A Preliminary Approach to Probabilistic Process Discovery

Michela Vespa[1,*], Elena Bellodi[1], Daniela Loreti[2] and Federico Chesani[2]

[1]Department of Engineering, University of Ferrara, Via Saragat 1, Ferrara, Italy
[2]Department of Computer Science and Engineering, Alma mater studiorum – University of Bologna, Viale Risorgimento 2, Bologna, Italy

Abstract

Process discovery is a fundamental task in Process Mining, concerned with the extraction of process models from event logs. Alongside traditional deterministic approaches, recent works have started to consider probabilistic declarative process models, where model constraints are learnt and then annotated with weights to account for uncertainty and variability in observed behavior, as well as noise. In this setting, weights have been interpreted, up to now, in terms of the constraints' support in the observed traces. However, the probabilistic process discovery task remains quite unexplored, often relying on the extraction of constraints by deterministic miners. In this paper, we report preliminary results on the use of PASCAL, a Probabilistic Inductive Constraint Logic algorithm, based on the Distribution Semantics, which is able to discover probabilistic declarative process models, i.e., integrity constraints annotated with a probability, directly from process traces labelled as positive and negative, without the need of a two-step approach. Preliminary results on two synthetic logs indicate that the learned theories recover most of the constraints underlying the process models.

Keywords

Declarative Process Modeling, Process Discovery, Probabilistic Integrity Constraints, Distribution Semantics

1. Introduction

Process discovery is the automatic induction of a process model from event logs recorded by information systems, and is one of the major tasks in Process Mining [1, 2]. The model produced by discovery is the basis on which conformance checking, performance analysis, and process improvement subsequently rely; its expressiveness and faithfulness to the underlying process therefore influence the validity of subsequent insights.

Traditionally, process models have been deterministic. Procedural miners (e.g. AlphaMiner [3], Inductive Miner [4], etc.) reconstruct an explicit control-flow graph that prescribes which sequences of activities are admissible; declarative miners (e.g. MINERful [5]), on the other hand, induce a set of *constraints* that traces must satisfy. The most relevant example of declarative modelling language is Declare [6, 7, 8], which provides constraints in terms of rules that have to be followed during execution, with temporal relations among log activities.

Both views share a binary notion of conformance: a trace, i.e. an ordered sequence of events referring to a process excecution, either complies with the model or it does not. This view may not be sufficient with contemporary business processes, such as knowledge-intensive workflows, clinical pathways, and regulatory protocols, where behavior is governed by guidelines rather than rigid procedures, exceptions are routine, and the strength with which a rule is enforced varies across cases and contexts [9]. In such settings, the relevant question is rarely whether a trace conforms to the model, but how strongly it conforms and which of the violated rules are critical rather than tolerated. *Probabilistic* declarative process modeling has emerged to tackle this issue: each constraint is annotated with a probability that quantifies its strength or expected satisfaction [10, 11].

CILC 2026: The 41st Italian Conference on Computational Logic, June 23–25, 2026, Ferrara, Italy
*Corresponding author.
michela.vespa@unife.it (M. Vespa); elena.bellodi@unife.it (E. Bellodi); daniela.loreti@unibo.it (D. Loreti); federico.chesani@unibo.it (F. Chesani)
0009-0004-4350-8151 (M. Vespa); 0000-0002-3717-3779 (E. Bellodi); 0000-0002-6507-7565 (D. Loreti); 0000-0003-1664-9632 (F. Chesani)
© 2026 This work is licensed under a "CC BY 4.0" license.

The discovery counterpart of this idea, however, remains unexplored. Current probabilistic declarative discovery proceeds in two disjoint steps: a deterministic miner first extracts the model, then probabilities are estimated from how often each constraint is satisfied in the log [11]. For instance, if a probabilistic constraint indicates that an order is accepted with probability 0.7 it means that 70% of the times in the log (in 70% of the traces) that constraint is satisfied. In this way, the probabilities, once attached, can only describe what the deterministic miner has already included in the model. Often the entire pipeline operates on positive traces only, so deviant executions cannot be taken into consideration.

