

Neuro-Symbolic Predictive Process Monitoring

Axel Mezini^{a,1}, Elena Umili^{b,1}, Ivan Donadello^a, Fabrizio Maria Maggi^a,
Matteo Mancanelli^b, Fabio Patrizi^b

^a*Faculty of Engineering, Free University of Bozen-Bolzano, NOI Techpark - via Bruno
Buozi, 1, Bolzano, 39100, Italy*

^b*Department of Computer, Control and Management Engineering, Sapienza, Università
di Roma, Via Ariosto, 25, Roma, 00185, Italy*

Abstract

This paper addresses the problem of suffix prediction in Business Process Management (BPM) by proposing a Neuro-Symbolic Predictive Process Monitoring (PPM) approach that integrates data-driven learning with temporal logic-based prior knowledge. While recent approaches leverage deep learning models for suffix prediction, they often fail to satisfy even basic logical constraints due to the absence of explicit integration of domain knowledge during training. We propose a novel method to incorporate Linear Temporal Logic over finite traces (LTL_f) into the training process of autoregressive sequence predictors. Our approach introduces a differentiable logical loss function, defined using a soft approximation of LTL_f semantics and the Gumbel-Softmax trick, which can be combined with standard predictive losses. This ensures the model learns to generate suffixes that are both accurate and logically consistent. Experimental evaluation on three real-world datasets shows that our method improves suffix prediction accuracy and compliance with temporal constraints. We also introduce two variants of the logic loss (local and global) and demonstrate their effectiveness under noisy and realistic settings. While developed in the context of BPM, our framework is applicable to any symbolic sequence generation task and contributes toward advancing Neuro-Symbolic AI.

Keywords: Suffix prediction, Neuro-Symbolic AI, Deep learning with logical knowledge, Linear Temporal Logic, Differentiable automata

¹Equal contribution

1. Introduction

Predictive Process Monitoring (PPM) is a field within Business Process Management (BPM) that focuses on forecasting future events or outcomes of ongoing process instances based on historical event data. This paper addresses the PPM problem of suffix prediction by leveraging both data and prior logical temporal knowledge. Suffix prediction is particularly important in BPM for forecasting the continuation of a process trace, enabling better resource allocation, and anticipating future steps for effective decision-making.

Recently, there has been significant interest in employing deep learning techniques for suffix prediction in BPM [1], including the use of Recurrent Neural Networks (RNNs) [2], Transformers [3], and Deep Reinforcement Learning algorithms [4]. Despite significant advances in machine learning and deep learning, several studies have shown that deep models can surprisingly fail to satisfy even the most basic logical constraints [5, 6], derived from commonsense reasoning or domain-specific knowledge. This occurs because such models are trained solely on data, and integrating logical knowledge into the training process remains an open challenge. As a result, in many domains like BPM, where both data and formal knowledge about the process are often available, the latter is typically underutilized.

However, domain-specific knowledge can play a crucial role in characterizing the context in which a process is executed, providing valuable information not explicitly contained in the event data alone. For example, when the particular variant of a process being executed is known (such as a client-specific scenario or a seasonally influenced workflow) this knowledge can guide predictions in ways that purely data-driven models cannot. Similarly, in environments subject to concept drift, where process behavior evolves over time, logical constraints can help identify and adapt to changes more robustly than relying solely on historical data. Furthermore, logical knowledge is instrumental in filtering out noise in the data, enhancing model robustness and prediction accuracy, as demonstrated in our experimental evaluation.

This work explores a novel Neuro-Symbolic PPM approach aimed at bridging this gap by incorporating prior knowledge, expressed in Linear Temporal Logic over finite traces (LTL_f), into the training of a deep generative model for suffix prediction. Existing works on knowledge-constrained sequence generation using autoregressive models are typically designed for test-time inference [7, 8, 9, 10, 11, 12, 13, 14], rather than for integration during training. A notable exception is STLnet [6], which incorporates Sig-

nal Temporal Logic (STL) specifications into the training process through a student-teacher framework. However, STLnet is specifically designed for continuous temporal sequences. Our attempts to adapt STLnet to discrete domains and LTL_f -based knowledge yielded poor results, highlighting its limitations in symbolic settings such as those encountered in PPM.

Our contribution is a Neuro-Symbolic method to integrate symbolic knowledge of certain process properties, expressed in Linear Temporal Logic over finite traces (LTL_f), into the training process of a neural sequence predictor. This enables us to leverage both sources of information (data and background LTL_f knowledge) during training. Specifically, we achieve this by defining a differentiable counterpart of the LTL_f knowledge and employing a differentiable sampling technique known as the Gumbel-Softmax trick [15]. By combining these two elements, we define a logical loss function that can be used alongside any other loss function employed by the predictor. This ensures that the network learns to generate traces that are both similar to those in the training dataset and compliant with the given temporal specifications *at the same time*.

We evaluate our method on several real-world BPM datasets and demonstrate that incorporating LTL_f knowledge at training time leads to predicted suffixes with both lower Damerau-Levenshtein (DL) distance [16] from the target suffixes and a significantly higher rate of satisfaction of the LTL_f constraints.

This paper builds upon our previous work [17], extending the approach to make it more robust and principled. We propose two distinct logic loss formulations: one that provides local, activity-level feedback, and another that enforces global, trace-level logical constraints during suffix prediction. In addition, we significantly broaden the empirical evaluation of our framework. We present a comprehensive set of results on real-world BPM datasets under challenging and noisy experimental conditions, demonstrating both the effectiveness and generality of our approach.

Overall, while our approach is instantiated in the context of PPM, its underlying principles are broadly applicable to any multi-step symbolic sequence generation task using autoregression. We believe the contributions of this work may be of general interest to the Machine Learning community, particularly in the area of Neuro-Symbolic AI.

The rest of the paper is structured as follows. Section 2 provides preliminary notions and notations necessary to understand the paper. Section 3 formulates the addressed problem and outlines the proposed solution. Sec-

tion 4 presents an experimental evaluation of the solution. Section 5 discusses related work, while Section 6 concludes the paper and spells out directions for future research.

2. Background and Notation

This section introduces the fundamental notions and notations essential for understanding the concepts discussed in the paper. It lays the groundwork by defining key terms and formalizing the terminology used throughout the work.

2.1. Notation

In this work, we consider *sequential* data of various types, including both symbolic and subsymbolic representations. Symbolic sequences are also called *traces*. Each element in a trace is a symbol σ drawn from a finite alphabet Σ . We denote sequences using bold notation. For example, $\boldsymbol{\sigma} = (\sigma_1, \sigma_2, \dots, \sigma_T)$ represents a trace of length T .

Each symbolic variable in the sequence can be grounded either categorically or probabilistically. In the case of categorical grounding, each element of the trace is assigned a symbol from Σ , denoted simply as σ_i , where σ_i can be encoded as an index in $\{1, 2, \dots, |\Sigma|\}$ or as a *one-hot vector* $\sigma_i \in \{0, 1\}^{|\Sigma|}$ such that $\sum_{j=1}^{|\Sigma|} \sigma_i[j] = 1$. In the case of probabilistic grounding, each symbolic variable is associated with a probability distribution over Σ , represented as a vector $\tilde{\sigma}_i \in \Delta(\Sigma)$, where $\Delta(\Sigma)$ denotes the probability simplex defined as:

$$\Delta(\Sigma) = \left\{ \tilde{\sigma} \in \mathbb{R}^{|\Sigma|} \left| \tilde{\sigma}[j] \geq 0, \sum_{j=1}^{|\Sigma|} \tilde{\sigma}[j] = 1 \right. \right\}.$$

Accordingly, we distinguish between categorically grounded sequences $\boldsymbol{\sigma}$ and probabilistically grounded sequences $\tilde{\boldsymbol{\sigma}}$, using the tilde notation.

Finally, we use subscripts to indicate time steps in the sequence, and $v[j]$ ($M[j]$) to denote the j -th component (tensor) of vector v (matrix M). For instance, $\tilde{\sigma}_i[j]$ denotes the j -th component of the probabilistic grounding of σ at time step i . We also use $+$ to denote trace concatenation; e.g., $\boldsymbol{a} + \boldsymbol{b}$ is the trace obtained by concatenating traces \boldsymbol{a} and \boldsymbol{b} .

