

# Neuro-symbolic Prescriptive Process Analytics: Learning Compliant and Effective Process Recommendations through Constraint Distillation

Gyunam Park

Eindhoven University of Technology  
Groene Loper 5, 5612 AZ Eindhoven, The Netherlands  
`g.park@tue.nl`

**Abstract.** Given a running process case, prescriptive process analytics (PrPA) recommends the next activity that best improves a desired outcome, such as reducing cycle time. Recommendations must also be operationally feasible: an activity that improves the predicted outcome but violates process constraints cannot be enacted. The core challenge is therefore not compliance alone, which inference-time filtering can enforce, but achieving high outcome quality *within* the compliant space. Existing approaches fall short: post-hoc filtering never learns which compliant activities are outcome-improving, while logic-as-loss penalties balance unstable competing gradients from data fit and compliance. Both also treat all constraints equally, so combining mandatory hard constraints with relaxable soft constraints can leave many cases with no feasible recommendation. We propose **NeSyPA**, a neurosymbolic PrPA method with two contributions: *Hierarchical Constraint Relaxation*, which drops soft constraints only when the full constraint set is unsatisfiable, and *Constraint-Compiled Outcome Distillation*, which compiles per-prefix feasible sets into feasibility-masked, outcome-weighted soft targets that teach the neural prescriptor to rank feasible activities by outcome merit. On four real-world event logs, NeSyPA achieves the highest Effective Compliance Rate on all test sets, with gains of +6.7% to +9.9%, driven by higher outcome quality within the compliant space.

**Keywords:** Prescriptive process analytics · Neurosymbolic AI · Constraint compliance · Knowledge distillation

## 1 Introduction

Given a running process case, prescriptive process analytics (PrPA) recommends the next activity that best improves a desired outcome, for example reducing cycle time or avoiding an SLA breach [22]. Unlike predictive monitoring, which forecasts what *will* happen, PrPA must recommend what *should* happen under process constraints that restrict the set of permissible activities. A recommendation to “skip the compliance check” may optimize predicted cycle time but is operationally infeasible. The core challenge is therefore not compliance alone (inference-time filtering can enforce that) but achieving high outcome improvement *within* the compliant space, so that recommendations are both feasible and effective.

Current PrPA approaches handle constraints in ways that sacrifice outcome quality. Post-hoc filtering trains an unconstrained model and discards infeasible recommendations at inference time. While this can maintain compliance, the model never learns *which compliant activities are outcome-improving*: the best recommendation is often discarded, and the fallback may be weak, since no learning signal flows back from the constraint structure. Penalty-based methods, sometimes called logic-as-loss approaches, add a constraint violation penalty to the training loss [13]. This forces the model to balance competing gradients from data fit and logic fit: the gradient tells the model *that* it is wrong but not *how* to fix it, and tuning the penalty weight  $\lambda$  is inherently unstable, often degrading both compliance and outcome quality.

Beyond these mechanism-level limitations, a fundamental practical challenge remains unaddressed. Real-world process constraint knowledge bases combine rules that operate at different levels of authority. Some constraints are *hard*: they derive from external regulations or domain expertise and must never be violated. Others are *soft*: they capture frequent behavioral patterns, e.g., discovered from historical data and reflect statistical regularities that may not hold for every case. When hard and soft constraints are enforced equally, they can conflict, causing the feasible set, i.e., the set of activities satisfying all constraints simultaneously, to become empty. In our experiments, the full constraint knowledge base renders 5.9% to 35.1% of prefixes irreparable across datasets. No existing method distinguishes constraint priority or handles these over-constrained states gracefully.

We propose NeSyPA, a neurosymbolic method for prescriptive process analytics that addresses both challenges through two contributions. First, *Hierarchical Constraint Relaxation* (HCR) partitions the constraint knowledge base into hard constraints that are never relaxed and soft constraints that may be dropped when the full set is unsatisfiable, reducing irreparable rates dramatically while preserving hard constraint compliance. Second, *Constraint-Compiled Outcome Distillation* (CCOD) pre-computes the hierarchical feasible set for each training prefix, constructs feasibility-masked soft targets weighted by predicted outcome improvement, and fine-tunes the neural prescriptor on these targets in a single phase. Unlike logic-as-loss methods, CCOD has no  $\lambda$  tradeoff parameter: the targets are zero outside the feasible set and concentrate probability mass on the compliant activities with the highest outcome lift. This teaches the model to optimize outcomes directly within the compliant space, rather than learning outcome improvement and compliance as competing objectives.

The method can be viewed as a form of knowledge distillation [11] where the teacher is a symbolic constraint-outcome oracle composed of deterministic finite automata and an outcome evaluator, and the student is the neural prescriptor that learns to approximate the oracle’s feasibility-masked, outcome-weighted distributions. The key difference from standard distillation is that the teacher is not a larger neural model but a symbolic reasoner that computes exact constraint satisfaction.

We evaluate NeSyPA on four real-world event logs from two domains, comparing against four baselines spanning unconstrained, post-hoc filtering, reinforcement learning, and logic-as-loss approaches.<sup>1</sup> NeSyPA achieves the highest Effective

<sup>1</sup> Code and data are available at [https://github.com/gyunamister/nesy\\_prpa](https://github.com/gyunamister/nesy_prpa).

Compliance Rate (ECR), the fraction of recommendations that are both outcome-improving and constraint-compliant, on all four test sets, driven by substantially higher outcome improvement while maintaining comparable constraint satisfaction. An ablation study confirms that CCOD is the dominant contributor, while a sensitivity analysis reveals the complementary roles of hard and soft constraints.

The remainder of this paper is organized as follows. Section 2 reviews related work. Section 3 introduces preliminaries. Section 4 presents the NeSyPA method. Section 5 describes the experimental design. Section 6 presents results and discussion. Section 7 discusses threats to validity. Section 8 concludes.

## 2 Related Work

Prescriptive process analytics has been approached through several distinct paradigms, each with characteristic strengths and limitations regarding constraint-aware recommendation.

*Alarm-Based PrPA.* Alarm-based approaches frame PrPA as a binary decision: whether or not to intervene in a running case [20,8]. These methods train classifiers to predict undesirable outcomes and trigger alarms when the predicted probability exceeds a threshold, potentially subject to resource constraints. While effective for intervention timing, alarm-based methods decide *whether* to act, not *which* activity to recommend. They are fundamentally complementary to, rather than competitive with, activity-level prescriptive methods, and they incorporate no notion of process constraint compliance.

