Probabilistic Compliance of Uncertain Traces in Declarative Process Mining

Michela Vespa, Elena Bellodi*

Dipartimento di Ingegneria, Università di Ferrara, Via Saragat 1, Ferrara, Italy

Abstract

This paper presents our work in progress about the integration of Probabilistic Logic Programming (PLP) with Declarative Process Mining (DPM) to address uncertainty in business process management. Traditional DPM approaches, such as DECLARE, use deterministic constraints to permit/forbid activities, but real-world processes often involve incomplete or unreliable data. To bridge this gap, we recap our previous work on introducing *in a separate way* probabilistic extensions for events, traces, and constraints inspired by PLP's Distribution Semantics. We present here an extension to our formal semantics to take into account *at the same time* uncertain events and uncertain constraints in order to perform compliance of a trace versus a process model. Preliminary experiments on a healthcare process demonstrate the approach's feasibility but highlight scalability challenges due to exponential complexity, that will be addressed in future work.

Keywords

Declarative Process Mining, Probabilistic Logic Programming, Distribution semantics

1. Introduction

Research in Business Process Management (BPM) has led to innovative techniques for modeling, discovering, and monitoring process executions. A business process is a structured series of activities that delivers a product or service to a specific customer. These processes range from highly repetitive production tasks (e.g., manufacturing line workflow, e-shop order fulfillment) to flexible, knowledge-intensive activities (e.g., patient treatment in a hospital) [1]. Within this domain, Process Mining (PM) has emerged as a promising field for extracting meaningful insights from logs generated by real-world systems [2, 3]. This is achieved through the three main tasks of PM: discovery, conformance checking and enhancement. At the core of these tasks lies the concept of a process model, a formal representation of the behavioral logic of a business process, which specifies the admissible sequences of activities. Traditional process models follow two main approaches: procedural and declarative. Procedural models (like BPMN [4], Petri nets [5]) explicitly prescribe flows and adopt a closed approach where only explicitly modeled behaviors are allowed. Declarative models [6], on the contrary, specify constraints and adopt an open approach where anything not explicitly forbidden is permitted; an example is: "the activity *register order* always takes place before the activity *approve order*". DECLARE [7] and DCR Graphs [8] are the most notable examples of declarative languages for process modelling, with the former being grounded on temporal logic, thus coming with a formal semantics.

While these approaches have proven effective in many scenarios, they often do not adequately handle the *uncertainty* intrinsic to real-world domains. On one side, logs are just a partial incomplete view of the reality; on the other side, the information in the log might be incomplete, partially specified, and even non reliable. This has driven recent research towards integrating probability within PM techniques at different levels: [9, 10, 11] address probabilistic traces and event data in procedural PM. [12] introduce the ProbDeclare framework, where constraints are uncertain: uncertainty is characterized through a frequentist notion of probability based on the ratio of traces in a log that are expected to satisfy the constraint. This work is extended in [13], where a formal semantics to ProbDeclare is given and the

CILC 2025: 40th Italian Conference on Computational Logic, June 25–27, 2025, Alghero, Italy

*Corresponding author.

michela.vespa@unife.it (M. Vespa); elena.bellodi@unife.it (E. Bellodi)

0009-0004-4350-8151 (M. Vespa); 0000-0002-3717-3779 (E. Bellodi)

authors explain how probabilistic constraints can be discovered from event data by means of existing algorithms for declarative process discovery.

We follow a different route with respect to those works, by taking advantage of the developments in the field of Probabilistic Logic Programming (PLP) in order to address uncertainty in the tasks of *declarative* conformance checking (or compliance), process discovery, and model selection. Conformance checking is the task of verifying whether a process execution complies with a predefined model. Process discovery envisages the discovery of a (procedural or declarative) process model guided by the traces that are recorded into an input log. Model selection is the identification of a preferable model in case there are multiple output models from the process discovery task. In our case, uncertainty is quantified by a probability value attached to constraints, process traces or events, telling how strong/important a constraint is or the degree of our belief in a specific trace or event happening.

In this paper we focus on probabilistic declarative compliance, leaving the other two tasks for future work. The overall research is performed in the context of the PRIN2022 project "Probabilistic declarative process mining (PRODE)"[1]. The project will build a set of techniques that target the issues above by means of new combinations of declarative Process Mining with *probabilistic* and *combinatorial* approaches. We illustrate our recent work on probabilistic conformance checking based on PLP under the Distribution Semantics (Section 3), where we defined probabilistic declarative process models, probabilistic traces and probabilistic logs. Then, in Section 4, we present our current efforts towards handling uncertainty both in process models and in process traces. Lastly, we identify future research directions in Section 5.