2.2. Linear Temporal Logic and Deterministic Finite Automata

Linear Temporal Logic (LTL) [18] is a formal language that extends traditional propositional logic with modal operators, allowing the specification of rules that must hold *through time*. In this work, we use LTL interpreted over finite traces (LTL_f) [19], which models finite, but length-unbounded, process executions, making it suitable for finite-horizon problems. Given a finite set Σ of atomic propositions, the set of LTL_f formulas ϕ is inductively defined as follows:

$$\phi ::= \top \mid \perp \mid \sigma \mid \neg\phi \mid \phi \wedge \phi \mid X\phi \mid \phi U \phi, \quad (1)$$

where $\sigma \in \Sigma$. We use \top and \perp to denote true and false, respectively. X (Strong Next) and U (Until) are temporal operators. Other temporal operators are N (Weak Next) and R (Release), defined as $N\phi \equiv \neg X\neg\phi$ and $\phi_1 R \phi_2 \equiv \neg(\neg\phi_1 U \neg\phi_2)$; G (globally) $G\phi \equiv \perp R \phi$; and F (eventually) $F\phi \equiv \top U \phi$.

A trace $\sigma = (\sigma_1, \sigma_2, \dots, \sigma_T)$ is a sequence of propositional assignments to the propositions in Σ , where $\sigma_t \subseteq \Sigma$ is the set of all and only propositions that are true at instant t . Additionally, $|\sigma| = T$ denotes the length of the trace. Since every trace is finite, $|\sigma| < \infty$. If the propositional symbols in Σ are all *mutually exclusive*, i.e., the domain produces exactly one symbol true at each step, then we have $\sigma_t \in \Sigma$. As is customary in BPM, we make this assumption, known as the *mutual exclusivity assumption* [20].

By $\sigma \models \phi$ we denote that the trace σ satisfies the LTL_f formula ϕ . We refer the reader to [19] for a formal description of the LTL_f semantics. Any LTL_f formula ϕ can be translated into a Deterministic Finite Automaton (DFA) [19] $A_\phi = (\Sigma, Q, q_0, \delta, F)$, where Σ is the automaton alphabet,² Q is the finite set of states, $q_0 \in Q$ is the initial state, $\delta : Q \times \Sigma \rightarrow Q$ is the transition function, and $F \subseteq Q$ is the set of final states. Additionally, we recursively define the extended transition function over traces $\delta^* : Q \times \Sigma^* \rightarrow Q$ as:

$$\begin{aligned} \delta^*(q, \epsilon) &= q \\ \delta^*(q, \sigma + \mathbf{x}) &= \delta^*(\delta(q, \sigma), \mathbf{x}), \end{aligned} \quad (2)$$

²Here the alphabet of the equivalent DFA is Σ because of the mutual exclusivity assumption. In the general case, the automaton alphabet is 2^Σ .

where $\sigma \in \Sigma$ is a symbol and $\mathbf{x} \in \Sigma^*$ is a trace. The automaton accepts the trace σ if $\delta^*(q_0, \sigma) \in F$, and in that case we say that σ belongs to the language of the automaton, denoted as $L(A_\phi)$. We have that ϕ and A_ϕ are equivalent because, for any trace $\sigma \in \Sigma^*$:

$$\sigma \in L(A_\phi) \iff \sigma \models \phi. \quad (3)$$

2.3. Deep Autoregressive Models and Suffix Prediction

Deep autoregressive models are a class of deep learning models that automatically predict the next component in a sequence by using the previous elements in the sequence as inputs. These models can be applied to both continuous and categorical (symbolic) data, finding applications in various generative AI tasks such as Natural Language Processing (NLP) and Large Language Models (LLM) [21, 22], image synthesis [23, 24], and time-series prediction [25]. They encompass deep architectures such as RNNs and Transformers and, in general, any neural model capable of estimating the probability:

$$P(x_t \mid x_1, x_2, \dots, x_{t-1}) = P(x_t \mid \mathbf{x}_{<t}). \quad (4)$$

The probability of a sequence of data \mathbf{x} can be calculated as:

$$P(\mathbf{x}) = \prod_{i=1}^T P(x_i \mid \mathbf{x}_{<i}). \quad (5)$$

In suffix prediction in BPM, given a subsequence (or prefix) of *past activities* $\mathbf{p}_t = (a_1, a_2, \dots, a_t)$ that the process has produced up to the current time step t , with a_i in a finite set of activities \mathcal{A} , we aim to complete the trace by generating the sequence of future events, also called the *suffix*, $\mathbf{s}_t = (a_{t+1}, \dots, a_{t+l}, \text{EOT})$. We use $\text{EOT} \notin \mathcal{A}$ to denote a special symbol marking the end of sequences, which is not included in \mathcal{A} .

Suffix prediction can be accomplished using autoregressive models, by choosing at each step the most probable next event according to the neural network, concatenating it with the prefix, and continuing to predict the next event in this manner until the **EOT** symbol is predicted or the trace has reached a maximum number of steps T . We define $\mathcal{A}^{\leq T} \text{EOT} \equiv \{\mathbf{a} + \text{EOT} \mid \mathbf{a} \in \mathcal{A}^{\leq T}\}$ as the set of possible complete traces $\mathbf{a} = \mathbf{p}_t + \mathbf{s}_t$ that can be generated in this way.

At each generation step, a symbol must be *sampled* from the next activity probability. A common way of selecting the next activity to feed into the

autoregressor is to *greedily* choose the activity maximizing the next symbol probability at each step, as follows:

$$a_k = \operatorname{argmax}_{a \in \mathcal{A} \cup \{\text{EOT}\}} P(a_t = \sigma \mid a_1, \dots, a_t, a_{t+1}, \dots, a_{k-1}), \quad t < k \leq T. \quad (6)$$

This greedy search strategy may not produce the *most probable suffix*, i.e., the trace that maximizes the probability in Equation 5. Other non-optimal sampling strategies commonly used for this task include Beam Search, Random Sampling, and Temperature Sampling [4].

3. Method

This section defines the specific suffix prediction problem we want to solve and outlines our Neuro-Symbolic PPM approach, which integrates prior knowledge expressed in Linear Temporal Logic over finite traces (LTL_f) into the training of a deep generative model. We also introduce two logic loss formulations: one providing *local*, activity-level feedback, and another enforcing *global*, trace-level constraints over the predicted suffix.

3.1. Problem Formulation

We assume an autoregressive neural model f_θ with trainable parameters θ to estimate an approximation P_θ of the probability of the next event a_t given a trace of previous events $\mathbf{a}_{<t}$ (Equation 4):

$$\begin{aligned} \tilde{y}_t &= f_\theta(\mathbf{a}_{<t}) \\ P(a_t = a_i \in \mathcal{A} \cup \{\text{EOT}\} \mid \mathbf{a}_{<t}) &\approx \tilde{y}_t[i]. \end{aligned} \quad (7)$$

Note that we do not make any assumptions about the neural model, except that it can estimate the probability of the next activity given a sequence of previous ones. As a result, our approach is entirely *model-agnostic* and can be readily applied to any autoregressive model.

We denote by P_θ the probability of a trace according to the network approximation:

$$P_\theta(\mathbf{a}) = \prod_{t=1}^{|\mathbf{a}|} \tilde{y}_t[a_t]. \quad (8)$$

The model parameters are typically trained using a supervised loss $L_{\mathcal{D}}$, evaluated on a dataset \mathcal{D} of ground-truth traces obtained by observing the process. The loss for a trace $\mathbf{a} \in \mathcal{D}$ of length T is defined as follows:

$$L_{\mathcal{D}}(\mathbf{a}) = \frac{1}{T} \sum_{t=1}^T \text{cross-entropy}(f_{\theta}(\mathbf{a}_{<t}), a_t). \quad (9)$$

This loss trains the network to predict the next symbol in a trace so as to closely mimic the data in the dataset.

In this work, we assume that certain properties of the process are also known and can be injected into the learning process during training. These properties, which form the *background* (or *prior*) knowledge about the process, are expressed as an LTL_f formula ϕ defined over an alphabet $\Sigma \subseteq \mathcal{A}$.

Our goal is for the language generated by the autoregressor f_{θ} to be strictly contained within the language of strings accepted by the formula, denoted as $L(A_{\phi})$. However, the language produced by the network is *unbounded*, as it is only *softly assigned*: in other words, the network can generate any possible string, each with a different probability.