In this paper, we report a preliminary investigation of an alternative approach: discovering the structure and the probabilities of a declarative process model jointly, in a single learning step, from traces labeled as compliant or not. In this way, we take into account information on both compliant and deviant traces as well as on noise which makes the domain uncertain, and we make a step further with respect to our previous work [12], where we assumed that the process model constraints' were manually annotated with probabilities.

We propose to apply PASCAL ("ProbAbiliStic inductive ConstrAint Logic") [13], an algorithm that learns Probabilistic Constraint Logic Theories (PCLTs), i.e. sets of integrity constraints annotated with probabilities. We showed that a PCLT is itself a probabilistic declarative process model in [14]: its integrity constraints encode Declare rules and the probability attached to each constraint is interpreted as the strength/importance of the corresponding process rule. Reframed this way, probabilistic process discovery becomes an instance of the Inductive Logic Programming learning-from-interpretations problem: each trace is a labelled logical interpretation and the discovered theory represents both a declarative probabilistic process model and a probabilistic classifier which can be used to assign a compliance probability of being positive to new traces.

Our contributions are the following: (i) a formulation of the probabilistic declarative process discovery task as a discriminative learning-from-interpretations problem with the formalism of PCLT; (ii) an empirical evaluation on two synthetic logs showing promising results in recovering the core constraints of the original process specifications and in classifying test instances with good classification performance.

The remainder of the paper is organized as follows. Section 2 discusses related work. Section 3 presents the formal background on declarative process specifications, PCLTs, and PASCAL. Section 4 reports the experimental setup and the results. Section 5 concludes the paper.

2. Related Work

Process discovery has historically focused on procedural paradigms. Procedural approaches, such as the α-algorithm [3] and the Heuristics Miner [15], reconstruct explicit control-flow graphs from event logs, prescribing a fixed set of allowed execution paths. While effective for highly structured workflows, these techniques induce deterministic models that, when applied to flexible, knowledge-intensive processes, tend either to overfit infrequent behavior or to produce overly permissive models that sacrifice precision. In particular, they struggle to abstract from noise and to capture high-level regularities in domains where variability and exceptional executions are the norm. To account for the inherent variability of real-world processes, stochastic process mining [16] shifts the focus from mere control-flow reconstruction to modeling the probability distribution over observed behaviors. This paradigm is grounded in trace multiplicity (i.e., the frequency of each unique trace in the log), which provides an empirical estimate of the likelihood that the process exhibits a given execution. Within this procedural stochastic landscape, recent works have tried to reduce the number of weight parameters needed when discovered process trees are converted into stochastic Petri nets. Frameworks such as the Toothpaste Miner [17] apply polynomial-time reduction and abstraction rules to synthesize Generalized Stochastic Petri Nets from weighted execution models, while Stochastic Process Trees (SPTs) [18, 19] enrich standard tree operators (choice, parallel, loop) with probabilistic arguments to reduce parameter dimensionality and clarify their localized impact on trace generation. Despite these structural refinements, procedural stochastic discovery remains fundamentally constrained by its indirect, frequency-driven pipeline: control-flow

structure is fixed first (typically via block-structured deterministic miners such as the Inductive Miner [4] or Split Miner[20]), and stochasticity is subsequently calibrated through frequency-, alignment-, or likelihood-based weight estimation against positive-only trace multiplicities. Consequently, the resulting probabilities capture routing frequencies and branching ratios at individual transitions rather than the strength, reliability, or tolerance of business rules. Moreover, because they operate exclusively on positive logs, they cannot natively leverage negative examples to discriminatively learn which deviations constitute critical violations versus benign exceptions [21].

These structural and semantic limitations motivate a shift toward declarative process modeling. Unlike procedural approaches that prescribe *how* a process must be executed, declarative formalisms such as Declare [8] and logic-based frameworks like SCIFF [22] specify *what* conditions valid executions must satisfy, by expressing temporal relationships between activities through LTL_f formulas or abductive integrity constraints that capture obligations, permissions, and prohibitions. Every execution is implicitly allowed unless it violates a constraint, naturally accommodating concurrency, loops, and alternative branching without exhaustive path enumeration [23]. This constraint-centric representation directly mirrors regulatory requirements and organizational policies, offering interpretability for domain experts and a clear separation of admissible from forbidden behavior.