2. Preliminaries

2.1. Process Mining

In Process Mining a *trace* or process instance represents a distinct execution of a process, potentially repeated multiple times. A trace typically consists of a sequence of activity executions, each identified by a distinct name and associated with temporal information that defines their order.

Definition 1 (Trace t and Log $\mathcal{L}$)**.** *Given a finite set $\mathcal{A}$ of symbols (i.e., activity names), a* trace t *is a finite, ordered sequence of symbols over $\mathcal{A}$, i.e. $t \in \mathcal{A}^*$, where $\mathcal{A}^*$ is the infinite set of all the possible finite sentences over $\mathcal{A}$. A* log $\mathcal{L}$ *is a finite set of traces.*

For example, a trace can be represented as an ordered sequence $t = \langle a, b, c, d \rangle$, where the timestamps of activities a, b, c, d satisfy $timestamp_a < timestamp_b < \ldots < timestamp_d$ to reflect the execution order.

Example 1. *Consider a process with activity set $\mathcal{A} = \{a, b, c, d\}$. An example log $\mathcal{L}$ over $\mathcal{A}$ may contain the following traces:*

$$\mathcal{L} = \{t_1 = \langle a, b, c \rangle, \quad t_2 = \langle a, b, a, d \rangle, \quad t_3 = \langle a, a, d \rangle, \quad t_4 = \langle a, b, c \rangle\}$$

Declarative Process Mining (DPM) is a subfield of PM that emphasizes flexibility over rigid procedural workflows. Instead of defining exact execution paths, DPM specifies what must or must not happen during a process execution through *constraints*.

Our work starts from the most representative declarative modeling formalism, DECLARE [7], which provides a set of graphical constraint templates with a formal semantics given by LTL_f logic [14]. Examples of such constraints are "*activity* a *must eventually be followed by* b" (which is called *response(a,b)*) or "activity *a* must be the first executed activity" (which is called *init(a)*). The semantics exploits the idea that each DECLARE template can be mapped onto one (or more) logical formula φ, and that *logical entailment* can be used to define the notion of *compliance* of a trace t w.r.t. to a constraint formula φ.

[1] https://prode.unife.it/

Definition 2 (Compliance of a Trace to a Constraint)**.** *A trace t is compliant with a DECLARE constraint if it satisfies the corresponding logical formula φ, denoted as $t \models \varphi$. Conversely, if $t \not\models \varphi$, we say t violates the constraint.*

In the following, we extend this notion with respect to a set of constraints, i.e. a process model, that we formally call Declarative Process Specification.

Definition 3 (Declarative Process Specification [15])**.** *A Declarative Process Specification (DS) is a triple $DS = (T, \mathcal{A}, C)$, where:*

- *T is a finite set of constraint templates, where each template is a predicate $c(x_1, \ldots, x_m) \in T$ on variables $x_1, \ldots, x_m$ (with $m \in \mathbb{N}$ the arity of c);*
- *$\mathcal{A}$ is a finite set of activity names;*
- *C is a finite set of constraints instantiated from T over $\mathcal{A}$. We will denote such constraints with $c(a_1, \ldots, a_m)$, $a_1, \ldots, a_m \in \mathcal{A}$.*

Definition 4 (Compliance of a trace versus a Declarative Process Specification)**.** *A trace is compliant with a DS if it entails the conjunction of the formulas φ_i corresponding to the $c_i \in C$: $t \models \varphi_1 \wedge \ldots \wedge \varphi_n$ where n is the cardinality of C.*

2.2. Probabilistic Logic Programming

Probabilistic Logic Programming (PLP) integrates LP with probability theory to handle uncertain domains. The *Distribution Semantics* [16] is a widely adopted formalism for PLP, originally introduced for PRISM and later adopted by many other languages. To describe the semantics we use as a reference language the one called "Logic Programs with Annotated Disjunctions" (LPADs) [17].

Definition 5. *A LPAD consists of annotated disjunctive clauses D_i of the form:*

$$h_{i1} : p_{i1}; \ldots; h_{in_i} : p_{in_i} \leftarrow b_{j1}, \ldots, b_{jm_j},$$

where each h_{in_i} is an atom, each b_{jm_j} is a literal, and $p_i \in [0, 1]$ are probabilities satisfying $\sum_{k=1}^{n_i} p_{ik} \leq 1$.