*Causal Inference PrPA.* Causal approaches estimate treatment effects of predefined interventions and decide on whom and when to apply them [2,3,18,19]. The treatment set is fixed by domain experts, and the methods focus on identifying cases where intervention would be most beneficial. Since treatments are predefined rather than discovered, process constraints are implicitly satisfied by construction of the treatment set. However, this secures feasibility only by restricting recommendations to a fixed, pre-enumerated set of expert-defined interventions, which limits the method to domains where the intervention space can be fully enumerated in advance and provides no mechanism for adapting to new or complex constraint structures.

*Reinforcement Learning PrPA.* Reinforcement learning approaches model PrPA as sequential decision-making, where an agent learns a policy that maps process states to activity recommendations [1,4]. Constraint handling in these methods is either post-hoc, filtering infeasible actions from the policy output, or penalty-based, incorporating constraint violations as negative rewards. Neither mechanism provides a constructive approach to producing the nearest feasible alternative when the top recommendation is infeasible. Furthermore, all constraints are treated equally, with no distinction between hard constraints (mandatory regulatory requirements) and soft constraints (relaxable organizational patterns), leaving no principled way to handle over-constrained states.

*Neurosymbolic Predictive Monitoring.* Recent work has explored neurosymbolic integration for process monitoring, though primarily in the predictive rather than prescriptive setting. This emerging line spans compliance-aware predictive monitoring [15], next-activity prediction via two-stage logic tensor networks with rule pruning [17], process anomaly detection [9], and domain-specific applications such as real-time sepsis mortality prediction [16]. These efforts confirm the value of injecting symbolic knowledge into neural process models, but they address predictive or diagnostic tasks rather than constraint-aware activity recommendation. Mezini et al. [13] compile  $LTL_f$  constraints into a differentiable loss function for autoregressive next-activity prediction. This is the closest neurosymbolic approach to ours, but it differs in several important ways: it uses a logic-as-loss formulation that creates a  $\lambda$  tradeoff between data fit and constraint compliance; it is purely predictive rather than prescriptive; it treats all constraints equally without hierarchical prioritization; and it cannot constructively generate compliant alternatives when the top prediction violates constraints.

In contrast to all of the above, our approach pre-computes the feasible activity set exactly via deterministic finite automata [10] and distills it into soft training targets, so compliance is enforced by construction and outcome quality is optimized within the compliant space, with no  $\lambda$  tradeoff. To our knowledge, no existing method distinguishes hard and soft constraint priority to handle irreparable states from over-constrained knowledge bases, or compiles constraint feasibility into training targets as soft outcome-weighted distributions.

### 3 Preliminaries

*Event Logs, Traces, and Prefixes.* An event log  $L$  consists of a set of traces, where each trace  $\sigma = \langle e_1, e_2, \dots, e_n \rangle$  records the sequence of events for a single process case. Each event  $e_i = (a_i, t_i, r_i, \mathbf{d}_i)$  comprises an activity label  $a_i \in \mathcal{A}$ , a timestamp  $t_i$ , a resource  $r_i$ , and a data payload  $\mathbf{d}_i$ . A prefix  $\sigma_k = \langle e_1, \dots, e_k \rangle$  is the subsequence of the first  $k$  events of a trace. The activity alphabet  $\mathcal{A}$  denotes the set of all possible activity labels in the log.

*Declare Constraints and Automata.* Declare [14] is a declarative process modeling language that specifies constraints as Linear Temporal Logic over finite traces ( $LTL_f$ ) formulas. Key templates used in this work include **existence** ( $A$  must occur at least once), **response** (if  $A$  occurs,  $B$  must eventually follow), **precedence** ( $B$  can only occur if  $A$  has occurred before), **not-co-existence** ( $A$  and  $B$  cannot both occur in the same trace), **alternate-response** (each occurrence of  $A$  must be followed by  $B$  before  $A$  can occur again), and **chain-precedence** ( $B$  can occur only if  $A$  occurred immediately before). Each Declare constraint can be compiled into a deterministic finite automaton (DFA)  $\mathcal{M}_j$  [10] with states classified as accepting ( $F$ ), pending, or permanently violated ( $V$ ). Given a prefix  $\sigma_k$ , the automaton tracks constraint status incrementally: appending activity  $a$  to  $\sigma_k$  transitions the automaton from state  $q_k^j$  to successor state  $q_{k+1}^j = \delta_j(q_k^j, a)$ , where  $\delta_j$  is the transition function of  $\mathcal{M}_j$ .

*Data-Aware Constraints.* Data-aware constraints take the form  $\phi(\mathbf{d}) \Rightarrow \psi$ , where  $\phi$  is a predicate over case or event attributes and  $\psi$  is a Declare constraint. For example, “if `loan_amount` > 10K then `existence(manual_review)`.” When the condition  $\phi(\mathbf{d}_k)$  evaluates to false for the current case, the constraint is vacuously satisfied.

*Constraint Hierarchy.* The constraint knowledge base  $\Phi$  is partitioned into two tiers based on their enforcement level: *hard constraints*  $\Phi_{\text{hard}}$ , which must never be violated and are never relaxed; and *soft constraints*  $\Phi_{\text{soft}}$ , which may be relaxed when they create infeasible states. In practice, hard constraints typically derive from external regulations and domain expertise (e.g., the EU Consumer Credit Directive mandating specific review steps), while soft constraints capture statistically frequent behavioral patterns that may not hold universally, e.g., discovered from training traces via Declare mining [5]. Because these two tiers originate from different authorities and operate at different levels of generality, conflicts between them are common: a soft constraint may prohibit an activity that a hard constraint requires, leaving no activity that satisfies both simultaneously. The full knowledge base is  $\Phi = \Phi_{\text{hard}} \cup \Phi_{\text{soft}}$ . The *hierarchical feasible set* for a prefix  $\sigma_k$  resolves such conflicts by prioritizing hard constraints:

$$\mathcal{F}_{\text{hier}}(\sigma_k) = \begin{cases} \mathcal{F}_{\text{full}}(\sigma_k) & \text{if } \mathcal{F}_{\text{full}}(\sigma_k) \neq \emptyset \\ \mathcal{F}_{\text{hard}}(\sigma_k) & \text{otherwise} \end{cases} \quad (1)$$

where  $\text{Compliant}_{\Psi}(\sigma_k, a)$  holds iff appending activity  $a$  to prefix  $\sigma_k$  does not violate any constraint in  $\Psi$ ,  $\mathcal{F}_{\text{full}}(\sigma_k) = \{a \in \mathcal{A} \mid \text{Compliant}_{\Phi}(\sigma_k, a)\}$ , and  $\mathcal{F}_{\text{hard}}(\sigma_k) = \{a \in \mathcal{A} \mid \text{Compliant}_{\Phi_{\text{hard}}}(\sigma_k, a)\}$ .