Our method therefore aims to maximize the probability $P_{\theta \models \phi}$ that traces $\mathbf{a} \sim P_{\theta}$, sampled from the autoregressor, satisfy the specification:

$$P_{\theta \models \phi} = \mathbb{E}_{\mathbf{a} \sim P_{\theta}}[\mathbf{a} \models \phi] = \sum_{\mathbf{a} \in \mathcal{A}^{\leq T} \text{EOT}} P_{\theta}(\mathbf{a}) \mathbb{1}\{\mathbf{a} \models \phi\}. \quad (10)$$

Note that in order to compute the exact probability of knowledge satisfaction, one would need to sample *all possible suffixes* of maximum length T and sum their acceptance probabilities. However, this set has an exponential size, making exact computation unfeasible. Furthermore, we aim to impose the maximization of this probability as a *training objective*. Therefore, our goal is to design a fast and differentiable procedure to approximate it at each optimization step of the autoregressor.

We propose two logic loss functions: a local, activity-level loss L_{ϕ}^{loc} , and a global, trace-level loss L_{ϕ}^{glob} . We show that both contribute positively to the learning process, demonstrating the effectiveness of integrating LTL_f knowledge into autoregressive training. The choice between the two logic losses depends on the type of LTL_f knowledge available. The global loss L_{ϕ}^{glob} can always be applied, regardless of the structure of the formula ϕ , but is more computationally demanding since it requires generating entire suffixes during training.

In general, the compliance of a trace with ϕ can only be assessed on complete traces. Indeed, whether the current partial prediction $\mathbf{a}^{\leq t}$ satisfies (or violates) ϕ does not guarantee that the final predicted trace \mathbf{a} will also satisfy (or violate) it. This introduces a challenge in evaluating LTL_f constraints, as opposed to approaches that rely on local constraints [14], which can be verified step-by-step during generation.

However, some types of formulas, such as safety constraints, once translated into a DFA, contain *failing states*, which can be used to guide generation locally. A failing state is a state in the DFA from which no accepting state is reachable. When a partial trace enters such a state, it has irreversibly violated the formula, and no continuation can satisfy the knowledge. Our local loss L_ϕ^{loc} exploits the presence of failing states to guide the autoregressor at every generation step. As a result, it provides richer feedback at the activity level, but it can only be used with formulas that admit such a DFA structure.

In the following sections, we describe how we compute the logic loss under the two scenarios. In either case, we combine it with the supervised loss $L_{\mathcal{D}}$ as follows:

$$L = \alpha L_{\mathcal{D}} + (1 - \alpha) L_\phi, \quad (11)$$

with α being a constant between 0 and 1 that balances the influence of each loss on the training process.

3.2. Local Guidance

In cases where the formula can be permanently violated, the corresponding DFA includes a set of failing states $Q^{\text{fail}} \subseteq Q$, from which the formula can no longer be satisfied. In this case, we aim to minimize the *probability that the next predicted activity will irreversibly violate prior knowledge*, denoted as $P(\mathbf{a}_t \not\models^\times \phi \mid \mathbf{a}_{<t})$. Here, the symbol $\not\models^\times$ denotes the permanent violation of the formula. Given a ground-truth trace $\mathbf{a} \in \mathcal{D}$, we define this probability as:

$$P(\mathbf{a}_t \not\models^\times \phi \mid \mathbf{a}_{<t}) = \sum_{a \in \mathcal{A} \cup \text{EOT}} f_\theta(\mathbf{a}_{<t})[a] \cdot \mathbb{1} \{ \delta^*(q_0, \mathbf{a}_{<t} + a) \in Q^{\text{fail}} \}. \quad (12)$$

In other words, at each step t of the trace, we consider the probability distribution over the next symbol given by $f_\theta(\mathbf{a}_{<t})$. For each symbol a in the vocabulary (including **EOT**), we simulate the state reached by the extended trace $\mathbf{a}_{<t} + a$ in the DFA. If the resulting state belongs to Q^{fail} , we add the probability of a to the probability that the trace permanently violates the

knowledge. We minimize this probability by minimizing the following loss on each ground-truth trace \mathbf{a} :

$$L_{\phi}^{\text{loc}} = \frac{1}{|\mathbf{a}|} \sum_{t=1}^{|\mathbf{a}|} -\log(1 - P(\mathbf{a}_t \not\models^{\times} \phi \mid \mathbf{a}_{<t})). \quad (13)$$

Note that traces can fall into one of the following categories: (i) they irreversibly violate the knowledge (by reaching a failure state either at termination or earlier); (ii) they do not satisfy the knowledge (by terminating in a non-accepting, non-failure state); (iii) they satisfy the knowledge (by terminating in an accepting state).

Both cases (i) and (ii) result in a violation of the knowledge. The loss L_{ϕ}^{loc} specifically targets only case (i), aiming to reduce occurrences of irreversible violations. By decreasing the probability in Equation 12, we increase the desired probability $P_{\theta \models \phi}$. However, knowledge satisfaction is not directly maximized, and it is still possible for traces with zero loss to violate the knowledge under case (ii). Despite this limitation, we observed in practice that L_{ϕ}^{loc} remains highly effective on many datasets, as we will show in Section 4.

3.3. Global Guidance

In the general case, an LTL_f formula ϕ cannot always be violated permanently. In such cases, it is impossible to supervise the suffix generation step by step, and the only guidance that can be provided to the autoregressor is *global*, i.e., applied to the entire generated trace. To this end, we define a second logic loss, L_{ϕ}^{glob} , which provides this global supervision to the autoregressor and can be applied to *any* LTL_f formula.

In particular, in the global case, we directly approximate the target probability $P_{\theta \models \phi}$ using a Monte Carlo estimation. We sample a set of complete traces $\{\mathbf{a}^{(1)}, \mathbf{a}^{(2)}, \dots, \mathbf{a}^{(N)}\} \sim P_{\theta}$ according to the distribution learned by the autoregressor, and calculate an approximation of the target probability $\hat{P}_{\theta \models \phi}$ as the empirical average compliance with the formula over the sampled set:

$$\hat{P}_{\theta \models \phi} = \frac{1}{N} \sum_{i=1}^N \mathbf{1}\{\mathbf{a}^{(i)} \models \phi\}. \quad (14)$$

Finally, we define the global loss as:

$$L_{\phi}^{\text{glob}} = -\log(\hat{P}_{\theta \models \phi}). \quad (15)$$

In order to maximize this estimate during training, we must be able to sample complete traces and evaluate their compliance with the knowledge in a fast and differentiable way. To achieve this, our method relies on two key components:

1. DeepDFA [26], a Neuro-Symbolic framework that encodes temporal logic properties as a recurrent layer, enabling efficient and differentiable evaluation of logical constraints;
2. Gumbel-Softmax sampling [15, 27] to generate differentiable, near one-hot suffixes during training.

By leveraging these two components, detailed in the following sections, we compute a *global logic loss* L_ϕ^{glob} that enforces the satisfaction of prior knowledge over entire traces. In the next sections, we first describe how the LTL_f formula ϕ is preprocessed and encoded using DeepDFA. We then illustrate how DeepDFA is integrated with suffix prediction during training through differentiable sampling based on Gumbel-Softmax.

3.4. Knowledge Preprocessing and Tensorization

In this section, we describe how the LTL_f formula ϕ is preprocessed and *tensorized* to enable the integration of prior knowledge into the learning process. Note that these steps are performed only once before training begins and are required for computing both the local and global losses.

3.4.1. Knowledge Preprocessing

First, we translate the LTL_f formula ϕ into an equivalent deterministic finite automaton (DFA) $A_\phi = (\Sigma, Q, q_0, \delta, F)$ using the automatic translation tool `ltlf2DFA` [28]. This step is necessary because all existing approaches for integrating temporal specifications into learning pipelines rely on automata-based representations [26, 29, 30], while the direct integration of LTL_f formulas remains an open challenge [31]. Although this translation has worst-case *double-exponential* complexity [19], it is often efficient in practice, and several scalable techniques exist to perform it for a given formula ϕ [32, 33, 34]. Moreover, in this work, we focus on *Declare* formulas [35], a standard for declaratively specifying business processes [36], which are known to yield DFAs of *polynomial size* with respect to the input formula [37].