Efforts to introduce uncertainty into declarative models have followed two main directions. The first, exemplified by ProbDeclare [11], lifts declarative constraints by annotating them with probability conditions specifying the expected proportion of satisfying traces. For the process discovery part, constraints are first mined deterministically using off-the-shelf discovery algorithms, and probabilities are subsequently estimated from empirical frequencies of satisfaction of each constraint in the log traces, rather than representing the strength or importance of a constraint in the process domain. The second direction grounds probabilistic declarative models in Probabilistic Logic Programming (PLP), introducing Probabilistic Declarative Process Specifications (PDSs) under the Distribution Semantics [12]. In [12] we exploited an inference module available in the PASCAL algorithm to perform the task of conformance checking, while in this paper we extend its application to process discovery.

A complementary line of work recasts discovery as a supervised learning problem by exploiting labeled traces. In the procedural domain, Goedertier et al. [24] introduced Artificially Generated Negative Events (AGNEs) to frame discovery as an ILP classification problem. In the declarative setting, early work adapted Inductive Constraint Logic to learn SCIFF integrity constraints from labeled traces [25, 26], demonstrating that counter-examples significantly constrain the hypothesis space and improve model precision; subsequent work on synthetic log generation further validated the value of labeled datasets for benchmarking and iterative refinement [27]. The resulting models, however, remain deterministic, classifying traces as strictly compliant or violating without quantifying the strength or reliability of individual constraints.

Our approach addresses these limitations by adopting the PASCAL algorithm to perform probabilistic process discovery. Unlike a two-step approach, our method leverages supervised learning from labeled traces to jointly induce structure (constraints) and probabilities.

3. Background

3.1. Declarative Process Specifications

In process mining, an *event log* $\mathcal{L}$ is a multiset of *traces*, where each trace $t = \langle e_1, e_2, \ldots, e_n \rangle$ records a temporally ordered sequence of activity executions for a single process instance. Each event e_i corresponds to an occurrence of an activity at a specific point in time, and the timestamps satisfy $timestamp(e_i) \leq timestamp(e_{i+1})$ for $1 \leq i < n$. *Process discovery* is the task of inferring a process model from $\mathcal{L}$ that explains the observed behavior and generalizes to unseen executions. Declarative approaches to this task represent the process not as an explicit control-flow graph but as a compact set of constraints that compliant traces must satisfy.

Our work builds upon Declare [8], the most prominent declarative modeling formalism, which offers a collection of (graphical) constraint templates with formal semantics grounded in LTL_f logic

[28]. Some representative constraints are, for instance, *response(a,b)*, meaning that whenever activity a occurs, activity b must eventually follow in the process instance, and *precedence(a,b)*, meaning that activity b can only be executed if activity a has been executed before.

Formally, a *Declarative Process Specification* is a triple $DS = (\textsc{Rep}, \Sigma, C)$, where Rep is a set of parameterized constraint templates (e.g., response, precedence, etc.), Σ is the activity alphabet, and C is a finite set of instantiated constraints over Σ [23]. In many real-world processes, constraints do not hold universally but are subject to exceptions, partial compliance, or domain uncertainty. To model this, a *Probabilistic Declarative Process Specification* (PDS) [12] extends a DS by annotating each constraint with a probability $p_i \in [0, 1]$, reflecting its strength or importance in the process domain.

Definition 1 (Probabilistic Constraint [12]). *A* probabilistic constraint *is an instantiated template* $c(a_1, \ldots, a_m) \in C$ *annotated with* $p \in [0, 1]$*, written* $p_i :: c_i(a_1, \ldots, a_m)$*. Constraints with* $p_i = 1$ *are* crisp*.*

Example 1. *Consider a probabilistic response constraint*

$$0.85 :: \textsc{response}(end_surgery, mobilization)$$

in a clinical domain, stating that whenever surgery ends, patient's mobilization will follow with probability 0.85*. The probability value reflects the constraint's strength in the process model, representing how mandatory it is for optimal care. Early mobilization is a core goal in clinical guidelines ([29]) to prevent complications like pneumonia, but exceptions are allowed (e.g., patient frailty, severe pain), and are taken into account by the attached probability.*

Under the Distribution Semantics [30], a PDS defines a probability distribution over deterministic specifications (possible worlds), each generated by independently including or excluding each probabilistic constraint with probability p_i.