*Knowledge Distillation.* Knowledge distillation [11] transfers knowledge from a teacher to a student via soft probability distributions. Given a teacher distribution  $q$  over classes, the student is trained to minimize the cross-entropy  $H(q, p)$  where  $p$  is the student’s output distribution. Soft targets encode richer information than hard labels: they convey relative preferences among classes, not just the top-1 choice.

## 4 The NeSyPA Method

### 4.1 Problem Statement

Given a prefix  $\sigma_k$  of a running case, the goal is to recommend the next activity  $a^* \in \mathcal{A}$  that maximizes estimated outcome improvement subject to hierarchical constraint compliance:

$$a^* = \arg \max_{a \in \mathcal{F}_{\text{hier}}(\sigma_k)} \text{Score}(\sigma_k, a) \quad (2)$$

The method operates in three phases: (1) warm-up training of the neural prescriptor and compilation of constraints into DFA automata, (2) constraint-compiled outcome distillation that constructs feasibility-masked targets and fine-tunes the prescriptor, and (3) hierarchical inference that generates candidates and applies constraint filtering. Figure 1 provides an overview.

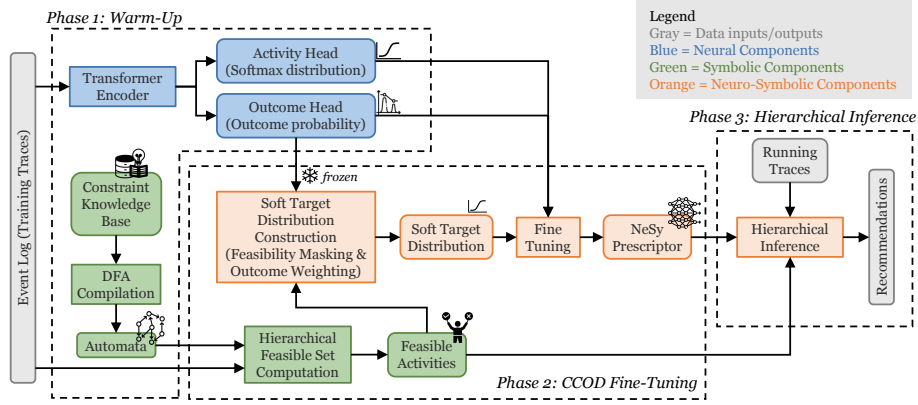


Fig. 1: Overview of the NeSyPA method. Phase 1 (Warm-Up) trains the neural prescriptor and compiles constraints into DFA automata. Phase 2 (CCOD Fine-Tuning) constructs feasibility-masked, outcome-weighted soft targets and fine-tunes the prescriptor. Phase 3 (Hierarchical Inference) generates candidates and applies hierarchical constraint filtering.

## 4.2 Phase 1: Warm-Up Training

*Prescriptor Architecture.* The neural prescriptor takes as input a prefix  $\sigma_k$  and produces both an activity distribution and an outcome prediction. Each event  $e_i$  is encoded as the concatenation of an activity embedding, a resource embedding, an MLP-encoded data payload, and a temporal encoding based on inter-event duration. The sequence of event encodings is processed by a Transformer encoder [21] to produce a context vector  $\mathbf{c}_k$ .

Two output heads operate on  $\mathbf{c}_k$ . The activity head produces a softmax distribution  $\mathbf{p}_k \in \mathbb{R}^{|A|}$  over all activities. The outcome head produces a predicted positive-outcome probability  $\hat{y}_k \in [0, 1]$ . The prescriptive score for a candidate activity  $a$  combines both signals:

$$\text{Score}(a | \sigma_k) = \alpha \cdot p_k(a) + (1 - \alpha) \cdot \Delta \hat{y}_k(a) \quad (3)$$

where  $\Delta \hat{y}_k(a) = \hat{y}(\sigma_k \cdot a) - \hat{y}(\sigma_k)$  is the estimated outcome lift from appending  $a$ , and  $\alpha \in [0, 1]$  weights process likelihood against outcome improvement.

*Warm-Up Loss.* The model is trained on a supervised loss combining next-activity prediction (cross-entropy) and outcome prediction (binary cross-entropy):

$$\mathcal{L}_{\text{data}} = \mathcal{L}_{\text{CE}}(a_{k+1}, \mathbf{p}_k) + \beta \cdot \mathcal{L}_{\text{BCE}}(y, \hat{y}_k) \quad (4)$$

where  $a_{k+1}$  is the ground-truth next activity,  $y$  is the binary outcome label, and  $\beta$  balances the two loss terms.

*Constraint Compilation.* Each constraint  $\phi_j \in \Phi$  is compiled to a DFA  $\mathcal{M}_j$ . Given a prefix  $\sigma_k$ , the automaton is in state  $q_k^j$ . Appending a candidate activity  $a$  yields the successor state  $q_{k+1}^j = \delta_j(q_k^j, a)$ , from which the compliance status  $\text{Status}_j(\sigma_k, a)$  is determined; specifically,  $\text{Status}_j(\sigma_k, a) = \text{violated}$  if  $q_{k+1}^j$  is a permanently violated state of  $\mathcal{M}_j$ . An activity is compliant with the full knowledge base if and only if no constraint enters a permanently violated state:

$$\text{Compliant}_\Phi(\sigma_k, a) = \bigwedge_j \text{Status}_j(\sigma_k, a) \neq \text{violated} \quad (5)$$

For data-aware constraints, the condition predicate  $\phi(\mathbf{d}_k)$  is evaluated first; if the condition is false, the constraint is vacuously satisfied and does not restrict the feasible set.

### 4.3 Phase 2: Constraint-Compiled Outcome Distillation

CCOD constructs feasibility-masked, outcome-weighted soft training targets and fine-tunes the prescriptor. For each training prefix  $\sigma_k$ :

*Step 1: Compute hierarchical feasible set.* The constraint reasoner computes feasibility in three stages using the compiled DFA automata. First, it computes  $\mathcal{F}_{\text{full}}(\sigma_k)$  by evaluating all activities against all automata. If  $\mathcal{F}_{\text{full}}(\sigma_k) \neq \emptyset$ , this set is used and all constraints are satisfied. If  $\mathcal{F}_{\text{full}}(\sigma_k) = \emptyset$ , the reasoner computes  $\mathcal{F}_{\text{hard}}(\sigma_k)$  by evaluating activities against only hard constraint automata, relaxing all soft constraints as a group. If  $\mathcal{F}_{\text{hard}}(\sigma_k) = \emptyset$ , the prefix is flagged as irreparable and excluded from CCOD training. Hard constraints are never relaxed; soft constraints are relaxed as a group: either all are enforced or all are dropped. We adopt this binary scheme for simplicity; selective per-constraint relaxation is feasible but introduces a combinatorial constraint selection problem that is orthogonal to our core contribution. We discuss this trade-off and a path toward selective relaxation in Sect. 6.4.

