\})$ and two eventually follows constraints: $c_2 = (\{B\}, \{X\}, \{eventually\ follows\})$, $c_3 = (\{C\}, \{Y\}, \{eventually\ follows\})$ are imposed on the process model. This is reasonable since constraints might change over time and these changes should be integrated into the process model as soon as possible. As our paper aims to address partially seen cases, thus, all predecessors of constraints (i.e., B, C, D) are already observed in the training set, while the successors of them (denoted in green and orange boxes in Fig. 2) (i.e., E, X, Y) remain unseen in the historic event log. However, those unseen activities from the set of constraints could occur in stream data as time goes by. Therefore, the test set with 200 cases is generated based on the whole process model with unseen activities E, X, Y additionally.

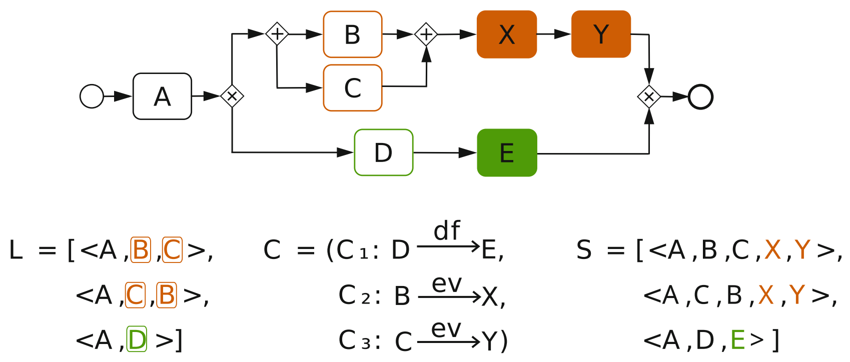

Fig. 2. Process model for synthetic data set

Helpdesk Dataset. The Helpdesk event log reflects a process to resolve tickets that are raised by users. The data set contains 21348 events with 14 distinct activities distributed among 4580 cases. The average case length of all process instances is 4.66, while the maximal value is up to 15. In addition to the event log, a set of constraints is needed. As this data set does not include any information about compliance constraints, we artificially create a directly follows constraint

$c_1 = (\{Resolve\ ticket\}, \{Closed\}, \{directly\ follows\})$ based on the analysis of the log file. To align with our assumption, all events with event label "Closed" are removed from the log file to keep the successor of the constraint unseen. The training set consists of all events without "Closed", while the test set is the same as the original log file with "Closed".

4.2 ATS vs. AATS Without Updates

To demonstrate the benefits of leveraging additional information from compliance constraints, we compare experimental results between traditional ATS and our envisioned approach without updates. The test results based on the Synthetic and Helpdesk data set are provided in Table 2.

In general, our approach performs better than ATS in terms of all metrics in both data sets. In particular, a significant improvement can be observed for the synthetic data set. This is because the prediction model for ATS is solely trained based on the historic event log in which unseen activities like E, X, Y have never been observed. Thus, the prediction model is unable to deal with these unknown situations in online prediction without updating accordingly. However, our approach has considered unseen process behavior originated from the constraint set in the training phase already. This enables $AATS$ to cope with unseen cases even without updating strategies. For the Helpdesk data set, the overall predictive quality for both approaches is not as expected. Considering that the compliance constraint $c_1 = (\{Resolve\ ticket\}, \{Closed\}, \{directly\ follows\})$ is artificially created to test the presented approach. There are many violated cases against constraint c_1 in the test set. In this case, without updating the model during predictions hinders the possibility for both two models to capture those "violated" cases caused by the introduction of the artificially generated constraint. The slight improvement for $AATS$ compared to ATS in this data set can be attributed to the limited information (i.e., only one directly follows constraint with "Closed" as an unseen event label) we can obtain from constraints.

Table 2. Results ATS vs. AATS (no updates)

Data set	Approach	Accuracy	F1-score	Precision	Recall
Synthetic	ATS	0.250	0.249	0.313	0.250
	AATS	**0.815**	**0.810**	**0.897**	**0.815**
Helpdesk	ATS	0.432	**0.345**	0.363	0.432
	AATS	**0.450**	0.344	**0.567**	**0.450**

4.3 ATS vs. AATS with Updates

We assume that both approaches are capable of updating the prediction models if unseen behavior is observed in the event stream (cf. results in Table 3).