Second, we adapt the DFA alphabet $\Sigma \subseteq \mathcal{A}$ so that it includes all activity symbols present in the event log \mathcal{A} . Specifically, for each symbol $s \in \mathcal{A} \setminus$

Σ and each state $q \in Q$, we add a self-loop $\delta(q, s) = q$ to the transition function δ . This ensures that symbols not constrained by the formula can still be processed by the DFA, without affecting its acceptance behavior. Note that while this technique is feasible in the BPM setting, it may become impractical in other application domains where the autoregressor’s symbol space is excessively large, such as in LLM-based applications [8].

Additionally, we extend the DFA to handle the special **EOT** (End of Trace) symbol. In particular, we define as accepted all and only those traces of the form $t+\text{EOT}+z$ such that $t \in \mathcal{A}^*$, $t \models \phi$, and z is any (possibly empty) trace in $(\mathcal{A} \cup \{\text{EOT}\})^*$. This implies that:

- (i) a non-terminating trace is considered *non-compliant* with the specification;
- (ii) a trace is evaluated *against the specification only at the first occurrence of EOT* – whether the specification is violated or satisfied *before* this point is irrelevant, as are any symbols in z occurring after the first **EOT**.

To implement this behavior, we add **EOT** to the alphabet and introduce two terminal states: a success state q_t^s and a failure state q_t^f . For each state $q \in Q$, we add the transition $\delta(q, \text{EOT}) = q_t^s$ if $q \in F$, and $\delta(q, \text{EOT}) = q_t^f$ otherwise. States q_t^s and q_t^f are *terminal*, meaning they are absorbing: once entered, the automaton cannot exit. Therefore, for each symbol $s \in \mathcal{A} \cup \{\text{EOT}\}$, we add the transitions $\delta(q_t^s, s) = q_t^s$ and $\delta(q_t^f, s) = q_t^f$. We designate q_t^s as the *only accepting state* of the extended DFA.

Finally, we compute the set of failure states Q^{fail} , which is only necessary when employing the local logic loss L_ϕ^{loc} . The final DFA after preprocessing is therefore:

$$A'_\phi = (\mathcal{A} \cup \{\text{EOT}\}, Q \cup \{q_t^s, q_t^f\}, q_0, \delta', \{q_t^s\}),$$

with the extended transition function δ' defined as:

$$\delta'(q, s) = \begin{cases} \delta(q, s) & \text{if } q \in Q \wedge s \in \Sigma, \\ q & \text{if } s \in \mathcal{A} \setminus \Sigma \vee q \in \{q_t^s, q_t^f\}, \\ q_t^s & \text{if } s = \text{EOT} \wedge q \in F, \\ q_t^f & \text{if } s = \text{EOT} \wedge q \notin F. \end{cases} \quad (16)$$

3.4.2. Knowledge Tensorization

Given the final DFA A'_ϕ , we transform it into a neural layer with DeepDFA [26]. DeepDFA is a neural, probabilistic relaxation of a standard determinis-

tic finite-state machine, where the automaton is represented in matrix form and the input symbols, states, and outputs are *probabilistically grounded*.

We define the transition function of the DFA in matrix form as $\mathcal{T} \in \mathbb{R}^{|\Sigma| \times |Q| \times |Q|}$, and the initial and final states with the vectors $\mu \in \mathbb{R}^{|Q|}$ and $\lambda \in \mathbb{R}^{|Q|}$, respectively. While this matrix representation is traditionally used for Probabilistic Finite Automata (PFA), here it is applied to deterministic automata to leverage tensor operations for fast and differentiable computation of state transitions and outputs. The DeepDFA model is defined as follows:

$$\begin{aligned}\tilde{q}_0 &= \mu, \\ \tilde{q}_t &= \sum_{j=1}^{|\Sigma|} \tilde{\sigma}_t[j] \cdot (\tilde{q}_{t-1} \cdot \mathcal{T}[j]), \\ \tilde{o}_t &= \tilde{q}_t \cdot \lambda^\top.\end{aligned}\tag{17}$$

Here, $\tilde{\sigma}_t$, \tilde{q}_t , and \tilde{o}_t represent the probabilistic representations of the input symbol, the automaton state, and the output (i.e., whether the trace is accepted or rejected) at time t . The neural network parameters μ , \mathcal{T} , and λ are initialized from the DFA as:

$$\begin{aligned}\mu[j] &= \begin{cases} 1 & \text{if } q_j = q_0 \\ 0 & \text{otherwise} \end{cases} \\ \mathcal{T}[s, q_i, q_j] &= \begin{cases} 1 & \text{if } \delta(q_i, s) = q_j \\ 0 & \text{otherwise} \end{cases} \\ \lambda[j] &= \begin{cases} 1 & \text{if } q_j \in F \\ 0 & \text{otherwise.} \end{cases}\end{aligned}\tag{18}$$

3.5. Differentiable Sampling

For the computation of the global logic loss, our goal is to generate complete suffixes and evaluate their compliance with the knowledge constraint during training. To this end, note that the next-activity predictor is trained on *perfectly one-hot* (i.e., symbolic) input sequences $\mathbf{a}_{\leq t}$ and produces continuous probability vectors $\tilde{\mathbf{y}}^{(t)}$ as output, which can differ significantly from one-hot vectors. As a result, we cannot directly feed these probability vectors back into the network as inputs in subsequent steps, as doing so may lead to unpredictable behavior. Instead, we need to *sample* from these distributions

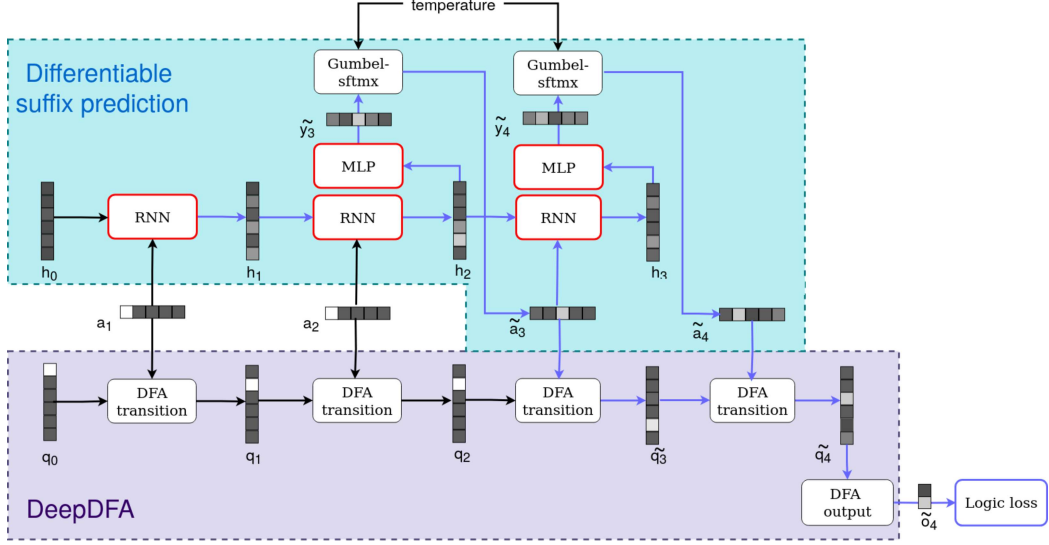


Figure 1: Global logic loss computation using a differentiable procedure for both suffix generation and formula evaluation. Violet arrows indicate the connections through which the loss is back-propagated, while components highlighted with a red border denote the modules whose parameters are updated by the logic loss.

to recover one-hot-like inputs. At the same time, this sampling must remain differentiable to enable backpropagation.

The computation of the global logic loss proceeds as follows:

1. given a prefix \mathbf{p}_t , generate N suffixes $\tilde{\mathbf{s}}_t^{(i)}$ that are simultaneously: (i) highly probable under the network’s distribution; (ii) differentiable; and (iii) *nearly symbolic*, i.e., as close as possible to one-hot vectors;
2. evaluate whether the generated complete traces $\mathbf{p}_t + \tilde{\mathbf{s}}_t^{(i)}$ satisfy the LTL_f formula ϕ using DeepDFA, which supports evaluation over both categorically grounded and probabilistically grounded traces;³
3. maximize the estimated satisfaction probability $\hat{P}_{\theta=\phi}$, computed as the empirical mean over the sampled traces.