*Step 2: Compute outcome lift.* For each activity  $a \in \mathcal{F}_{\text{hier}}(\sigma_k)$ , the estimated outcome lift is computed as  $\Delta\hat{y}(a) = \hat{y}_{\text{frozen}}(\sigma_k \cdot a) - \hat{y}_{\text{frozen}}(\sigma_k)$  using a frozen copy of the warm-up outcome head. The frozen copy ensures stable targets throughout fine-tuning, preventing a feedback loop where changing outcome predictions shift the targets that drive further training.

*Step 3: Construct soft target distribution.* The target distribution concentrates probability mass on feasible activities proportionally to their outcome lift:

$$q(a) = \begin{cases} \frac{\exp(\Delta\hat{y}(a)/\tau)}{\sum_{a' \in \mathcal{F}_{\text{hier}}(\sigma_k)} \exp(\Delta\hat{y}(a')/\tau)} & \text{if } a \in \mathcal{F}_{\text{hier}}(\sigma_k) \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

where  $\tau > 0$  is a temperature parameter. Eq. 6 is a standard softmax over the feasible set, so  $q$  is a valid probability distribution supported only on  $\mathcal{F}_{\text{hier}}(\sigma_k)$  for any  $\tau$ ; the temperature is therefore a design choice that controls target sharpness, not a correctness condition. Low  $\tau$  concentrates mass on the highest-outcome-lift feasible activity; high  $\tau$  spreads mass more uniformly across feasible activities.

*Step 4: Fine-tune.* The prescriptor is fine-tuned by minimizing the cross-entropy between the soft target and the model’s full (unmasked) softmax output, plus the outcome loss on real labels:

$$\mathcal{L}_{\text{CCOD}} = - \sum_{a \in \mathcal{A}} q(a) \log p_k(a) + \beta \cdot \mathcal{L}_{\text{BCE}}(y, \hat{y}_k) \quad (7)$$

Since  $q(a) = 0$  for  $a \notin \mathcal{F}_{\text{hier}}(\sigma_k)$ , the cross-entropy only accumulates terms over feasible activities. Crucially, the gradient  $\partial \mathcal{L} / \partial z_a = p_k(a) - q(a)$  is non-zero for all activities: for infeasible activities where  $q(a) = 0$ , the gradient equals  $p_k(a) > 0$ , which pushes their probabilities down. This teaches the model to internalize constraint compliance, not just ranking within the feasible set but actively avoiding infeasible activities. Only the activity head is fine-tuned on CCOD targets; the outcome head continues to train only on observed outcome labels to prevent label corruption.

CCOD has several properties that distinguish it from prior approaches. Unlike logic-as-loss methods that balance  $\mathcal{L}_{\text{data}} + \lambda \cdot \mathcal{L}_{\text{logic}}$ , CCOD has no compliance-versus-outcome tradeoff parameter. Compliance is enforced by construction, and outcome optimization happens within the compliant space. The soft target encodes a full ranking over feasible activities weighted by outcome merit, providing richer signal than a hard label. The fine-tuning is one-shot and non-iterative: feasible sets and outcome lifts are pre-computed once, making the approach both simple and efficient.

#### 4.4 Phase 3: Hierarchical Inference

At inference time, the prescriptor scores all activities for a given prefix  $\sigma_k$  using Eq. 3 and selects the top- $K$  candidates. These candidates are then passed through the hierarchical constraint reasoner, which computes  $\mathcal{F}_{\text{hier}}(\sigma_k)$  over the candidate set: if  $\mathcal{F}_{\text{full}}(\sigma_k) \neq \emptyset$ , all constraints are enforced; if  $\mathcal{F}_{\text{full}}(\sigma_k) = \emptyset$ , the reasoner relaxes to  $\mathcal{F}_{\text{hard}}(\sigma_k)$ ; if  $\mathcal{F}_{\text{hard}}(\sigma_k) = \emptyset$ , the prefix is flagged as irreparable. The recommended activity is the highest-scoring feasible candidate:  $a^* = \arg\max_{a \in \mathcal{F}_{\text{hier}}(\sigma_k)} \text{Score}(a \mid \sigma_k)$ .

#### 4.5 Computational Cost

Each Declare constraint is compiled once into a small deterministic finite automaton (typically two to five states), so compilation is a negligible one-time cost. For a prefix  $\sigma_k$ , computing the feasible set evaluates every candidate activity against every constraint automaton, which is  $O(|\mathcal{A}| \cdot |\Phi|)$ , where each evaluation is an  $O(1)$  cached state transition. In our datasets  $|\mathcal{A}| \leq 36$  and  $|\Phi|$  is on the order of 10 to 25, giving at most a few hundred automaton lookups per prefix. CCOD adds no online overhead: feasible sets and outcome lifts are pre-computed once before fine-tuning, and fine-tuning itself is one-shot. The resulting feasible sets are small in practice (on average 1.3 to 3.0 activities under the full knowledge base, and 8.5 to 25 under hard constraints only), so candidate scoring at inference is inexpensive. Because the knowledge base is fixed at training time, adapting to changed soft constraints at runtime requires only recompiling the affected automata and re-running this one-time pre-computation, with no change to the model architecture.

Table 1: Dataset overview.

| Dataset       | Domain           | Cases | Act. | Hard constr. | Outcome        |
|---------------|------------------|-------|------|--------------|----------------|
| BPIC 2012     | Loan application | ~13K  | 36   | 9 CF + 1 DA  | Accepted       |
| BPIC 2017     | Loan application | ~31K  | 26   | 8 CF         | Offer accepted |
| Traffic Fines | Fine management  | ~150K | 11   | 10 CF        | Paid on time   |

## 5 Experimental Design

### 5.1 Datasets

We evaluate NeSyPA on four real-world event logs, summarized in Table 1. BPIC 2012 [6] records a loan application process at a Dutch financial institution with 36 distinct activities and approximately 13,000 cases. BPIC 2017 [7] covers a similar loan application process with 26 activities and approximately 31,000 cases; following established practice, we evaluate on two temporal splits (“before” and “after”), yielding distinct test sets. Traffic Fines [12] records an Italian road traffic fine management process with 11 activities and approximately 150,000 cases. The outcome variable is application acceptance for BPIC 2012, offer acceptance for BPIC 2017, and timely fine payment for Traffic Fines.