3.2. Probabilistic Constraint Logic Theories as Process Models

The PASCAL algorithm learns Probabilistic Constraint Logic Theories (PCLT) [13]. A PCLT is a set of *Probabilistic Integrity Constraints* (PICs), each of the form:

$$p_i :: L_1, \ldots, L_b \rightarrow \exists(P_1); \ldots; \exists(P_n); \forall\neg(N_1); \ldots; \forall\neg(N_m) \tag{1}$$

where $p_i \in [0, 1]$ is a probability, each L_i is a literal and each P_j and N_k is a conjunction of literals $d_1 \wedge \cdots \wedge d_k$. The semicolon here represents a disjunction. We call each $\exists(P_j)$ a P disjunct and each $\forall\neg(N_k)$ an N disjunct. $L_1, \ldots, L_b$ is called the body of C ($Body(C)$) and $\exists(P_1); \ldots; \exists(P_n); \forall\neg(N_1); \ldots; \forall\neg(N_m)$ is called the head of C ($Head(C)$). Body variables are universally quantified over the full PIC; head variables not appearing in the body are quantified existentially in P disjuncts and universally in N disjuncts.

As done in [14], Declare constraints can be encoded as PICs, as shown in the example below.

Example 2. *According to Eq. 1, the* Declare *constraint from Example 1 is translated into the following PIC:*

$$0.85 :: end_surgery(T_1) \rightarrow mobilization(T_2) \wedge T_2 > T_1$$

where T_1 *and* T_2 *are the timestamps of the two events.*

By representing a probabilistic process model, a PCLT $T = \{p_1 :: C_1, \ldots, p_n :: C_n\}$ can assign a process trace t, represented as a logical interpretation I_t, a probability of being compliant.

According to the underlying Distribution Semantics [30], each probability p_i is the sum of the probabilities of the possible worlds where a grounding of constraint C_i is present. For a more detailed discussion on the semantics please refer to [13].

3.3. Learning PCLTs from Labeled Traces

Traditional process discovery aims to reconstruct a model that generalizes beyond observed executions. In our framework, it can be seen as the problem of learning a PCLT from a set of labeled traces, divided into positive interpretations $\mathcal{I}^+$ (compliant executions) and negative interpretations $\mathcal{I}^-$ (deviant executions). The PASCAL algorithm [13] solves this problem in the ILP *learning from interpretations* setting [31]. Rather than seeking a deterministic theory that perfectly separates the two classes, PASCAL learns a PCLT that assigns each trace I the probability $P(\oplus \mid I)$ of belonging to the positive class $\oplus$, in other words of being a compliant execution:

$$P(\oplus \mid I) = \prod_{i=1}^{n} (1 - p_i)^{m_i} \tag{2}$$

where m_i is the number of groundings of constraint C_i not satisfied in I, and n is the number of PICs in the theory.

Discovery proceeds in three steps as follows.

1. Each process trace is represented as a logical interpretation, with one ground fact for each activity in the trace, with at least one argument for the timestamp and possibly other domain-specific attributes. Each interpretation is labeled pos or neg, as requested for discriminative learning.

   Example 3. *Consider a process trace from an e-commerce log, represented as a logical interpretation. Each event is encoded as a ground fact with the event as predicate name, some relevant attributes if any, and the timestamp:*

   ```
   begin(model(1)).
     access(917.2).
     login('M', 33, 924.5).
     purchase(9, 140.90, 939.7).
     feedback(4, 947.2).
     pos.
   end(model(1)).
   ```

   Here, $access(917.2)$ *records a site visit at time 917.2;* $login('M', 33, 924.5)$ *records a login by a male user aged 33 at time 924.5;* $purchase(9, 140.9, 939.7)$ *a purchase of 9 items for €140.90 at time 939.7;* $feedback(4, 947.2)$ *records a feedback score of 4 at time 947.2. The label* pos *marks the trace as compliant with respect to the business process.*

2. Definition of the Language Bias. The hypothesis space of the algorithm is constrained via *mode* declarations ([32]), which specify which predicates may appear in clauses' head and body respectively. The language bias is designed so that every learnable PIC corresponds to a Declare template.
3. Learning phase. PASCAL performs a beam search over the clause space, with the aim of maximizing $\prod_{I \in \mathcal{L}} P(\oplus \mid I)$, i.e. the joint likelihood of the training set of interpretations $\mathcal{L}$ belonging to the positive class.