$b_{i1}, \ldots, b_{im_i}$ are literals and are indicated with $body(D_i)$. If $\sum_{k=1}^{n_i} p_{ik} < 1$, the head implicitly contains an extra atom $null$ that does not appear in the body of any clause and whose annotation is $1 - \sum_{k=1}^{n_i} p_{ik}$. Each clause represents a probabilistic choice among one of the head atoms, given that $body(D_i)$ holds. A LPAD program defines a probability distribution over normal logic programs called *worlds*. A survey of the distribution semantics in PLP can be found in [18].

We consider here ground LPADs, and we denote by $ground(L)$ the grounding of an LPAD L. An *atomic choice* [19] is a triple (D_i, θ_j, k) where $D_i \in L$, θ_j is a substitution that grounds D_i and $k \in \{1, \ldots, n_i\}$ identifies one of the head atoms. (D_i, θ_j, k) means that, for the ground clause $D_i\theta_j$, the head h_{ik} was chosen. A set of atomic choices κ is *consistent* if only one head is selected from the same ground clause; we assume independence between the different choices. A *composite choice* κ is a consistent set of atomic choices [19]. The *probability* $P(\kappa)$ *of a composite choice* κ is the product of the probabilities of the independent atomic choices, i.e. $P(\kappa) = \prod_{(D_i,\theta_j,k)\in\kappa} p_{ik}$. A *selection* σ is a composite choice that, for each clause $D_i\theta_j$ in $ground(L)$, contains an atomic choice (D_i, θ_j, k). Let us indicate with S_L the set of all selections. A selection σ identifies a normal logic program w_σ defined as $w_\sigma = \{(h_{ik} \leftarrow body(D_i))\theta_j | (D_i, \theta_j, k) \in \sigma\}$. w_σ is called a (possible) *world* of L. Since selections are composite choices, we can assign a probability to worlds: $P(w_\sigma) = P(\sigma) = \prod_{(D_i,\theta_j,k)\in\sigma} p_{ik}$.

We denote the set of all worlds of L by W_L. $P(W_L)$ is a probability distribution over worlds, i.e., $\sum_{w\in W_L} P(w) = 1$. A composite choice κ identifies a set of worlds $w_\kappa = \{w_\sigma | \sigma \in S_L, \sigma \supseteq \kappa\}$. The set of possible worlds associated to a set of composite choices K is $W_K = \bigcup_{\kappa\in K} w_\kappa$.

3. Probabilistic Event Data and Process Specifications

In this section we recap the results obtained so far in our recent works. All of them take inspiration from the Distribution Semantics and allow to *separately* manage uncertainty at different levels: events, traces, logs and process constraints.

Definition 6 (Probabilistic Event [20])**.** *A* Probabilistic Event *is a couple Prob:EventDescription, where EventDescription is a symbol describing an event (EventDescription* $\in \mathcal{A}$*), while* Prob $\in [0, 1]$ *is the probability that the event happened. A probability value of 1 means the event happened, and we will refer to it as "certain". Otherwise it represents the degree of our belief in the event happening.*

Definition 7 (Probabilistic Trace [20])**.** *A* Probabilistic Trace *is a trace where at least one event is probabilistic.*

Example 2. *The trace:* $t = \langle 0.9 :$ *register_order*, *approve_order*, *schedule_delivery*, *invoice_customer*$\rangle$ *describes the situation where* $register\_order$ *was not logged, however it is very probable that it happened due to the standard process (the associated probability is high).* $register\_order$ *is a probabilistic event, and* t *is a probabilistic trace.*

Definition 8 (Probabilistic Log [21])**.** *A* probabilistic log $\mathcal{L}_p$ *is a log where at least one trace* t *is annotated with a probability* p*. A probability value of 1 means the trace certainly happened and the value will be omitted.*

Example 3. *The probabilistic log* $\mathcal{L}_p = \{t_1, 0.9 : t_2, t_3\}$ *describes the case in which the process instances* t_1 *and* t_3 *were observed and recorded, while* t_2 *was not observed but there is a high probability (0.9) that it happened.*

Definition 9 (Probabilistic Declarative Specification [22])**.** *A* Probabilistic Declarative Process Specification *PDS is a Declarative Process Specification DS where each constraint* $c_i \in DS$ *is a probabilistic constraint.*