*Data Preparation.* We split each log temporally by case start time into training (60%), validation (20%), and test (20%) partitions, so that no case spans more than one partition. From every trace we extract all prefixes of length  $k \in [2, n-1]$ , each labeled with its ground-truth next activity, rather than sampling or fixing a single prefix length. This yields 42,587 (BPIC 2012), 86,240 (BPIC 2017 before), 113,896 (BPIC 2017 after), and 52,984 (Traffic Fines) test prefixes.

For each dataset, the constraint knowledge base  $\Phi$  is constructed from two complementary sources. Hard constraints  $\Phi_{\text{hard}}$  encode process-specific business rules that the authors formulated from publicly available process descriptions and the corresponding regulations, rather than through a formal expert-elicitation study: e.g., the mandatory ordering of verification and approval steps in loan applications (BPIC 2012 and 2017), and the legally prescribed notification sequences in fine management (Traffic Fines). We treat these as reliable by assumption; validating them with domain experts is left to future work. Soft constraints  $\Phi_{\text{soft}}$  are discovered from all training traces using frequency-based Declare mining [5] with support  $\geq 0.3$  and confidence  $\geq 0.85$ , representing organizational process patterns. Data-aware constraints combine hard conditional rules with decision-tree-mined rules over event attributes. This hybrid construction ensures that constraints encode knowledge orthogonal to the outcome-optimizing model’s training signal.

Under the full knowledge base, 5.9–35.1% of test prefixes are irreparable (no activity satisfies all constraints simultaneously), confirming that the combination of hard and soft constraints frequently leads to over-constrained states and motivating the need for hierarchical relaxation.

## 5.2 Baselines

We compare NeSyPA against four baselines, all using the same Transformer encoder and the same constraint knowledge base  $\Phi$  for fair comparison. **B1 (Unconstrained)**: The neural prescriptor with no constraint mechanism, producing a raw top-1 recommendation. **B2 (Post-hoc)**: The neural prescriptor with post-hoc hierarchical filtering at inference, establishing a strong filtering baseline that isolates the value of CCOD training beyond filtering alone. **B3 (RL)**: A reinforcement learning prescriptor following the approach described by Agarwal et al. [1], with constraint violations penalized in the reward function using the full knowledge base. **B4 (NeSy-PPM)**: An adaptation of the neurosymbolic predictive monitoring approach by Mezini et al. [13] from the predictive to the prescriptive setting. The original method trains a next-activity predictor with a differentiable LTL<sub>f</sub> logic loss to steer predictions toward constraint-compliant activities. Our adaptation retains this logic-as-loss training (using the full knowledge base  $\Phi$ ) but extends the architecture with an outcome prediction head trained jointly via binary cross-entropy. At inference, candidates are ranked by a prescriptive score  $\alpha \cdot p(a) + (1 - \alpha) \cdot \Delta \hat{y}(a)$ , combining the activity probability  $p(a)$  with the estimated outcome lift  $\Delta \hat{y}(a)$ , and the top candidate is recommended.

## 5.3 Metrics

The primary metric is the *Effective Compliance Rate* (ECR), defined as the fraction of recommendations that are both outcome-improving and constraint-compliant:

$$\text{ECR} = \frac{|\{i : \text{outcome-improving}(i) \wedge \text{compliant}(i)\}|}{|\text{all recommendations}|} \quad (8)$$

ECR captures both dimensions simultaneously: a method with high compliance but poor outcomes, or vice versa, scores low. Supporting metrics include: the Outcome Improvement Rate (OIR), the fraction of recommendations that are outcome-improving regardless of compliance; and the Constraint Satisfaction Rate (CSR), the fraction satisfying all constraints regardless of outcome.

## 5.4 Experimental Protocol

**Experiment 1: Main comparison.** All five methods are evaluated on all four test sets. We report ECR, OIR, and CSR with means and standard deviations over 5 random seeds per method and dataset.

**Experiment 2: Ablation.** Four variants of NeSyPA are compared: the full method, no-CCOD (B2 with hierarchy, isolating CCOD’s contribution), no-hierarchy (full KB only, isolating hierarchy’s contribution), and uniform targets (uniform distribution over  $\mathcal{F}_{\text{hier}}$  instead of outcome-weighted). We report ECR, OIR, and CSR over 5 seeds.

**Experiment 3: Sensitivity to soft constraint quality.** On BPIC 2012, we inject 10%, 20%, and 30% spurious constraints into  $\Phi_{\text{soft}}$  only, since hard constraints

Table 2: Experiment 1: Main results (mean  $\pm$  std over 5 seeds). Best ECR per dataset in **bold**.

| Dataset        | Method           | ECR                             | OIR             | CSR             |
|----------------|------------------|---------------------------------|-----------------|-----------------|
| BPIC 2012      | B1 Unconstrained | .476 $\pm$ .069                 | .711 $\pm$ .054 | .646 $\pm$ .025 |
|                | B2 Post-hoc      | .502 $\pm$ .047                 | .709 $\pm$ .052 | .681 $\pm$ .001 |
|                | B3 RL            | .426 $\pm$ .078                 | .673 $\pm$ .098 | .622 $\pm$ .061 |
|                | B4 NeSy-PPM      | .470 $\pm$ .034                 | .697 $\pm$ .057 | .649 $\pm$ .023 |
|                | <b>NeSyPA</b>    | <b>.569<math>\pm</math>.030</b> | .849 $\pm$ .031 | .677 $\pm$ .004 |
| BPIC 2017 aft. | B1 Unconstrained | .318 $\pm$ .043                 | .505 $\pm$ .061 | .644 $\pm$ .000 |
|                | B2 Post-hoc      | .322 $\pm$ .043                 | .506 $\pm$ .062 | .648 $\pm$ .001 |
|                | B3 RL            | .339 $\pm$ .033                 | .577 $\pm$ .044 | .605 $\pm$ .013 |
|                | B4 NeSy-PPM      | .324 $\pm$ .045                 | .516 $\pm$ .064 | .644 $\pm$ .001 |
|                | <b>NeSyPA</b>    | <b>.437<math>\pm</math>.058</b> | .718 $\pm$ .072 | .648 $\pm$ .001 |
| BPIC 2017 bef. | B1 Unconstrained | .354 $\pm$ .044                 | .547 $\pm$ .052 | .647 $\pm$ .002 |
|                | B2 Post-hoc      | .356 $\pm$ .044                 | .547 $\pm$ .052 | .650 $\pm$ .000 |
|                | B3 RL            | .316 $\pm$ .105                 | .502 $\pm$ .163 | .623 $\pm$ .013 |
|                | B4 NeSy-PPM      | .358 $\pm$ .046                 | .555 $\pm$ .059 | .647 $\pm$ .001 |
|                | <b>NeSyPA</b>    | <b>.456<math>\pm</math>.048</b> | .735 $\pm$ .062 | .650 $\pm$ .001 |
| Traffic Fines  | B1 Unconstrained | .381 $\pm$ .076                 | .532 $\pm$ .132 | .834 $\pm$ .121 |
|                | B2 Post-hoc      | .384 $\pm$ .080                 | .428 $\pm$ .079 | .940 $\pm$ .002 |
|                | B3 RL            | .398 $\pm$ .080                 | .445 $\pm$ .078 | .940 $\pm$ .001 |
|                | B4 NeSy-PPM      | .325 $\pm$ .056                 | .554 $\pm$ .122 | .753 $\pm$ .111 |
|                | <b>NeSyPA</b>    | <b>.399<math>\pm</math>.080</b> | .454 $\pm$ .081 | .941 $\pm$ .000 |