Thanks to the differentiability of both the knowledge evaluator and the

³Note that the prefix \mathbf{p}_t is categorically grounded, whereas the suffix $\tilde{\mathbf{s}}_t$ is probabilistically grounded.

sampling process, the resulting loss can be back-propagated through the generated suffixes and used to update the parameters θ of the next-activity predictor, as illustrated in Figure 1. To sample the next activity \tilde{a}_t from the probability distribution \tilde{y}_t produced by the network (while ensuring differentiability), we employ the Gumbel-Softmax reparameterization trick [15, 27]:

$$\tilde{a}_t = \text{softmax} \left(\frac{\log(\tilde{y}_t) + G}{\tau} \right), \quad (19)$$

where G is a random vector sampled from the Gumbel distribution, and τ is a temperature parameter that controls the sharpness of the output distribution. As $\tau \rightarrow 0$, the output approaches a discrete one-hot vector, while for $\tau = 1$, it remains close to the original continuous probabilities in \tilde{y}_t . Since the next activity is only *probabilistically* grounded, we denote it as \tilde{a}_t .

4. Experiments

This section describes the experimental setup, including the datasets, evaluation metrics, and comparative approaches, followed by a detailed analysis of the results obtained. The experiments are reproducible using our implementations of the GLL and LLL at <https://github.com/axelmezini/suffix-prediction> and <https://github.com/axelmezini/nesy-suffix-prediction-dfa> respectively.

4.1. Experimental Setup

Our methods have been evaluated using three real-world datasets.⁴ Given an event log, the traces are ordered by the timestamp of the first event and split into training and test sets using an 80-20 ratio. Knowledge is extracted from the test set in the form of Declare constraints using the Declare Miner [38], each with a minimum support value of 85% (i.e., satisfied in at least 85% of the traces). These constraints are then translated into their corresponding LTL_f formulas and combined in an LTL_f model using conjunctions. To enable controlled experiments, traces that violate the extracted LTL_f model (i.e., that do not satisfy all the discovered constraints) are removed from both the training and test set.

⁴<https://www.tf-pm.org/resources/logs>

In addition, to the training set containing only positive traces four levels of noise are introduced: 10%, 20%, 30%, and 40%. Noise is applied by randomly replacing the activity label of selected events with other labels from the activity vocabulary. During both training and testing, different prefix lengths are used, determined based on the median trace length of the training split of the log. For each distinct configuration, 15 runs are executed.

	BPIC 2013	BPIC 2020	Sepsis
# Traces	818	2395	690
# Activities	7	51	16
# Constraints	7	56	39
# DFA states	6	13	10
# Failure states	2	2	2

Table 1: Key statistics of the datasets used in our experiments.

Table 1 summarizes the key statistics of the datasets used in our experiments. The datasets vary in size and complexity, with the number of traces ranging from 690 to 2395, and the number of distinct activities ranging from 7 to 51. Correspondingly, the number of discovered Declare constraints and the size of the resulting deterministic finite automata (DFA) differ across datasets, reflecting their behavioral complexity. Despite these differences, all datasets include a small number of failure states in their DFAs, which indicate states violating the discovered constraints.

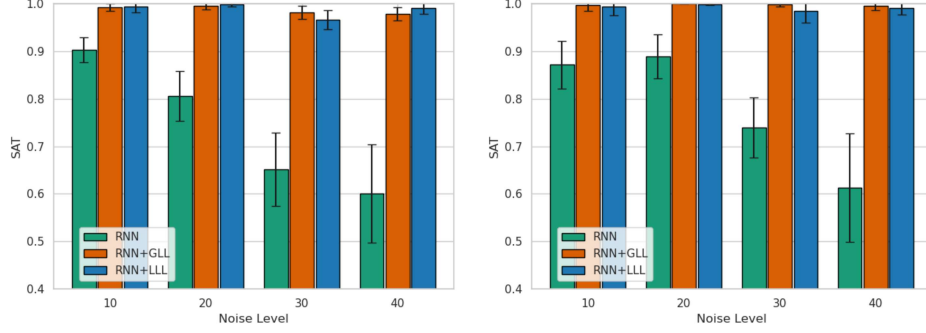
Baselines. We base our evaluation on RNN-based next activity predictors. For each dataset configuration, we test: (i) a base RNN model trained only with supervised loss (**RNN**), (ii) RNNs trained with supervised and local logic loss (**RNN+LLL**), and (iii) RNNs trained with supervised and global logic loss (**RNN+GLL**). All RNNs used are two-layer LSTMs with 100 neurons per layer, trained using a batch size of 64 and the Adam optimizer. Two different sampling strategies are tested: greedy decoding and temperature-based sampling.

We compare the proposed approaches using two evaluation metrics: the Damerau-Levenshtein similarity with respect to the ground-truth traces and the satisfaction rate with respect to the LTL_f model.

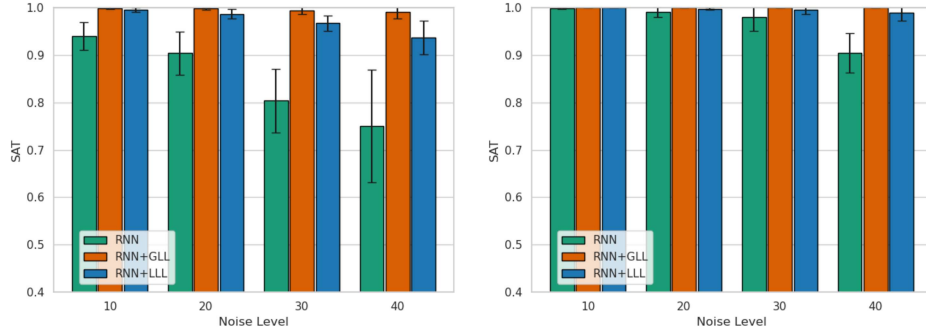
4.2. Empirical Results

Figures 3 and 2 show the performance across all datasets and noise levels for the satisfiability and similarity metrics, respectively.

Sepsis



BPIC 2013



BPIC 2020

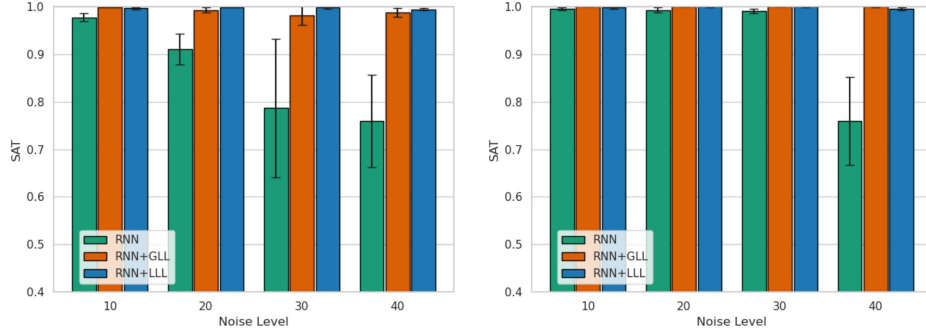
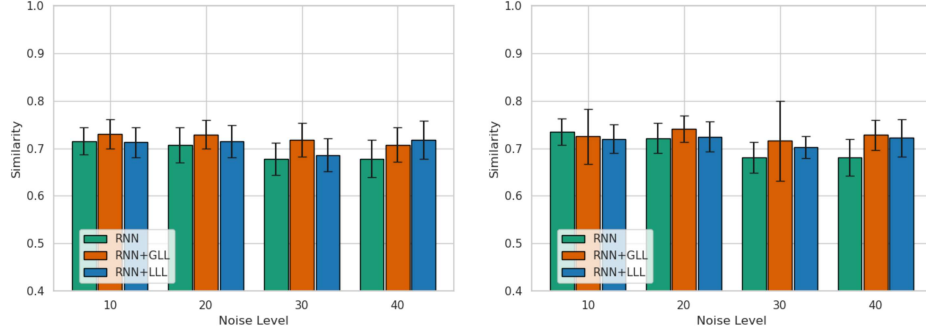
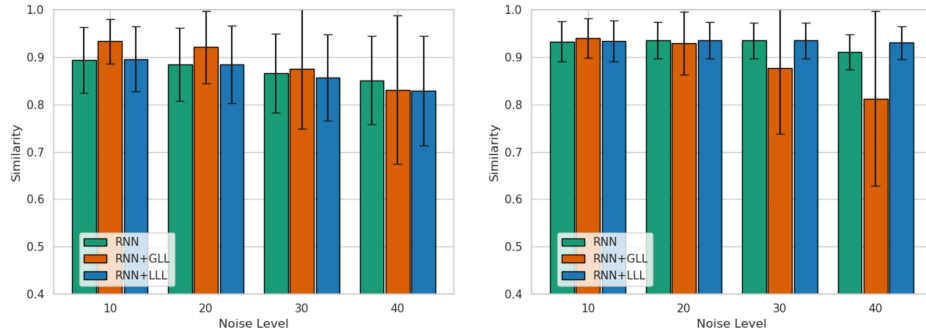


Figure 2: Comparison of satisfiability (SAT) of predicted traces with respect to logical constraints across three datasets (Sepsis, BPIC 2013, BPIC 2020) and multiple noise levels. Each row corresponds to a dataset, and each column shows temperature-based (left) and greedy (right) sampling. Methods compared: pure RNN, RNN with global logic loss (GLL), and RNN with local logic loss (LLL). Higher SAT scores indicate better compliance with logical constraints.