Example 4. *The following PDS:*

$$C = \{ \quad 0.8 :: \textit{response(register\_order, approve\_order)} \qquad (c_1) \quad \}$$

includes one probabilistic constraint c_1 *which indicates that the fact that* $register\_order$ *is potentially followed by* $approve\_order$ *carries relatively high importance in the business process.*

We will refer to constraints annotated with probability $p_i = 1$ as crisp constraints. When every constraint is crisp, the result is essentially a Declarative Specification. By taking inspiration from the Distribution semantics, a PDS defines a probability distribution over regular (non-probabilistic) DSs that correspond to worlds: c_i may be chosen to be included in a world with probability p_i, or not with probability $1 - p_i$. In this way, we can assign a probability to each DS, that we indicate with $P(DS)$, given by the product of the probabilities of the selected probabilistic constraints c_i and the complement of the probabilities of the excluded constraints. This corresponds to a selection σ as defined in subsection 2.2. The probability distribution over DSs guarantees that $\sum_i P(DS_i) = 1$.

In [22] we introduced how to compute the compliance of a *certain* trace versus a PDS:

Definition 10 (Compliance of a trace versus a PDS [22])**.** *Given a PDS, the probability of compliance of a trace* t *w.r.t. a PDS is defined as:*

$$Comp(t, PDS) = \sum_{\sigma_i : t \models DS_i} P(DS_i), \tag{1}$$

where DS_i *represents each deterministic specification induced by the selections* σ_i *over the PDS, and* $P(DS_i)$ *is the probability associated with each deterministic specification.*

Example 5. *Let us consider the following PDS:*

$$C = \{ \quad 0.8 :: \textit{response(register\_order, approve\_order)} \quad (c_1)$$
$$0.9 :: \textit{init(register\_order)} \quad (c_2) \quad \}$$

Such a PDS leads to 4 selections and 4 possible worlds; so, 4 different regular DSs are possible, each one corresponding to a world, as shown in Table 1.

Consider the trace $t = \langle \textit{receive\_order\_request}, \textit{register\_order}, \textit{approve\_order} \rangle$. t *is compliant with* DS_2 *and* DS_4*, i.e. those specifications that do not contain* c_2*, since the first event of the trace is not* $register\_order$*.* t*'s probability of compliance is* $Comp(t, PDS) = P(DS_2) + P(DS_4) = 0.08 + 0.02 = 0.1$.

Selection	DS	$P(DS_i)$
σ_1	$DS_1 = \{$response(register_order, approve_order), init(register_order)$\}$	$P(DS_1) = 0.8 \times 0.9 = 0.72$
σ_2	$DS_2 = \{$response(register_order, approve_order)$\}$	$P(DS_2) = 0.8 \times 0.1 = 0.08$
σ_3	$DS_3 = \{$init(register_order)$\}$	$P(DS_3) = 0.2 \times 0.9 = 0.18$
σ_4	$DS_4 = \{ \quad \}$	$P(DS_4) = 0.1 \times 0.2 = 0.02$

Table 1
Declarative Process Specifications generated by the PDS of Example 5.

4. Probabilistic Compliance of Uncertain Traces

In this section we propose our current work regarding the possibility to define the compliance of a probabilistic trace with respect to a set of probabilistic constraints (a PDS). Recalling the Distribution Semantics, for each probabilistic event in a probabilistic trace we can make an atomic choice, which determines whether a probabilistic event appears or not in the trace, as follows:

Definition 11 (Atomic choice, Composite choice and Selection [20])**.** *An atomic choice is a pair* $(\textit{EventDescription}_i, k)$ *where* $\textit{EventDescription}_i$ *is a probabilistic event appearing in the* i*-th position in a probabilistic trace and* $k \in \{0, 1\}$. k *indicates whether* $\textit{EventDescription}_i$ *is chosen to be included in a world with probability* p_i *(k=1), or not with probability* $1 - p_i$ *(k=0).*

A Composite choice $\kappa(t)$ *is a consistent set of atomic choices over probabilistic events in* t*. The probability of a composite choice is* $P(\kappa(t)) = \prod_{(EventDescription_i, 1) \in \kappa} p_i \prod_{(EventDescription_i, 0) \in \kappa} (1 - p_i)$*, where* p_i *is the probability associated with* $\textit{EventDescription}_i$*.*

A Selection $\sigma(t)$ *over a probabilistic trace* t *is a composite choice containing an atomic choice* $(\textit{EventDescription}_i, k)$ *for each probabilistic event in* t*. A selection* $\sigma(t)$ *identifies a world* $w_\sigma(t)$ *in this way:* $w_\sigma(t) = \{EventDescription_i | (EventDescription_i, 1) \in \sigma(t)\}$.