Compared to Sect. 4.2, if ATS is embedded with updating strategies during prediction, it outperforms our approach especially for the Synthetic data set. This is explainable since our approach strives to cover all possible variants of enriching the event log with additional information from constraints. Given that we use the probability of relative occurrences as the measurement function, the increase in the total amount of possible transitions for one state to another results in a decrease in the probability of the appropriate transition. Moreover, $AATS$ does not undergo updates as ATS did because the test set lacks unseen event labels for $AATS$ to update after the augmentation of constraints. Instead, it only updates the probability for each transition of the evolving stream. For the Helpdesk data set, the performances of both approaches with updating strategies are significantly improved compared to the results in Table 2. Since the test set contains lots of violated cases and unseen behavior (i.e., "Closed") for ATS, it is hard to tell the source of it's improvement. By contrast, the better performance of $AATS$ is owing to updating on those violations as we have incorporated the unseen event label into model training.

Table 3. Results ATS vs. AATS (with updates)

Data set	Approach	Accuracy	F1-score	Precision	Recall
Synthetic	ATS	**0.865**	**0.867**	**0.934**	**0.865**
	AATS	0.816	0.812	0.898	0.816
Helpdesk	ATS	0.705	0.670	**0.748**	0.705
	AATS	0.705	0.670	0.745	0.705

4.4 AATS vs. Deep-Learning Model Without Updates

Approaches using transition systems and deep-learning-based models without updating mechanisms are compared to provide insights into model selections in predictive process monitoring. Here we adopt Process Transformer proposed by [3] as the prediction model. The reason is that the Process Transformer uses the self-attention mechanism to reason over long-range dependencies and is able to process inputs in parallel [3]. We use the default hyperparameter configurations and enrich the training set with unseen event labels from the set of constraints. Then the prediction model (i.e., Process Transformer) is trained based on the enriched data set with all possible variations regarding constraints we considered. The test results are provided in Table 4.

For Synthetic data set, $AATS$ performs better than the deep-learning-based approach. However, the latter outperforms $AATS$ considerably on Helpdesk data set. This could be attributed to the type of constraints we considered. Two eventually follows constraints imposed on the Synthetic data set result in an enriched training set with plenty of variations, e.g., $< A, C, Y, B, X >$, while most of the augmented samples are incorrect cases that will not occur in the event stream (i.e., the test set only contains traces $< A, B, C, X, Y >$ and $< A, C, B, X, Y >$). Note that $AATS$ covers all these variations as well, but with the count of 1 for each variation regardless of the data size, whereas the deep-learning-based approach does augmentations for each individual trace if the partially seen criteria is satisfied. Thus, there are lots of incorrectly enriched cases in the training set. Conversely, the Helpdesk data set only considers one directly follows constraint, thus, no misleading samples are generated based on this constraint.

Table 4. Results DL vs. AATS (no updates)

Data set	Approach	Accuracy	F1-score	Precision	Recall
Synthetic	DL	0.732	0.714	0.729	0.732
	AATS	**0.815**	**0.810**	**0.897**	**0.815**
Helpdesk	DL	**0.844**	**0.812**	**0.796**	**0.844**
	AATS	0.450	0.344	0.567	0.450

4.5 AATS vs. Deep-Learning Model with Updates

To shed light on the effectiveness of the proposed approach, we conduct experiments comparing $AATS$ and existing approaches in dealing with unseen process behavior, i.e., updating deep-learning models on demand. We choose the most comparable deep-learning-based approach proposed in [14] with a single $LSTM$ layer. Following the experimental settings as stated in [14], we expand the Synthetic data set to include 1000 cases, of which 10% are allocated as a training set, and the remaining 90% are assigned to the test set. This means 100 cases are selected from the training set as we described in Sect. 4.1, and 900 cases are generated based on the process model with compliance constraints. The same training-test splitting is applied to Helpdesk data set as well. Moreover, the timestamp in the test set of the Synthetic data needs to be adapted to simulate an online setting as mentioned in [14], i.e., daily prediction. Experimental results are summarized in Table 5. Different update strategies provided in [14] are denoted as S_0 (do not update), S_1 (update on new activities), S_2 (update on new sequences) and S_3 (update every day). Time spent for each update strategy is represented as hours:minutes:seconds.