are assumed reliable whereas soft constraints are sensitive to log quality. The model trains and infers with the noisy knowledge base; metrics are evaluated against the original clean knowledge base. We report ECR, OIR, CSR, and the noisy-KB relaxation rate.

*Implementation Details.* Each model is trained with early stopping on the validation split, and metrics are reported as means and standard deviations over 5 random seeds. Hyperparameters were fixed, guided by preliminary runs on the validation split, to the following values: the prescriptive score weight  $\alpha=0.3$ , the outcome loss weight  $\beta=1.0$ , the CCOD temperature  $\tau=0.5$ , and the candidate set size  $K=5$ .

## 6 Results and Discussion

### 6.1 Main Results

Table 2 presents the main comparison across all methods and test sets. NeSyPA achieves the highest ECR on all four test sets, with gains of +6.7pp (BPIC 2012), +9.7pp (BPIC 2017 after), +9.9pp (BPIC 2017 before), and +0.03pp (Traffic Fines) over the respective best baselines.

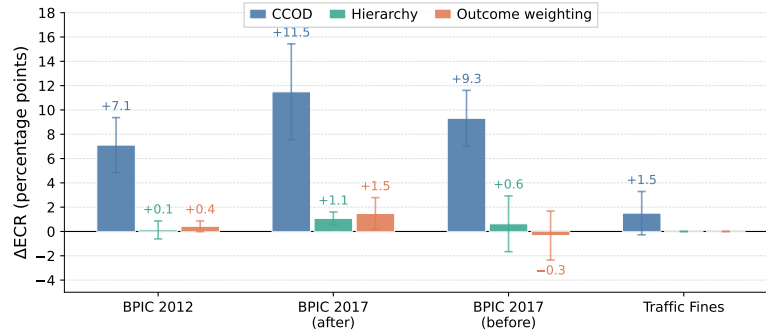


Fig. 2: ECR contribution of each component across datasets. CCOD provides the dominant improvement (+7 to +12pp on BPIC datasets), while hierarchy contributes structurally rather than incrementally.

On the BPIC datasets, the ECR gains are driven by dramatically higher OIR. NeSyPA achieves OIR values of 0.849 (BPIC 2012), 0.718 (BPIC 2017 after), and 0.735 (BPIC 2017 before), compared to 0.505–0.711 for the baselines. CSR remains comparable across methods ( $\sim 0.60$ – $0.68$  on BPIC datasets). On Traffic Fines, OIR differences are smaller (0.454 for NeSyPA vs. 0.428–0.554 for baselines), resulting in a negligible ECR margin of +0.03pp.

B3 (RL) and B4 (NeSy-PPM) perform at or below B1 and B2 on most datasets. B3 shows particularly high variance on BPIC 2017 before (coefficient of variation = 0.33). B4 performs poorly on Traffic Fines (ECR = 0.325), with CSR degrading to 0.753, well below the filtering baselines (0.940).

*Why CSR is below 1.* CSR is computed against the full knowledge base, so it can fall below 1 even though hard-constraint compliance is guaranteed by construction. When the full feasible set is empty, hierarchical relaxation deliberately drops the soft constraints and recommends an activity satisfying only the hard constraints; these recommendations, together with the small fraction of prefixes that remain irreparable, are counted as soft-constraint violations and lower CSR. This is by design, not a failure of enforcement. For the same reason, the marginal CSR difference between B2 and NeSyPA on BPIC 2012 (0.681 vs. 0.677) lies within one standard deviation ( $\pm 0.001$  and  $\pm 0.004$ ): both methods apply the same hierarchical filtering at inference, so the gap reflects only the different top- $K$  candidate sets produced after CCOD training, not a difference in how compliance is enforced.

## 6.2 Ablation

Fig. 2 presents the ablation results across all four test sets. CCOD is the dominant contributor: the full model exceeds the no-CCOD variant (B2 with hierarchy) by +7.1pp (BPIC 2012), +11.5pp (BPIC 2017 after), +10.0pp (BPIC 2017 before), and +1.5pp (Traffic Fines). The gap is largest on BPIC datasets ( $|\mathcal{F}_{\text{full}}| \approx 2$ – $3$  over 26–36 activities) and smallest on Traffic Fines (11 activities, already-high baseline CSR).

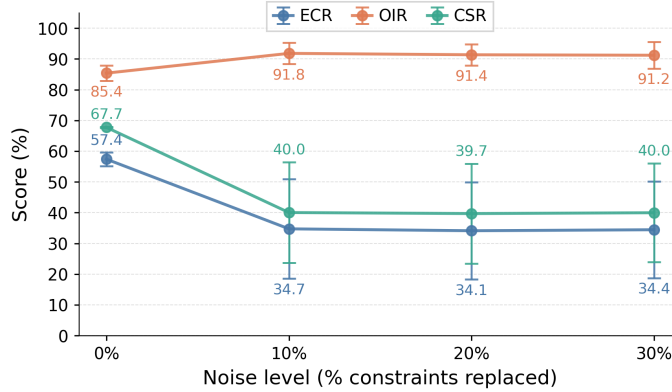


Fig. 3: Sensitivity to soft constraint noise on BPIC 2012. ECR and CSR drop sharply at 10% noise and plateau, while OIR increases as diffuse hard-only CCOD targets make the model more outcome-aggressive. Error bars show standard deviation over 5 seeds.

Hierarchical relaxation provides minimal incremental ECR lift: +0.1pp (BPIC 2012), +1.1pp (BPIC 2017 after), +0.6pp (BPIC 2017 before), and approximately 0pp (Traffic Fines). Without hierarchy, the irreparable rate is 31.8% (BPIC 2012) and approximately 35% (BPIC 2017), reduced to 2.3% and approximately 0% respectively; approximately 30–35% of test prefixes on BPIC datasets are hierarchically recovered.