4. Experiments

We assess whether PASCAL can discover the integrity constraints of a declarative process specification that we know a priori, when given both compliant and deviant traces. PASCAL is implemented in SWI-Prolog and distributed as a SWI-Prolog pack[1]. All experiments were executed on a standard workstation (3.2 GHz, 16 GB RAM), with a 24-hour wall-clock limit.

We evaluated it on two synthetic logs: an e-commerce process and a clinical process derived from the "Enhanced Recovery After Surgery" (ERAS®) protocol [29], a multimodal perioperative care protocol aimed at reducing surgical stress and accelerating patient recovery after surgery.

[1] https://eu.swi-prolog.org/pack/list?p=pascal

Table 1
Possible sequences of events in the traces for the e-commerce log (left) and the ERAS clinical log (right).

E-commerce	ERAS
access → login → purchase → shipping → feedback	preadmission_info → [smoking_cessation ‖ nutrition_screen ‖ anaemia_management]
access → login → purchase → feedback	[epidural_placement ⊕ tap_block] → anesthesia_start → abx_prophylaxis
access → login → purchase → shipping	incision → end_surgery → icu_admission → mobilization → discharge
access → login → purchase	incision → end_surgery → mobilization → discharge
access → purchase → shipping	anesthesia_start → abx_prophylaxis → incision → end_surgery
access → purchase	preadmission_info → anesthesia_start → discharge

Notation: → sequential dependency; ‖ parallel execution; ⊕ mutually exclusive choice.

The e-commerce log is generated from a ground-truth declarative specification of an on-line shopping process over five activity types: *access(Time), login(Gender, Age, Time), purchase(#Items, Price, Time), shipping(Cost, Time), and feedback(Score, Time)*, where *Time* denotes the event timestamp. The specification admits 6 user behaviors with an optional login and optional events after purchase (Table 1, left).

The ERAS log contains traces representing patients for whom 13 clinical activities are registered, from pre-operative preparation to surgery to intra-operative care, to post-operative recovery: *preadmission_info(Age, Bmi, Smoking_status, Hb_baseline, Incision_approach, Time), Smoking_cessation(Packyears nutrition_screen(Nrs2002, Bmi, Albumin, Time), anaemia_management(Iv_iron, Rbc_units, Time), epidural_placement(Attempts, Time), tap_block[2](Success, Time), Anesthesia_start(Time), abx[3]_prophylaxis(Time), incision(Approach, Time), end_surgery(Duration_min, Blood_loss_ml, Time), icu[4]_admission(Duration_min, Time), mobilization(Goal_met, Time), and discharge(Los_days, Time).*

Both logs contain 5,000 traces (2,500 positive, 2,500 negative). Negative traces are produced by randomly removing events from valid pathways. Each log is split according to the proportion 70/30% with stratification, yielding 3,500 training traces (1,750 per class) and 1,500 test traces (750 per class).

PASCAL was configured with the following hyperparameters: gradient descent as the parameter learning algorithm with maximum 100 iterations, maximum number of body literals = 1 and of head disjuncts = 1, maximum number of beam search iterations = 10, size of the beam = 4, maximum number of final PICs = 50.

4.1. Results

PASCAL returns a PCLT and several machine learning metrics computed on the test set: log-likelihood (LL), area under the ROC curve (AUC-ROC), and area under the precision-recall curve (AUC-PR).

We qualitatively checked if and how the learnt PCLTs matched the ground-truth specifications, i.e. the ability to directly discover a probabilistic declarative process specification (PDS) from event logs. We report below the most relevant models for each log.