For both two data sets, $AATS$ underperforms the deep-learning approach with updating strategies like S_3 (update every day). By contrast, in terms of computational time, deep-learning approaches with updating mechanisms can take hours and even days to update prediction models when coping with relatively large data sets (e.g., Helpdesk data set).

Findings: To sum up, $AATS$ performs better than ATS if no updates are conducted during online prediction. When considering updates, ATS can learn from the proper incoming traces under the violation-free assumption directly. This leads to a better predictive performance than $AATS$ as it cannot handle

Table 5. Results DL vs. AATS (with updates)

Data set	Approach	Accuracy	F1-score	Precision	Recall	Time
Synthetic	S_0	0.403	0.401	0.498	0.403	00:00:01
	S_1	0.688	0.638	0.697	0.688	00:00:02
	S_2	0.403	0.401	0.498	0.403	00:00:01
	S_3	**0.829**	**0.828**	**0.899**	**0.829**	00:00:47
	AATS	0.813	0.809	0.896	0.813	00:00:02
Helpdesk	S_0	0.275	0.274	0.460	0.275	00:00:20
	S_1	0.657	0.652	0.659	0.657	00:04:46
	S_2	**0.778**	**0.756**	0.741	**0.778**	09:36:10
	S_3	0.776	0.755	**0.742**	0.776	28:42:32
	AATS	0.703	0.671	0.735	0.703	00:06:53

the reduction of the augmented transition system appropriately. In light of prediction model selections, deep-learning models demonstrate superior prediction quality compared to transition systems if eventually follows constraints are augmented properly. Nevertheless, it is computationally expensive for deep-learning-based approaches when considering update mechanisms. We discuss options for improvement in Sect. 6.

5 Related Work

This paper addresses research questions at the intersection of the i) exploitation of (external) context information and ii) handling of unseen behavior in predictive process monitoring. For i) [2,6,17] present a taxonomy for process context information, differentiating its origin into internal/intrinsic (within the log) and external (outside the log) context data. [25] distinguish between structured and unstructured context information, stating that *"the majority of the works on context-aware predictive business process monitoring relies on structured context data created by the information system itself like process performance metrics."*. [7] discover context-aware prediction models based on internal context data. [22] include internal, textual data as unstructured context data to predict the outcome of a running case. By contrast, [28] provide an approach for remaining time prediction by considering sentiments triggered by news (external context data). [5] propose an approach to identify context information for performance indicator prediction based on expert and domain knowledge. [21] exploit (external) sensor streams for predicting and explaining concept drift. The only approach exploiting information on compliance constraints in predictive process monitoring is [8] by ranking predictions based on their compliance with imposed constraints. However, the predictions are still based on observed behavior, i.e., they do not consider unseen behavior yet. Only few approaches envision strategies to cope with unseen behavior (ii). [4,20] elaborate update strategies for prediction models. Incremental learning for predicting the outcome of a process instance has been applied by [9,13]. [18] use incremental learning for predicting the next activity. In [14], we conclude that an "*"update on demand" strategy yields the best results in terms of balancing prediction quality and performance."*. These approaches do not predict unseen behavior themselves as advocated in this work, but can be combined in order to deal with the evolving event stream.

6 Conclusion

This paper provides an approach to exploit contextual information from a set of compliance constraints C for predicting unseen behavior in the training data, i.e., in the event logs. For this, we augment state transition systems with the constraint information and use them as prediction model. A first insight is that for violation-free logs, i.e., logs that respect the compliance constraints we considered, the prediction quality is higher for the prediction with context information. When updating the models without context information with unseen behavior, the prediction quality converges to similar values. An interesting insight is that we drop the assumption of violation-free event logs, the prediction quality of the augmented prediction model might not exceed the quality of the prediction models without augmentation. Conversely, an unexpectedly low prediction quality of the approach with compliance constraint information can be interpreted as an indicator that the underlying logs contain violations, contributed by, for example, interleaving. Interleaving occurs if the directly follows semantic of a constraint is "broken" by executing an activity from another parallel branch. In these situations, the approach can also be used as a mechanism to detect possible compliance violations.

Discussion: We can understand the assumptions made in Sect. 2 as current limitations of the approach. At first, we only consider directly follows and eventually follows semantics of the constraints. However, compliance constraints can be more expressive, e.g., restricting the resource perspective via a separation of duty constraint. Second, we assume violation-free logs which might not be the case for real-world logs. In this case, prediction quality lower than expected can indicate compliance violations. Another limitation refers to the measurement function that is used for calculating the probabilities in the augmented state transition system. In this work, it uses occurrence frequencies. In future work, additional information can be exploited such as remaining time or data values from the event log to weigh probabilities differently. Third, we do not explicitly consider changes in constraint set C, though the presented approach is able to deal with such changes if they are under the partially seen scenario by re-running the augmentation algorithm.