Example 6. *Given the trace in Example 2, the possible selections are*

$$\sigma_1(t) = \{(\textit{register\_order}, 1)\}, \sigma_2(t) = \{(\textit{register\_order}, 0)\}$$

The corresponding possible worlds are:

$$w_{\sigma_1}(t) = \langle \textit{register\_order}, \textit{approve\_order}, \textit{schedule\_delivery}, \textit{invoice\_customer} \rangle$$

$$w_{\sigma_2}(t) = \langle \textit{approve\_order}, \textit{schedule\_delivery}, \textit{invoice\_customer} \rangle$$

Note that we ended up with a set of regular (non-probabilistic) traces, that correspond to the worlds $w_{\sigma_i}(t)$.

Definition 12 (Probability of a Selection [20])**.** *The probability of a selection $\sigma(t)$ is defined as:*

$$P(\sigma(t)) = \prod_{(EventDescription_i,1)\in\sigma(t)} p_i \prod_{(EventDescription_i,0)\in\sigma(t)} (1-p_i)$$

The probability of a selection corresponds to the probability of a world $w_\sigma(t)$, i.e. $P(w_\sigma(t)) = P(\sigma(t))$.

Example 7. *Given the trace in Example 2, the probabilities of the two corresponding worlds in Example 6 are: $P(w_{\sigma_1}(t)) = 0.9$ and $P(w_{\sigma_2}(t)) = 0.1$.*

Now we introduce our idea of compliance of a probabilistic trace w.r.t. a Probabilistic Declarative Process Specification. Given a PDS and the notion of compliance of a (certain) trace w.r.t. a PDS, as per Definition 10, we can extend this notion to a probabilistic trace t by considering the compliance of each world $w_\sigma(t)$ generated by the trace versus each Declarative Process Specification DS generated by the PDS. The probability of compliance of t w.r.t. the PDS will be calculated by summing up the products between the probability of the world compliant with a DS and the probability of the DS itself.

Definition 13 (Compliance of a probabilistic trace versus a PDS)**.** *Given a a Probabilistic Declarative Process Specification PDS and a Probabilistic Trace t, let us consider all the possible selections σ_i over the PDS and all the possible selections $\sigma_i(t)$ over t. Let us consider all the possible Declarative Specifications DS_i associated with σ_i and all the possible worlds $w_{\sigma_i}(t)$ associated with $\sigma_i(t)$.*

We define the compliance Comp(t,PDS) of a probabilistic trace t w.r.t. PDS as:

$$Comp(t, PDS) = \sum_{\substack{w_{\sigma_i}(t) \\ DS_i}} \begin{cases} P(w_\sigma(t)) \cdot P(DS_i) & \text{if } w_{\sigma_i}(t) \text{ is compliant with } DS_i, \\ 0 & \text{otherwise.} \end{cases}$$

Example 8. *Consider the PDS from Example 5 and the probabilistic trace from Example 2. The total number of combinations is $4 \times 2 = 8$, corresponding to the 4 DSs of Table 1 multiplied by the 2 worlds $w_{\sigma_1}(t), w_{\sigma_2}(t)$ of Example 6, as shown in Table 2.*

Trace	DS	Compliance	Probability
register_order, approve_order, schedule_delivery, invoice_customer	DS_1	True	$0.9 \times 0.72 = 0.648$
register_order, approve_order, schedule_delivery, invoice_customer	DS_2	True	$0.9 \times 0.08 = 0.072$
register_order, approve_order, schedule_delivery, invoice_customer	DS_3	True	$0.9 \times 0.18 = 0.162$
register_order, approve_order, schedule_delivery, invoice_customer	DS_4	True	$0.9 \times 0.02 = 0.018$
approve_order, schedule_delivery, invoice_customer	DS_1	False	$0.1 \times 0.72 = 0.072$
approve_order, schedule_delivery, invoice_customer	DS_2	False	$0.1 \times 0.08 = 0.008$
approve_order, schedule_delivery, invoice_customer	DS_3	False	$0.1 \times 0.18 = 0.018$
approve_order, schedule_delivery, invoice_customer	DS_4	True	$0.1 \times 0.02 = 0.002$

Table 2
Compliance of the probabilistic trace of Ex. 2 w.r.t. the PDS of Example 5. The "compliance" column indicates if the given trace is compliant w.r.t the given DS. The last column computes the product of the probability of every $w_{\sigma_i}(t)$ and every $P(DS_i)$.