[2]transversus abdominis plane block
[3]antibiotic prophylaxis
[4]Intensive Care Unit

As for the e-commerce log, PASCAL discovers the following PICs in the form of *precedence* constraints:

$$
\begin{array}{lll}
0.262 :: & \texttt{access}(A) \rightarrow \texttt{purchase}(B, C, D) & (1) \\
0.255 :: & \texttt{access}(A) \rightarrow \texttt{login}(B, C, D) & (2) \\
0.253 :: & \texttt{purchase}(A, B, C) \rightarrow \texttt{feedback}(D, E) & (3) \\
0.262 :: & \texttt{login}(A, B, C) \rightarrow \texttt{feedback}(D, E) & (4)
\end{array}
$$

This theory obtains, on the test set, an AUCROC of 0.84, AUCPR of 0.79 and log-likelihood of -663.142.

The performance metrics show that the theory is able to classify well the test traces between compliant and non compliant.

On the ERAS clinical log, PASCAL discovers the following PICs:

$$
\begin{array}{lll}
0.290 :: & \texttt{preadmission_info} \rightarrow \texttt{incision} \wedge \texttt{nutrition_screen} & (5) \\
0.168 :: & \texttt{preadmission_info} \rightarrow \texttt{abx_prophylaxis} & (6) \\
0.157 :: & \texttt{end_surgery} \rightarrow \texttt{icu_admission} \wedge \texttt{mobilization} & (7) \\
0.379 :: & \texttt{preadmission_info} \rightarrow \texttt{discharge} & (8) \\
0.168 :: & \texttt{smoking_cessation} \rightarrow \texttt{mobilization} & (9)
\end{array}
$$

These are the best PICs reconstructing the sequences selected across multiple runs. This experiment is harder due to the higher complexity of the domain in terms of number of activities and sequences between activities. We tried to capture this complexity by allowing longer probabilistic clauses, as shown above. Analyzing PICS no. 1,3,4 relative to the e-commerce log and PIC no. 8 relative to ERAS log it's easy to see that PASCAL is able to capture indirect constraints between activities that are not immediately sequential in time.

5. Conclusions and Future Work

In this paper, we have reported a preliminary investigation on probabilistic declarative process discovery as a discriminative learning-from-interpretations problem in probabilistic logic programming. By adopting the PASCAL algorithm, we learn both the structure and the probabilities of a Probabilistic Constraint Logic Theory directly from labeled event logs in one step. This theory corresponds to a probabilistic declarative process specification.

Future work will address several limitations of the current approach: managing the complexity of the most challenging domains by leveraging on the language bias, consistently discover temporal constraints and testing on real-world logs, both to test the method under realistic noise and to assess the interpretability of the learned theories.

Acknowledgments

Research funded by the Italian Ministerial grant PRIN 2022 "Probabilistic Declarative Process Mining (PRODE)", n. 20224C9HXA - CUP F53D23004240006, funded by European Union – Next Generation EU.

Declaration on Generative AI

The authors declare that in the planning, drafting, and/or revision of the work have made no use of generative AI tools.

References

[1] W. van der Aalst, Process Mining: Data Science in Action, 2nd ed., Springer Publishing Company, Incorporated, 2016. doi:10.1007/978-3-662-49851-4.