Future work aims at exploiting compliance constraints referring to process perspectives, e.g., data attributes, beyond control flow when predicting next event labels. We will also test different abstraction functions, cf. [1]. Moreover, prediction goals such as remaining time will be added to the approach. Future work will also feature experiments with different update strategies in case of constraint set updates.

Acknowledgements. This work is funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – project number 277991500.

References

1. van der Aalst, W.M.P., Schonenberg, M.H., Song, M.: Time prediction based on process mining. Inf. Syst. **36**(2), 450–475 (2011). https://doi.org/10.1016/j.is.2010.09.001
2. Brunk, J., Stierle, M., Papke, L., Revoredo, K., Matzner, M., Becker, J.: Cause vs. effect in context-sensitive prediction of business process instances. Inf. Syst. **95**, 101635 (2021). https://doi.org/10.1016/j.is.2020.101635
3. Bukhsh, Z.A., Saeed, A., Dijkman, R.M.: Processtransformer: predictive business process monitoring with transformer network. CoRR abs/2104.00721 (2021). https://arxiv.org/abs/2104.00721
4. Chamorro, A.E.M., Nepomuceno-Chamorro, I.A., Resinas, M., Ruiz-Cortés, A.: Updating prediction models for predictive process monitoring. In: Advanced Information Systems Engineering, pp. 304–318 (2022). https://doi.org/10.1007/978-3-031-07472-1_18
5. Chamorro, A.E.M., Revoredo, K., Resinas, M., del-Río-Ortega, A., Santoro, F.M., Ruiz-Cortés, A.: Context-aware process performance indicator prediction. IEEE Access **8**, 222050–222063 (2020). https://doi.org/10.1109/ACCESS.2020.3044670
6. Ehrendorfer, M., Mangler, J., Rinderle-Ma, S.: Assessing the impact of context data on process outcomes during runtime. In: Hacid, H., Kao, O., Mecella, M., Moha, N., Paik, H. (eds.) ICSOC 2021. LNCS, vol. 13121, pp. 3–18. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-91431-8_1
7. Folino, F., Guarascio, M., Pontieri, L.: Discovering context-aware models for predicting business process performances. In: Meersman, R., et al. (eds.) OTM 2012. LNCS, vol. 7565, pp. 287–304. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33606-5_18
8. Di Francescomarino, C., Ghidini, C., Maggi, F.M., Petrucci, G., Yeshchenko, A.: An eye into the future: leveraging a-priori knowledge in predictive business process monitoring. In: Carmona, J., Engels, G., Kumar, A. (eds.) BPM 2017. LNCS, vol. 10445, pp. 252–268. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-65000-5_15
9. Francescomarino, C.D., Ghidini, C., Maggi, F.M., Rizzi, W., Persia, C.D.: Incremental predictive process monitoring: How to deal with the variability of real environments. CoRR abs/1804.03967 (2018). http://arxiv.org/abs/1804.03967
10. Hashmi, M., Governatori, G., Lam, H.-P., Wynn, M.T.: Are we done with business process compliance: state of the art and challenges ahead. Knowl. Inf. Syst. **57**(1), 79–133 (2018). https://doi.org/10.1007/s10115-017-1142-1
11. Hüllermeier, E., Waegeman, W.: Aleatoric and epistemic uncertainty in machine learning: an introduction to concepts and methods. Mach. Learn. **110**(3), 457–506 (2021). https://doi.org/10.1007/s10994-021-05946-3
12. Ly, L.T., Maggi, F.M., Montali, M., Rinderle-Ma, S., van der Aalst, W.M.P.: Compliance monitoring in business processes: functionalities, application, and tool-support. Inf. Syst. **54**, 209–234 (2015). https://doi.org/10.1016/j.is.2015.02.007
13. Maisenbacher, M., Weidlich, M.: Handling concept drift in predictive process monitoring. In: IEEE International Conference on Services Computing, pp. 1–8 (2017). https://doi.org/10.1109/SCC.2017.10