The probability of compliance of t is computed by summing the probabilities of the worlds where each non-probabilistic trace is compliant with a DS: $Comp(t, PDS) = 0.648 + 0.072 + 0.162 + 0.018 + 0.002 = 0.902$.

Compliance is not a binary outcome but a weighted evaluation over all possible DSs weighted by the probability of the world generated by a probabilistic trace.

4.1. Preliminary Experimental Results

We used the same set-up presented in [22] both in terms of the algorithm [23] and the case study, based on the ERAS® [24] colorectal-surgery protocol, from which we took inspiration for building a

process model with 21 constraints and a trace of 21 events representing a patient. All experiments were executed on a Linux machine equipped with two AMD® EPYC 9124 16-core CPUs and a 60 GB Prolog stack, with a 24-hour timeout per run. For each iteration k (with $k = 1$ to 21), we built a test case by randomly selecting k constraints in the PDS to be *probabilistic* (with user-defined probabilities), while treating the remaining $21 - k$ constraints as crisp. Simultaneously, we consider k events in the trace as probabilistic (with a user-defined value), while keeping the remaining $21 - k$ as certain. Table 3 shows execution times for computing the probability of compliance for the different configurations of the experiments.

Experiment	# of prob. constraints	# of prob. events	Time (s)
1	1	1	793.96
2	2	2	1450.72
3	3	3	3503.28
4	4	4	7896.75
5	5	5	17751.73
6	6	6	44133.15
7	7	7	T.O.

Table 3
Execution time (in seconds) for computing the probability of compliance of a trace with an increasing number of probabilistic events versus an increasing number of probabilistic constraints (from 1 to 7). T.O. indicates a time-out.

As expected, enumerating all possible combinations of regular traces versus regular DS leads to an exponential trend in execution times as the number of probabilistic constraints together with the number of probabilistic events increases. Currently we are studying how to perform an efficient computation of the compliance without relying on this solution.

5. Conclusion and Future Work

In this paper we have presented our work in progress to evaluate compliance in declarative process mining when uncertain information may affect both event logs and process models. Ongoing work focuses on implementing a scalable solution for this task.

Future works include:

- exploiting the availability of positive and negative examples for process discovery: in many cases, user experts provide traces with desired (positive) and undesired (negative) behaviour, but the majority of the discovery approaches exploits only the positive set;
- developing discovery algorithms for learning declarative process models in uncertain domains, in order to automatically associate probabilities to constraints;
- identifying preferable models for the user when performing the process discovery task, which could output multiple models.

The project will build a set of techniques that target the issues above by means of new combinations of declarative Process Mining with probabilistic and combinatorial approaches. The final aim is to produce more verifiable and understandable explanations of its processes to an organization.

Acknowledgments

Finanziato dall'Unione europea NextGenerationEU | Ministero dell'Università e della Ricerca | Italiadomani | Università degli Studi di Ferrara

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 author(s) have not employed any Generative AI tools.

References

[1] T. Slaats, Declarative and hybrid process discovery: Recent advances and open challenges, Journal on Data Semantics 9 (2020) 3–20. doi:10.1007/s13740-020-00112-9.

[2] van der Aalst et al., Process mining manifesto, in: F. Daniel, K. Barkaoui, S. Dustdar (Eds.), Business Process Management Workshops - BPM 2011 International Workshops, Clermont-Ferrand, France, August 29, 2011, Revised Selected Papers, Part I, volume 99 of *Lecture Notes in Business Information Processing*, Springer, 2011, pp. 169–194. doi:10.1007/978-3-642-28108-2_19.

[3] W. van der Aalst, Process Mining: Data Science in Action, 2nd ed., Springer Publishing Company, Incorporated, 2016.

[4] O. M. Group, Business Process Model and Notation (BPMN), Version 2.0, 2011. URL: https://www.bpmn.org/.

[5] 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 - BPM 2013 International Workshops, Beijing, China, August 26, 2013, Revised Papers, volume 171 of *Lecture Notes in Business Information Processing*, Springer, 2013, pp. 15–27.