Sepsis



BPIC 2013



BPIC 2020

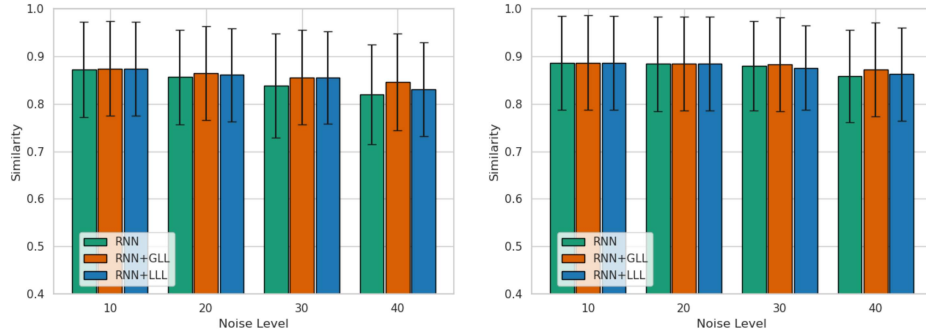


Figure 3: Comparison of predicted suffix similarity (based on scaled Damerau-Levenshtein distance) across three datasets (Sepsis, BPIC 2013, BPIC 2020) under varying levels of injected noise. Each row corresponds to a dataset, and each column shows the results for temperature-based (left) and greedy (right) sampling. The methods compared include: pure RNN, RNN with global logic loss (GLL), and RNN with local logic loss (LLL). Higher values indicate greater similarity to the ground truth suffixes.

The empirical results demonstrate that integrating background knowledge during training consistently increases the satisfaction rate of the predicted traces, regardless of the sampling strategy. These improvements are especially pronounced under higher noise levels, confirming the utility of incorporating logical background knowledge when predictive uncertainty increases. Notably, the satisfaction rate remains close to 100% even at the highest noise level of 40%.

Importantly, this increase in satisfaction rate does not negatively affect the model’s ability to learn from training data. The similarity to ground truth data remains consistent across configurations. These results suggest that the integrated knowledge helps the model distinguish compliant traces from noisy patterns in the training sets. Overall, the impact of the sampling strategy is limited, although greedy decoding often shows slightly better performance compared to temperature-based sampling for both metrics.

Differences observed across datasets likely reflect their inherent characteristics, such as vocabulary size, number and frequency of variants, and training data volume. Overall, our empirical evaluation confirms that integrating background knowledge at training time consistently improves performance and that the methodology is applicable across various real-world datasets and scenarios.

We also report the average number of training epochs required by each method in Table 2. RNN+GLL shows the most efficient training across all configurations, often converging in fewer than 600 epochs, compared to over 1500 epochs for the baseline. RNN+LLL requires more epochs than RNN+GLL but still fewer than the baseline in many cases. This proves that integrating background knowledge not only improves predictive quality but also accelerates model convergence.

5. Related Work

Recently, there has been significant interest in employing deep Neural Networks (NN) in PPM, for tasks such as next activity prediction, suffix prediction, and attribute prediction [1]. Despite significant advances in the field, nearly all works rely on training these models solely on data without utilizing any formal prior knowledge about the process. They mainly focus on two aspects: (i) enhancing the neural model, ranging from RNNs [39, 40, 7, 41], Convolutional NN (CNN) [42], Generative Adversarial Networks (GANs) [43, 41], Autoencoders [41], and Transformers [41]; and (ii) wisely

Dataset	Noise	RNN	RNN+GLL	RNN+LLL
BPIC 2013	10	1544.73	572.40	1080.80
BPIC 2013	20	1692.47	558.67	1158.27
BPIC 2013	30	1721.67	565.27	1194.60
BPIC 2013	40	1985.07	561.80	1228.40
BPIC 2020	10	1871.40	568.47	1364.47
BPIC 2020	20	1817.93	574.93	1634.33
BPIC 2020	30	1851.80	572.07	1947.33
BPIC 2020	40	1902.00	599.73	1999.00
SEPSIS	10	1406.67	577.73	1486.13
SEPSIS	20	1585.33	603.93	1873.87
SEPSIS	30	1488.40	576.87	1586.07
SEPSIS	40	1445.53	595.27	1789.27

Table 2: Average number of training epochs required by each method across datasets and noise levels. Best results in bold.

choosing the sampling technique to query the network at test time to generate the suffix, mostly using greedy search [39], random search [40], or beam search [7], and more recently, policies trained with Reinforcement Learning (RL) [44, 4]. Among all these works, only one exploits prior process knowledge [7], expressed as a set of LTL_f formulas, but it uses this knowledge only at test time, modifying the beam search sampling algorithm to select potentially compliant traces with the background knowledge.

In this work, we take a radically different approach by introducing a principled way to integrate background knowledge in LTL_f with a deep NN model for suffix prediction at *training time*. This is based on defining a logical loss that can be combined with the loss of any autoregressive neural model and any sampling technique at test time, drawing inspiration from the literature in Neuro-Symbolic AI [45]. In this field, many prior works focus on exploiting temporal logical knowledge in deep learning tasks, but none have been used for multi-step symbolic sequence generation.

T-leaf [46] creates a semantic embedding space to represent both formulas and traces and uses it in tasks such as sequential action recognition and imitation learning, which do not involve multi-step prediction. In [29], an extension of Logic Tensor Networks (LTN) [47] to represent fuzzy automata is proposed, and employed to integrate LTL_f background knowledge in image sequence classification tasks. STLnet [6] adopts a student-teacher training scheme where the student network proposes a suffix based on the data, that is corrected by the teacher network to satisfy the formula. This work uses Signal Temporal Logic (STL) formulas and is applied to continuous trajectories

rather than discrete traces. Our attempts to apply it to discrete data and LTL_f formulas translated into STL yielded poor results, as the resulting STL formulas were extremely challenging for the framework to handle.

A recent line of research focuses on constraining Large Language Models (LLMs) with structured temporal knowledge, either by employing constrained beam search [8, 9, 10], or training auxiliary models [11, 12], or exploiting conditioned sampling techniques [13, 14]. However, all these approaches are exclusively designed for test-time inference and have no influence on the training of the LLM. While this may be reasonable in the context of LLMs, where prior knowledge is often available only for specific subtasks, in PPM, structured global knowledge about the process may be available *before* data collection. In such cases, incorporating this knowledge during training, rather than only at inference time, can significantly benefit the learning process.

Our work is the first to integrate temporal knowledge in the generation of multi-step symbolic sequences at training time. It is based on encoding LTL_f formulas using a matrix representation that we previously used for very different tasks, such as learning RL policies for non-Markovian tasks [48] and inducing automata from a set of labeled traces [49], that we adapt here for use in the generative task of suffix prediction.

6. Conclusions and Future Work

This paper introduces a novel Neuro-Symbolic approach that seamlessly integrates temporal logic knowledge, expressed in LTL_f , into the training of neural suffix predictors for PPM. By combining data-driven learning with formal background knowledge, our approach achieves improved prediction accuracy and higher compliance with logical constraints, even under noisy conditions. The proposed logical loss formulations, offering both local and global perspectives, demonstrate the effectiveness and generality of the method across different real-world datasets.

Future work will focus on extending the logical loss framework to support additional types of constraints that capture diverse process dimensions such as resources, numeric attributes, and event timestamps. We also aim to assess the method in the presence of concept drift, where process behavior evolves over time. Finally, further investigation into the synergy between local and global constraints, as well as their integration with Large Language Models,

could be promising for advancing the state-of-the-art in multi-step symbolic sequence generation.