14. Mangat, A.S., Rinderle-Ma, S.: Next-activity prediction for non-stationary processes with unseen data variability. In: Almeida, J.P.A., Karastoyanova, D., Guizzardi, G., Montali, M., Maggi, F.M., Fonseca, C.M. (eds.) EDOC 2022. LNCS, vol. 13585, pp. 145–161. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-17604-3_9
15. Mangler, J., Rinderle-Ma, S.: Cloud process execution engine: architecture and interfaces (2022). https://doi.org/10.48550/ARXIV.2208.12214
16. Márquez-Chamorro, A.E., Resinas, M., Ruiz-Cortés, A.: Predictive monitoring of business processes: a survey. IEEE Trans. Serv. Comput. **11**(6), 962–977 (2018). https://doi.org/10.1109/TSC.2017.2772256
17. Park, G., Benzin, J., van der Aalst, W.M.P.: Detecting context-aware deviations in process executions. In: Di Ciccio, C., Dijkman, R., del Río Ortega, A., Rinderle-Ma, S. (eds.) BPM 2022. LNBIP, pp. 190–206. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-16171-1_12
18. Pauwels, S., Calders, T.: Incremental predictive process monitoring: the next activity case. In: Polyvyanyy, A., Wynn, M.T., Van Looy, A., Reichert, M. (eds.) BPM 2021. LNCS, vol. 12875, pp. 123–140. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-85469-0_10
19. Polato, M., Sperduti, A., Burattin, A., Leoni, M.: Time and activity sequence prediction of business process instances. Computing **100**(9), 1005–1031 (2018). https://doi.org/10.1007/s00607-018-0593-x
20. Rizzi, W., Di Francescomarino, C., Ghidini, C., Maggi, F.M.: How do I update my model? On the resilience of Predictive Process Monitoring models to change. Knowl. Inf. Syst. (9), 1–32 (2022). https://doi.org/10.1007/s10115-022-01666-9
21. Stertz, F., Rinderle-Ma, S., Mangler, J.: Analyzing process concept drifts based on sensor event streams during runtime. In: Fahland, D., Ghidini, C., Becker, J., Dumas, M. (eds.) BPM 2020. LNCS, vol. 12168, pp. 202–219. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58666-9_12
22. Teinemaa, I., Dumas, M., Maggi, F.M., Di Francescomarino, C.: Predictive business process monitoring with structured and unstructured data. In: La Rosa, M., Loos, P., Pastor, O. (eds.) BPM 2016. LNCS, vol. 9850, pp. 401–417. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-45348-4_23
23. Teinemaa, I., Dumas, M., Rosa, M.L., Maggi, F.M.: Outcome-oriented predictive process monitoring: review and benchmark. ACM Trans. Knowl. Discov. Data **13**(2), 17:1-17:57 (2019). https://doi.org/10.1145/3301300
24. Verenich, I., Dumas, M., Rosa, M.L., Maggi, F.M., Teinemaa, I.: Survey and cross-benchmark comparison of remaining time prediction methods in business process monitoring. ACM Trans. Intell. Syst. Technol. **10**(4), 34:1-34:34 (2019). https://doi.org/10.1145/3331449
25. Weinzierl, S., Revoredo, K.C., Matzner, M.: Predictive business process monitoringwith context information from documents. In: 27th European Conference on Information Systems - Information Systems for a Sharing Society, ECIS 2019, Stockholm and Uppsala, Sweden, June 8–14, 2019 (2019). https://aisel.aisnet.org/ecis2019_rip/59
26. Weytjens, H., Weerdt, J.D.: Learning uncertainty with artificial neural networks for predictive process monitoring. Appl. Soft Comput. 109134 (2022). https://doi.org/10.1016/j.asoc.2022.109134

27. Xian, Y., Lampert, C.H., Schiele, B., Akata, Z.: Zero-shot learning - a comprehensive evaluation of the good, the bad and the ugly. IEEE Trans. Pattern Anal. Mach. Intell. **41**(9), 2251–2265 (2019). https://doi.org/10.1109/TPAMI.2018.2857768
28. Yeshchenko, A., Durier, F., Revoredo, K., Mendling, J., Santoro, F.: Context-aware predictive process monitoring: the impact of news sentiment. In: Panetto, H., Debruyne, C., Proper, H.A., Ardagna, C.A., Roman, D., Meersman, R. (eds.) OTM 2018. LNCS, vol. 11229, pp. 586–603. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-02610-3_33