Outcome weighting provides modest and inconsistent benefit over uniform targets: +0.4pp (BPIC 2012), +1.5pp (BPIC 2017 after), −0.3pp (BPIC 2017 before), and approximately 0pp (Traffic Fines).

### 6.3 Sensitivity to Soft Constraint Quality

Fig. 3 presents the noise sensitivity results on BPIC 2012. ECR drops from 0.574 to approximately 0.345 (−40%) at 10% noise and plateaus. Hierarchical relaxation triggers for approximately 65% of training prefixes under noise (versus 30% at baseline), shifting CCOD targets from  $\mathcal{F}_{\text{full}}$  ( $\approx 3$  activities) to  $\mathcal{F}_{\text{hard}}$  ( $\approx 22.5$  activities). OIR *increases* from 0.854 to 0.918 while CSR *decreases* from 0.677 to 0.400.

Per-seed variation is high at noise levels above 0% (ECR std  $\approx 0.18$ ). The per-seed noisy relaxation rate correlates strongly with ECR degradation ( $r = -0.91$ ). A single `not_co_existence` constraint between two common activities can render  $\mathcal{F}_{\text{full}} = \emptyset$  for more than 90% of prefixes, while constraints on rare activities have minimal effect; the damage saturates at 10% noise.

### 6.4 Discussion

The experimental results illuminate the mechanism by which NeSyPA achieves its effectiveness and reveal important design considerations.

The primary mechanism of CCOD is outcome distillation within the feasible space. Pre-computing feasibility and constructing outcome-weighted targets teaches the model to produce candidates with high outcome lift among feasible activities. The inference-time filtering then selects the best compliant recommendation from this improved candidate set. The model does not need to internalize compliance at the raw top-1 level; it needs to produce a top- $K$  set rich in outcome-improving feasible activities. The dramatically higher OIR for NeSyPA (0.849, 0.718, 0.735 on BPIC datasets versus 0.505–0.711 for baselines) confirms this mechanism. Meanwhile, CSR parity across methods ( $\sim 0.60$ – $0.68$ ) confirms that CCOD improves outcome quality without sacrificing constraint satisfaction, with no compliance-versus-outcome tradeoff. The failure of B4 (NeSy-PPM) on Traffic Fines (CSR = 0.753 vs. 0.940 for filtering baselines) further illustrates the fragility of logic-as-loss approaches for strict sequential constraints, where gradient-based compliance signals are insufficient.

The absolute ECR values (roughly 0.40 to 0.57) should be read against an attainable ceiling rather than the nominal maximum of 1. A recommendation counts toward ECR only when it is both compliant and outcome-improving, where outcome-improving means that the predicted positive-outcome probability strictly increases when the recommended activity is appended. For many prefixes no feasible activity raises the predicted outcome, for instance when the case is already on a favorable trajectory or its outcome is largely determined, so an ECR of 1 is unreachable even for an oracle restricted to the feasible set. The meaningful signal is therefore the relative margin over baselines on the same ceiling, where NeSyPA improves ECR by +6.7 to +9.9pp on the BPIC datasets, driven by the higher OIR reported above.

The ablation and sensitivity experiments together clarify the distinct role of each constraint tier. CCOD’s gains originate primarily from prefixes where the full feasible set is non-empty: tight feasible sets ( $|\mathcal{F}_{\text{full}}| \approx 3$ ) produce highly concentrated soft targets that teach the model precisely which compliant activities yield the highest outcome lift, directly fulfilling the paper’s core motivation. The ablation (Experiment 2) shows that hierarchy adds minimal incremental ECR (approximately 0 to +1.1pp across datasets), which might suggest it is unnecessary. However, without hierarchy, 31.8% of prefixes on BPIC 2012 would be irreparable: no recommendation can be made and no CCOD training signal can be computed for them. Hierarchy reduces this to 2.3%, enabling both inference coverage and CCOD training for those prefixes. Yet, as the ablation reveals, CCOD operates on diffuse hard-only targets ( $|\mathcal{F}_{\text{hard}}| \approx 22.5$ ) for these recovered prefixes, yielding only modest and mixed ECR gains. Hierarchy’s value is therefore structural rather than incremental: it ensures the system remains operational and trainable for cases that would otherwise be excluded entirely. The sensitivity analysis (Experiment 3) confirms that soft constraint quality is critical for CCOD effectiveness: when noisy soft constraints trigger frequent hierarchical relaxation, targets become diffuse and the training signal weakens sharply ( $r = -0.91$  between relaxation rate and ECR).

Several limitations should be noted. CCOD effectiveness is sensitive to soft constraint quality: when noisy soft constraints trigger frequent hierarchical relaxation, CCOD targets become diffuse and the model fails to learn soft constraint compli-

ance. This is inherent to the compilation approach, since the quality of compiled targets cannot exceed the quality of the input constraints. The frozen outcome head used for computing outcome lifts is trained on correlational rather than causal data, introducing potential bias in the outcome estimates. The method assumes Declare-expressible constraints and cannot handle arbitrary procedural constraints. Finally, the binary relaxation mechanism (i.e., enforcing all soft constraints or dropping all of them) is a notable limitation. When a single low-confidence soft constraint conflicts with otherwise feasible activities, the entire soft constraint set is discarded rather than selectively relaxing only the conflicting constraint. Because each soft constraint is already compiled into an individual DFA, this limitation is architecturally bounded: identifying which constraint(s) cause an empty feasible set requires at most  $n$  DFA evaluations per activity, so selective relaxation is computationally tractable. The challenge is therefore not in feasibility checking but in the selection policy, i.e., deciding which constraints to retain when not all can be simultaneously satisfied, and in how variable constraint subsets across prefixes affect the stationarity of CCOD training targets.

## 7 Threats to Validity

*Internal Validity.* Soft constraint mining quality affects all constraint-aware methods equally, as the same knowledge base  $\Phi$  is used for all methods. Hyperparameters were fixed across methods to values guided by preliminary runs on the validation split, rather than tuned per method, which could understate the best achievable performance of individual baselines. To ensure fair comparison, B2 receives the same hierarchical relaxation mechanism as NeSyPA at inference time, isolating the contribution of CCOD training. All methods share the same Transformer encoder architecture and training infrastructure.

*External Validity.* We evaluate on four event logs, providing reasonable but not exhaustive generalizability. All datasets come from the standard BPM benchmark suite, and performance on processes with fundamentally different characteristics, such as highly unstructured creative processes or very short traces, remains to be investigated. The constraint knowledge bases were constructed by the authors, with hard constraints following published regulations and soft constraints discovered via mining; different constraint formulations could yield different results.