Acknowledgments

The work of Fabio Patrizi, Elena Umili and Matteo Mancanelli was supported by the PNRR MUR project PE0000013-FAIR. This study was funded by the European Union - NextGenerationEU, in the framework of the iN-EST - Interconnected Nord-Est Innovation Ecosystem (iNEST ECS00000043 – CUP I43C22000250006). The views and opinions expressed are solely those of the authors and do not necessarily reflect those of the European Union, nor can the European Union be held responsible for them.

References

- [1] E. Rama-Maneiro, J. C. Vidal, M. Lama, Deep learning for predictive business process monitoring: Review and benchmark, *IEEE Transactions on Services Computing* 16 (2020) 739–756.
- [2] N. Tax, I. Verenich, M. L. Rosa, M. Dumas, Predictive business process monitoring with LSTM neural networks, in: *Advanced Information Systems Engineering - 29th International Conference, CAiSE 2017, Essen, Germany, June 12-16, 2017, Proceedings, 2017*, pp. 477–492. doi:10.1007/978-3-319-59536-8_30. URL https://doi.org/10.1007/978-3-319-59536-8_30
- [3] G. Rivera Lazo, R. Nanculef, Multi-attribute transformers for sequence prediction in business process management, in: *Discovery Science: 25th International Conference, DS 2022, Montpellier, France, October 10–12, 2022, Proceedings, Springer-Verlag, Berlin, Heidelberg, 2022*, p. 184–194. doi:10.1007/978-3-031-18840-4_14.
- [4] E. Rama-Maneiro, F. Patrizi, J. C. Vidal, M. Lama, Towards learning the optimal sampling strategy for suffix prediction in predictive monitoring, in: *Proc. of CAISE 2024, 2024*, p. To Appear.
- [5] E. Giunchiglia, M. C. Stoian, S. Khan, F. Cuzzolin, T. Lukasiewicz, ROAD-R: the autonomous driving dataset with logical requirements, *Mach. Learn.* 112 (9) (2023) 3261–3291. doi:10.1007/S10994-023-06322-Z.

- [6] M. Ma, J. Gao, L. Feng, J. Stankovic, Stlnet: Signal temporal logic enforced multivariate recurrent neural networks, in: H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, H. Lin (Eds.), *Advances in Neural Information Processing Systems*, Vol. 33, Curran Associates, Inc., 2020, pp. 14604–14614.
- [7] C. Di Francescomarino, F. M. Maggi, G. Petrucci, A. Yeshchenko, An eye into the future: Leveraging a-priori knowledge in predictive business process monitoring, in: *Business Process Management - 15th International Conference, BPM 2017, Barcelona, Spain, September 10-15, 2017, Proceedings*, 2017, pp. 252–268. doi:10.1007/978-3-319-65000-5_15.
- [8] V. Collura, K. Tit, L. Bussi, E. Giunchiglia, M. Cordy, TRIDENT: temporally restricted inference via dfa-enhanced neural traversal, *CoRR* abs/2506.09701 (2025). arXiv:2506.09701, doi:10.48550/ARXIV.2506.09701.
URL <https://doi.org/10.48550/arXiv.2506.09701>
- [9] X. Lu, P. West, R. Zellers, R. Le Bras, C. Bhagavatula, Y. Choi, NeuroLogic decoding: (un)supervised neural text generation with predicate logic constraints, in: K. Toutanova, A. Rumshisky, L. Zettlemoyer, D. Hakkani-Tur, I. Beltagy, S. Bethard, R. Cotterell, T. Chakraborty, Y. Zhou (Eds.), *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Association for Computational Linguistics, Online, 2021, pp. 4288–4299. doi:10.18653/v1/2021.naacl-main.339.
URL <https://aclanthology.org/2021.naacl-main.339/>
- [10] X. Lu, S. Welleck, P. West, L. Jiang, J. Kasai, D. Khashabi, R. Le Bras, L. Qin, Y. Yu, R. Zellers, N. A. Smith, Y. Choi, NeuroLogic a*esque decoding: Constrained text generation with lookahead heuristics, in: M. Carpuat, M.-C. de Marneffe, I. V. Meza Ruiz (Eds.), *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Association for Computational Linguistics, Seattle, United States, 2022, pp. 780–799. doi:10.18653/v1/2022.naacl-main.57.
URL <https://aclanthology.org/2022.naacl-main.57/>

- [11] B. Krause, A. D. Gotmare, B. McCann, N. S. Keskar, S. Joty, R. Socher, N. F. Rajani, GeDi: Generative discriminator guided sequence generation, in: M.-F. Moens, X. Huang, L. Specia, S. W.-t. Yih (Eds.), Findings of the Association for Computational Linguistics: EMNLP 2021, Association for Computational Linguistics, Punta Cana, Dominican Republic, 2021, pp. 4929–4952. doi:10.18653/v1/2021.findings-emnlp.424.
URL <https://aclanthology.org/2021.findings-emnlp.424/>
- [12] H. Zhang, P.-N. Kung, M. Yoshida, G. V. den Broeck, N. Peng, Adaptable logical control for large language models, in: The Thirty-eighth Annual Conference on Neural Information Processing Systems, 2024.
URL <https://openreview.net/forum?id=58X9v92zRd>
- [13] N. Miao, H. Zhou, L. Mou, R. Yan, L. Li, Cgmh: constrained sentence generation by metropolis-hastings sampling, in: Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, AAAI’19/IAAI’19/EAAI’19, AAAI Press, 2019. doi:10.1609/aaai.v33i01.33016834.
URL <https://doi.org/10.1609/aaai.v33i01.33016834>
- [14] J. Loula, B. LeBrun, L. Du, B. Lipkin, C. Pasti, G. Grand, T. Liu, Y. Emara, M. Freedman, J. Eisner, R. Cotterell, V. Mansinghka, A. K. Lew, T. Vieira, T. J. O’Donnell, Syntactic and semantic control of large language models via sequential monte carlo, in: The Thirteenth International Conference on Learning Representations, 2025.
URL <https://openreview.net/forum?id=xoXn62FzD0>
- [15] E. Jang, S. Gu, B. Poole, Categorical reparameterization with gumbel-softmax, in: 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings, 2017.
- [16] F. J. Damerau, A technique for computer detection and correction of spelling errors, Commun. ACM 7 (3) (1964) 171–176. doi:10.1145/363958.363994.

- [17] E. Umili, G. P. Licks, F. Patrizi, Enhancing deep sequence generation with logical temporal knowledge, in: Proceedings of the 3rd International Workshop on Process Management in the AI Era (PMAI 2024) co-located with 27th European Conference on Artificial Intelligence (ECAI 2024), Santiago de Compostela, Spain, October 19, 2024, 2024, pp. 23–34.
URL <https://ceur-ws.org/Vol-3779/paper4.pdf>
- [18] A. Pnueli, The temporal logic of programs, in: 18th Annual Symposium on Foundations of Computer Science, Providence, Rhode Island, USA, 31 October - 1 November 1977, IEEE Computer Society, 1977, pp. 46–57. doi:10.1109/SFCS.1977.32.
URL <https://doi.org/10.1109/SFCS.1977.32>
- [19] G. De Giacomo, M. Y. Vardi, Linear temporal logic and linear dynamic logic on finite traces, in: Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence, IJCAI '13, AAAI Press, 2013, p. 854–860.
- [20] G. De Giacomo, R. De Masellis, M. Montali, Reasoning on ltl on finite traces: Insensitivity to infiniteness, Proceedings of the AAAI Conference on Artificial Intelligence 28 (1) (Jun. 2014). doi:10.1609/aaai.v28i1.8872.
URL <https://ojs.aaai.org/index.php/AAAI/article/view/8872>
- [21] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, G. Lample, Llama: Open and efficient foundation language models, CoRR abs/2302.13971 (2023). arXiv:2302.13971, doi:10.48550/ARXIV.2302.13971.
URL <https://doi.org/10.48550/arXiv.2302.13971>
- [22] OpenAI, Gpt-4 technical report (2024). arXiv:2303.08774.
- [23] A. van den Oord, N. Kalchbrenner, K. Kavukcuoglu, Pixel recurrent neural networks, in: M. F. Balcan, K. Q. Weinberger (Eds.), Proceedings of The 33rd International Conference on Machine Learning, Vol. 48 of Proceedings of Machine Learning Research, PMLR, New York, New York, USA, 2016, pp. 1747–1756.