[6] W. M. P. van der Aalst, M. Pesic, H. Schonenberg, Declarative workflows: Balancing between flexibility and support, Computer Science - Research and Development 23 (2009) 99–113.

[7] 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.

[8] T. T. Hildebrandt, R. R. Mukkamala, Declarative event-based workflow as distributed dynamic condition response graphs, in: K. Honda, A. Mycroft (Eds.), Proceedings of the Third Workshop on Programming Language Approaches to Concurrency and communication-cEntric Software, PLACES 2010, Paphos, Cyprus, 21st March 2010, volume 69 of *EPTCS*, 2010, pp. 59–73. doi:10.4204/EPTCS.69.5.

[9] M. Pegoraro, W. M. P. van der Aalst, Mining uncertain event data in process mining, in: International Conference on Process Mining, ICPM 2019, Aachen, Germany, June 24-26, 2019, IEEE, 2019, pp. 89–96.

[10] M. Pegoraro, M. S. Uysal, W. M. P. van der Aalst, Discovering process models from uncertain event data, in: C. D. Francescomarino, R. M. Dijkman, U. Zdun (Eds.), Business Process Management Workshops - BPM 2019 International Workshops, Vienna, Austria, September 1-6, 2019, Revised Selected Papers, volume 362 of *Lecture Notes in Business Information Processing*, Springer, 2019, pp. 238–249.

[11] S. J. J. Leemans, W. M. P. van der Aalst, T. Brockhoff, A. Polyvyanyy, Stochastic process mining: Earth movers' stochastic conformance, Inf. Syst. 102 (2021) 101724.

[12] F. M. Maggi, M. Montali, R. Peñaloza, A. Alman, Extending temporal business constraints with uncertainty, in: D. Fahland, C. Ghidini, J. Becker, M. Dumas (Eds.), Business Process Management - 18th International Conference, BPM 2020, Seville, Spain, September 13-18, 2020, Proceedings, volume 12168 of *Lecture Notes in Computer Science*, Springer, 2020, pp. 35–54.

[13] A. Alman, F. M. Maggi, M. Montali, R. Peñaloza, Probabilistic declarative process mining, Inf. Syst. 109 (2022) 102033. URL: https://doi.org/10.1016/j.is.2022.102033. doi:10.1016/J.IS.2022.102033.

[14] 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.

[15] 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.

[16] 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.

[17] J. Vennekens, S. Verbaeten, M. Bruynooghe, Logic programs with annotated disjunctions, in: B. Demoen, V. Lifschitz (Eds.), 20th International Conference on Logic Programming (ICLP 2004), volume 3131 of *LNCS*, Springer, 2004, pp. 431–445. doi:10.1007/978-3-540-27775-0_30.

[18] E. Bellodi, The distribution semantics in probabilistic logic programming and probabilistic description logics: a survey, Intelligenza Artificiale 17 (2023) 143 – 156. doi:10.3233/IA-221072.

[19] D. Poole, The Independent Choice Logic for modelling multiple agents under uncertainty, Artificial Intelligence 94 (1997) 7–56.

[20] M. Vespa, E. Bellodi, F. Chesani, D. Loreti, P. Mello, E. Lamma, A. Ciampolini, M. Gavanelli, R. Zese, Probabilistic traces in declarative process mining, in: A. Artale, G. Cortellessa, M. Montali (Eds.), AIxIA 2024 - Advances in Artificial Intelligence - XXIIIrd International Conference of the Italian Association for Artificial Intelligence, AIxIA 2024, Bolzano, Italy, November 25–28, 2024 Proceedings, volume 15450 of *Lecture Notes in Computer Science*, Springer, 2024, pp. 330–345.

[21] M. Vespa, A probabilistic semantics for process mining, in: D. Bacciu, I. Donadello (Eds.), Proceedings of the AIxIA Doctoral Consortium 2024 co-located with the 23rd International Conference of the Italian Association for Artificial Intelligence (AIxIA 2024), volume 3914, CEUR-WS, 2024, pp. 1–6.

[22] 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.

[23] E. Bellodi, M. Gavanelli, R. Zese, E. Lamma, F. Riguzzi, Nonground abductive logic programming with probabilistic integrity constraints, Theory and Practice of Logic Programming 21 (2021) 557–574. doi:10.1017/S1471068421000417.

[24] 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.