[2] W. M. P. van der Aalst, Foundations of process discovery, in: W. M. P. van der Aalst, J. Carmona (Eds.), Process Mining Handbook, Springer International Publishing, Cham, 2022, pp. 37–75. doi:10.1007/978-3-031-08848-3_2.
[3] W. van der Aalst, T. Weijters, L. Maruster, Workflow mining: discovering process models from event logs, IEEE Transactions on Knowledge and Data Engineering 16 (2004) 1128–1142. doi:10.1109/TKDE.2004.47.
[4] S. J. J. Leemans, D. Fahland, W. M. P. van der Aalst, Discovering block-structured process models from event logs - a constructive approach, in: J.-M. Colom, J. Desel (Eds.), Application and Theory of Petri Nets and Concurrency, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013, pp. 311–329.
[5] C. Di Ciccio, M. Montali, Declarative process specifications: Reasoning, discovery, monitoring, in: W. M. P. van der Aalst, J. Carmona (Eds.), Process Mining Handbook, Springer International Publishing, Cham, 2022, pp. 108–152. doi:10.1007/978-3-031-08848-3_4.
[6] M. Pesic, Constraint-based workflow management systems : shifting control to users, Phd thesis 1 (research tu/e / graduation tu/e), Industrial Engineering and Innovation Sciences, 2008. doi:10.6100/IR638413, proefschrift.
[7] W. M. van der Aalst, M. Pesic, H. Schonenberg, Declarative workflows: Balancing between flexibility and support, Computer Science - Research and Development 23 (2009) 99–113. URL: https://api.semanticscholar.org/CorpusID:18017854.
[8] 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.
[9] C. Di Ciccio, A. Marrella, A. Russo, Knowledge-intensive processes: An overview of contemporary approaches, in: Proceedings of the 1st International Workshop on Knowledge-intensive Business Processes (KiBP 2012) Rome, Italy, June 15, 2012, volume 861, CEUR Workshop Proceedings, Sun SITE Central Europe, 2012, pp. 33–47.
[10] M. Vespa, E. Bellodi, et al., Probabilistic compliance of uncertain traces in declarative process mining, in: Proceedings of the 40th Italian Conference on Computational Logic Alghero, Italy, June 25-27, 2025, volume 4003, CEUR WORKSHOP PROCEEDINGS, Sun SITE Central Europe, 2025, pp. 1–9.
[11] A. Alman, F. M. Maggi, M. Montali, R. Peñaloza, Probabilistic declarative process mining, Inf. Syst. 109 (2022) 102033. doi:10.1016/J.IS.2022.102033.
[12] M. Vespa, E. Bellodi, F. Chesani, D. Loreti, P. Mello, E. Lamma, A. Ciampolini, Probabilistic compliance in declarative process mining, in: G. D. Giacomo, V. Fionda, F. Fournier, A. Ielo, L. Limonad, M. Montali (Eds.), 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, volume 3779 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2024, pp. 11–22. URL: https://ceur-ws.org/Vol-3779/paper1.pdf.
[13] F. Riguzzi, E. Bellodi, R. Zese, M. Alberti, E. Lamma, Probabilistic inductive constraint logic, Machine Learning 110 (2021) 723–754. doi:10.1007/s10994-020-05911-6.
[14] M. Vespa, E. Bellodi, et al., A scalable approach to probabilistic compliance in declarative process mining, in: Joint Proceedings of the Workshops and Doctoral Consortium of the 41st International Conference on Logic Programming (ICLP-WS-DC 2025) co-located with 41st International Conference on Logic Programming (ICLP 2025), Rende, Italy, September 9-13, 2025., volume 4117, CEUR Workshop Proceedings, Sun SITE Central Europe, 2025, pp. 1–12.
[15] A. Weijters, W. Aalst, A. Medeiros, Process Mining with the Heuristics Miner-algorithm, volume 166, 2006.
[16] A. Rogge-Solti, W. M. P. van der Aalst, M. Weske, Discovering stochastic petri nets with arbitrary delay distributions from event logs, in: N. Lohmann, M. Song, P. Wohed (Eds.), Business Process Management Workshops, Springer International Publishing, Cham, 2014, pp. 15–27.
[17] A. Burke, S. J. Leemans, M. T. Wynn, Discovering stochastic process models by reduction and abstraction, in: D. Buchs, J. Carmona (Eds.), Application and Theory of Petri Nets and Concurrency: 42nd International Conference, PETRI NETS 2021, Virtual Event, June 23–25, 2021, Proceedings, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), Springer, Switzerland, 2021, pp. 312–336. doi:10.1007/978-3-030-76983-3_16.
[18] P. Cry, A. Horváth, P. Ballarini, Stochastic process trees: A formal framework for stochastic process discovery, in: 2025 7th International Conference on Process Mining (ICPM), 2025, pp. 1–8. doi:10.1109/ICPM66919.2025.11220624.
[19] A. Horváth, P. Ballarini, P. Cry, Probabilistic process discovery with stochastic process trees, in: M. Gribaudo, M. Iacono, S. Sedigh Sarvestani (Eds.), Performance Evaluation Methodologies and Tools, Springer Nature Switzerland, Cham, 2026, pp. 47–68.
[20] A. Augusto, R. Conforti, M. Dumas, M. La Rosa, A. Polyvyanyy, Split miner: automated discovery of

accurate and simple business process models from event logs, Knowl. Inf. Syst. 59 (2019) 251–284. URL: https://doi.org/10.1007/s10115-018-1214-x. doi:10.1007/s10115-018-1214-x.