- [24] T. Salimans, A. Karpathy, X. Chen, D. P. Kingma, Pixelcnn++: A pixelcnn implementation with discretized logistic mixture likelihood and other modifications, in: ICLR, 2017.
- [25] A. Casolaro, V. Capone, G. Iannuzzo, F. Camastra, Deep learning for time series forecasting: Advances and open problems, *Information* 14 (11) (2023). doi:10.3390/info14110598.
URL <https://www.mdpi.com/2078-2489/14/11/598>
- [26] E. Umili, R. Capobianco, Deepdfa: Automata learning through neural probabilistic relaxations, in: ECAI 2024 - 27th European Conference on Artificial Intelligence, 19-24 October 2024, Santiago de Compostela, Spain - Including 13th Conference on Prestigious Applications of Intelligent Systems (PAIS 2024), 2024, pp. 1051–1058. doi:10.3233/FAIA240596.
URL <https://doi.org/10.3233/FAIA240596>
- [27] C. J. Maddison, A. Mnih, Y. W. Teh, The concrete distribution: A continuous relaxation of discrete random variables, in: 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings, 2017.
URL <https://openreview.net/forum?id=S1jE5L5gl>
- [28] F. Fuggitti, Ltlf2dfa (March 2019). doi:10.5281/zenodo.3888410.
- [29] E. Umili, R. Capobianco, G. D. Giacomo, Grounding ltlf specifications in image sequences, in: Proceedings of the 20th International Conference on Principles of Knowledge Representation and Reasoning, KR 2023, Rhodes, Greece, September 2-8, 2023, 2023, pp. 668–678. doi:10.24963/KR.2023/65.
URL <https://doi.org/10.24963/kr.2023/65>
- [30] N. Manginas, G. Paliouras, L. D. Raedt, Nesya: Neurosymbolic automata, *CoRR* abs/2412.07331 (2024). arXiv:2412.07331, doi:10.48550/ARXIV.2412.07331.
URL <https://doi.org/10.48550/arXiv.2412.07331>
- [31] I. Donadello, P. Felli, C. Innes, F. M. Maggi, M. Montali, Conformance checking of fuzzy logs against declarative temporal specifications, in: Business Process Management - 22nd International Conference, BPM

- 2024, Krakow, Poland, September 1-6, 2024, Proceedings, 2024, pp. 39–56. doi:10.1007/978-3-031-70396-6_3.
URL https://doi.org/10.1007/978-3-031-70396-6_3
- [32] S. Zhu, L. M. Tabajara, J. Li, G. Pu, M. Y. Vardi, Symbolic ltl synthesis, in: Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17, 2017, pp. 1362–1369. doi:10.24963/ijcai.2017/189.
URL <https://doi.org/10.24963/ijcai.2017/189>
- [33] S. Bansal, Y. Li, L. M. Tabajara, M. Y. Vardi, Hybrid compositional reasoning for reactive synthesis from finite-horizon specifications, in: The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020, AAAI Press, 2020, pp. 9766–9774.
URL <https://aaai.org/ojs/index.php/AAAI/article/view/6528>
- [34] G. D. Giacomo, M. Favorito, Compositional approach to translate ltl/ldf into deterministic finite automata, in: S. Biundo, M. Do, R. Goldman, M. Katz, Q. Yang, H. H. Zhuo (Eds.), Proceedings of the Thirty-First International Conference on Automated Planning and Scheduling, ICAPS 2021, Guangzhou, China (virtual), August 2-13, 2021, AAAI Press, 2021, pp. 122–130.
URL <https://ojs.aaai.org/index.php/ICAPS/article/view/15954>
- [35] M. Pesic, W. M. van der Aalst, A declarative approach for flexible business processes management, in: Business Process Management Workshops, 2006.
- [36] M. Pesic, H. Schonenberg, W. M. van der Aalst, Declare: Full support for loosely-structured processes, in: 11th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2007), 2007, pp. 287–287. doi:10.1109/EDOC.2007.14.
- [37] M. Westergaard, Better algorithms for analyzing and enacting declarative workflow languages using ltl, in: S. Rinderle-Ma, F. Toumani,

- K. Wolf (Eds.), Business Process Management, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 83–98.
- [38] A. Alman, C. D. Ciccio, D. Haas, F. M. Maggi, A. Nolte, Rule mining with rum, in: ICPM, IEEE, 2020, pp. 121–128.
 - [39] N. Tax, I. Verenich, M. L. Rosa, M. Dumas, Predictive business process monitoring with LSTM neural networks, in: Advanced Information Systems Engineering - 29th International Conference, CAiSE 2017, Essen, Germany, June 12-16, 2017, Proceedings, 2017, pp. 477–492. doi:10.1007/978-3-319-59536-8_30.
 - [40] J. Evermann, J. Rehse, P. Fettke, Predicting process behaviour using deep learning, Decis. Support Syst. 100 (2017) 129–140. doi:10.1016/J.DSS.2017.04.003.
URL <https://doi.org/10.1016/j.dss.2017.04.003>
 - [41] I. Ketykó, F. Mannhardt, M. Hassani, B. F. van Dongen, What averages do not tell - predicting real life processes with sequential deep learning, CoRR abs/2110.10225 (2021). arXiv:2110.10225.
URL <https://arxiv.org/abs/2110.10225>
 - [42] N. D. Mauro, A. Appice, T. M. A. Basile, Activity prediction of business process instances with inception CNN models, in: AI*IA 2019 - Advances in Artificial Intelligence - XVIIIth International Conference of the Italian Association for Artificial Intelligence, Rende, Italy, November 19-22, 2019, Proceedings, 2019, pp. 348–361. doi:10.1007/978-3-030-35166-3_25.
URL https://doi.org/10.1007/978-3-030-35166-3_25
 - [43] F. Taymouri, M. L. Rosa, S. M. Erfani, A deep adversarial model for suffix and remaining time prediction of event sequences, in: Proceedings of the 2021 SIAM International Conference on Data Mining, SDM 2021, Virtual Event, April 29 - May 1, 2021, 2021, pp. 522–530. doi:10.1137/1.9781611976700.59.
URL <https://doi.org/10.1137/1.9781611976700.59>
 - [44] A. Chiorrini, C. Diamantini, A. Mircoli, D. Potena, A preliminary study on the application of reinforcement learning for predictive process monitoring, in: Process Mining Workshops - ICPM 2020 International Work-

shops, Padua, Italy, October 5-8, 2020, Revised Selected Papers, 2020, pp. 124–135. doi:10.1007/978-3-030-72693-5_10.

- [45] T. R. Besold, A. S. d’Avila Garcez, S. Bader, H. Bowman, P. M. Domingos, P. Hitzler, K. Kühnberger, L. C. Lamb, P. M. V. Lima, L. de Penning, G. Pinkas, H. Poon, G. Zaverucha, Neural-symbolic learning and reasoning: A survey and interpretation, in: *Neuro-Symbolic Artificial Intelligence: The State of the Art*, 2021, pp. 1–51. doi:10.3233/FAIA210348.
- [46] Y. Xie, F. Zhou, H. Soh, Embedding symbolic temporal knowledge into deep sequential models, *CoRR* abs/2101.11981 (2021). arXiv:2101.11981.
URL <https://arxiv.org/abs/2101.11981>
- [47] S. Badreddine, A. d’Avila Garcez, L. Serafini, M. Spranger, Logic tensor networks, *Artificial Intelligence* 303 (2022) 103649. doi:<https://doi.org/10.1016/j.artint.2021.103649>.
URL <https://www.sciencedirect.com/science/article/pii/S0004370221002009>
- [48] E. Umili, F. Argenziano, A. Barbin, R. Capobianco, Visual reward machines, in: *Proceedings of the 17th International Workshop on Neural-Symbolic Learning and Reasoning*, La Certosa di Pontignano, Siena, Italy, July 3-5, 2023, 2023, pp. 255–267.
URL <https://ceur-ws.org/Vol-3432/paper23.pdf>
- [49] E. Umili, R. Capobianco, DeepDFA: a transparent neural network design for dfa induction (2023). doi:10.13140/RG.2.2.25449.98401.