*Construct Validity.* ECR captures the dual objective of prescriptive monitoring but weights compliance and outcome improvement equally. In practice, one dimension may be more important than the other, and a weighted variant of ECR could better reflect domain-specific priorities. The frozen B1 outcome head used for both outcome evaluation and CCOD target computation may introduce systematic bias if the outcome predictor has learned spurious correlations from the training data.

## 8 Conclusion

We presented NeSyPA, a neurosymbolic method for prescriptive process monitoring that addresses two practical challenges: over-constrained knowledge bases that leave substantial fractions of cases without feasible recommendations, and the lack of mechanisms to compile constraint knowledge into the neural training process. Hierarchical Constraint Relaxation distinguishes hard constraints from relaxable soft constraints, reducing irreparable rates from 31.8% to 2.3% while preserving hard constraint compliance. Constraint-Compiled Outcome Distillation creates feasibility-masked, outcome-weighted targets that teach the model to produce candidates with dramatically higher outcome quality, with no iterative loop and no  $\lambda$  tradeoff parameter. Experiments on four real-world event logs demonstrate that NeSyPA achieves the best Effective Compliance Rate on all test sets, with gains of +6.7 to +9.9 percentage points over the strongest baselines on BPIC datasets, driven primarily by improved outcome quality within the compliant recommendation space.

Several directions for future work emerge from our findings. Granular per-constraint relaxation could selectively relax individual soft constraints based on confidence or importance scores, avoiding the coarseness of binary relaxation. Multi-step lookahead could extend feasibility checking to consider whether an activity opens or closes future feasible paths, not just immediate compliance. Online CCOD could support incremental target updates as new cases complete, rather than one-shot fine-tuning. Finally, extending the prescriptor to recommend activity-time pairs would enable meaningful temporal constraint enforcement.

## References

1. Agarwal, P., Gupta, A., Sindhgatta, R., Dechu, S.: Goal-oriented next best activity recommendation using reinforcement learning. *CoRR* **abs/2205.03219** (2022)
2. Bozorgi, Z.D., Teinemaa, I., Dumas, M., Rosa, M.L., Polyvyanyy, A.: Process mining meets causal machine learning: Discovering causal rules from event logs. In: van Dongen, B.F., Montali, M., Wynn, M.T. (eds.) *ICPM*. pp. 129–136. IEEE (2020)
3. Bozorgi, Z.D., Teinemaa, I., Dumas, M., Rosa, M.L., Polyvyanyy, A.: Prescriptive process monitoring for cost-aware cycle time reduction. In: Ciccio, C.D., Francescomarino, C.D., Soffer, P. (eds.) *ICPM*. pp. 96–103. IEEE (2021)
4. Branchi, S., Francescomarino, C.D., Ghidini, C., Massimo, D., Ricci, F., Ronzani, M.: Learning to act: A reinforcement learning approach to recommend the best next activities. In: Ciccio, C.D., Dijkman, R.M., del-Río-Ortega, A., Rinderle-Ma, S. (eds.) *BPM Forum*. pp. 137–154. LNBIP, Springer (2022)
5. Ciccio, C.D., Mecella, M.: On the discovery of declarative control flows for artful processes. *ACM Trans. Manag. Inf. Syst.* **5**(4), 24:1–24:37 (2015)
6. van Dongen, B.: BPI challenge 2012. *4TU.ResearchData* (2012)
7. van Dongen, B.: BPI challenge 2017. *4TU.ResearchData* (2017)
8. Fahrenkrog-Petersen, S.A., Tax, N., Teinemaa, I., Dumas, M., Leoni, M.d., Maggi, F.M., Weidlich, M.: Fire now, fire later: alarm-based systems for prescriptive process monitoring. *Knowl. Inf. Syst.* **64**(2), 559–587 (Feb 2022)

9. Gaikwad, D., van der Aalst, W.M.P., Park, G.: Neuro-symbolic process anomaly detection. In: Fuentes, L., Plebani, P., Combi, C., Reijers, H. (eds.) CAiSE. LNCS, Springer (2026)
10. Giacomo, G.D., Vardi, M.Y.: Linear temporal logic and linear dynamic logic on finite traces. In: Rossi, F. (ed.) IJCAI. pp. 854–860. IJCAI/AAAI (2013)
11. Hinton, G.E., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. CoRR **abs/1503.02531** (2015)
12. de Leoni, M., Mannhardt, F.: Road traffic fine management process. 4TU.ResearchData (2015)
13. Mezini, A., Umili, E., Donadello, I., Maggi, F.M., Mancanelli, M., Patrizi, F.: Neuro-symbolic predictive process monitoring. CoRR **abs/2509.00834** (2025)
14. Pesic, M., van der Aalst, W.M.P.: A declarative approach for flexible business processes management. In: Eder, J., Dustdar, S. (eds.) BPM Workshops. pp. 169–180. LNCS, Springer (2006)
15. Santis, F.D., Park, G., van der Aalst, W.M.P., Zanichelli, F.: Compliance-aware predictive process monitoring: A neuro-symbolic approach. In: Fuentes, L., Plebani, P., Combi, C., Reijers, H. (eds.) CAiSE. LNCS, Springer (2026)
16. Santis, F.D., Park, G., Zanichelli, F.: Neuro-symbolic artificial intelligence for real-time sepsis mortality prediction. Eng. Appl. Artif. Intell. **177**, 114920 (2026)
17. Santis, F.D., Park, G., Zanichelli, F.: Neuro-symbolic learning for predictive process monitoring via two-stage logic tensor networks with rule pruning. CoRR **abs/2603.26944** (2026)
18. Shoush, M., Dumas, M.: Prescriptive process monitoring under resource constraints: A causal inference approach. In: Munoz-Gama, J., Lu, X. (eds.) ICPM Workshops. pp. 180–193. LNBIP, Springer (2021)
19. Shoush, M., Dumas, M.: When to intervene? prescriptive process monitoring under uncertainty and resource constraints. In: Ciccio, C.D., Dijkman, R.M., del-Río-Ortega, A., Rinderle-Ma, S. (eds.) BPM Forum. pp. 207–223. LNBIP, Springer (2022)
20. 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) (Mar 2019)
21. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: Guyon, I., von Luxburg, U., Bengio, S., Wallach, H.M., Fergus, R., Vishwanathan, S.V.N., Garnett, R. (eds.) NeurIPS. pp. 5998–6008 (2017)
22. Weinzierl, S., Dunzer, S., Zilker, S., Matzner, M.: Prescriptive business process monitoring for recommending next best actions. In: Fahland, D., Ghidini, C., Becker, J., Dumas, M. (eds.) BPM. LNBIP, vol. 392, pp. 193–209. Springer (2020)