[21] F. Chesani, C. Di Francescomarino, C. Ghidini, G. Grundler, D. Loreti, F. M. Maggi, P. Mello, M. Montali, S. Tessaris, Shape your process: Discovering declarative business processes from positive and negative traces taking into account user preferences, in: Enterprise Design, Operations, and Computing: 26th International Conference, EDOC 2022, Bozen-Bolzano, Italy, October 3–7, 2022, Proceedings, Springer-Verlag, Berlin, Heidelberg, 2022, p. 217–234. doi:10.1007/978-3-031-17604-3_13.

[22] M. Alberti, F. Chesani, M. Gavanelli, E. Lamma, P. Mello, P. Torroni, Verifiable agent interaction in abductive logic programming: The SCIFF framework, ACM Trans. Comput. Log. 9 (2008) 29:1–29:43. doi:10.1145/1380572.1380578.

[23] C. D. Ciccio, M. Montali, Declarative process specifications: Reasoning, discovery, monitoring, in: W. M. P. van der Aalst, J. Carmona (Eds.), Process Mining Handbook, volume 448 of *LNBIP*, Springer, 2022, pp. 108–152. doi:10.1007/978-3-031-08848-3_4.

[24] S. Goedertier, D. Martens, J. Vanthienen, B. Baesens, Robust process discovery with artificial negative events, Journal of Machine Learning Research 10 (2009) 1305–1340. URL: http://jmlr.org/papers/v10/goedertier09a.html.

[25] E. Lamma, P. Mello, F. Riguzzi, S. Storari, Applying inductive logic programming to process mining, in: International Conference on Inductive Logic Programming, Springer, 2007, pp. 132–146.

[26] F. Chesani, E. Lamma, P. Mello, M. Montali, F. Riguzzi, S. Storari, Exploiting inductive logic programming techniques for declarative process mining, in: Transactions on Petri Nets and Other Models of Concurrency II: Special Issue on Concurrency in Process-Aware Information Systems, Springer, 2009, pp. 278–295.

[27] D. Loreti, F. Chesani, A. Ciampolini, P. Mello, Generating synthetic positive and negative business process traces through abduction, Knowl. Inf. Syst. 62 (2020) 813–839. URL: https://doi.org/10.1007/s10115-019-01372-z. doi:10.1007/S10115-019-01372-Z.

[28] G. D. Giacomo, M. Y. Vardi, Linear temporal logic and linear dynamic logic on finite traces, in: F. Rossi (Ed.), IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013, IJCAI/AAAI, 2013, pp. 854–860. URL: http://www.aaai.org/ocs/index.php/IJCAI/IJCAI13/paper/view/6997.

[29] U. O. Gustafsson, et al., Guidelines for perioperative care in elective colorectal surgery: Enhanced recovery after surgery (eras®) society recommendations: 2018, World Journal of Surgery 43 (2019) 659–695. doi:10.1007/s00268-018-4844-y.

[30] T. Sato, A statistical learning method for logic programs with distribution semantics, in: L. Sterling (Ed.), Logic Programming, Proceedings of the Twelfth International Conference on Logic Programming, Tokyo, Japan, June 13-16, 1995, MIT Press, 1995, pp. 715–729.

[31] L. De Raedt, W. Van Laer, Inductive constraint logic, in: Proceedings of the 6th International Conference on Algorithmic Learning Theory, ALT '95, Springer-Verlag, Berlin, Heidelberg, 1995, p. 80–94.

[32] S. Muggleton, Inverse entailment and progol, New Gen. Comput. 13 (1995) 245–286. URL: https://doi.org/10.1007/BF03037227. doi:10.1007/BF03037227.