Probabilistic Traces in Declarative Process Mining

Michela Vespa[1][0009−0004−4350−8151], Elena Bellodi[1][0000−0002−3717−3779], Federico Chesani[2][0000−0003−1664−9632], Daniela Loreti[2][0000−0002−6507−7565], Paola Mello[2][0000−0002−5929−8193], Evelina Lamma[1][0000−0003−2747−4292], Anna Ciampolini[2][0000−0002−9314−1958], Marco Gavanelli[1][0000−0001−7433−5899], and Riccardo Zese[3][0000−0001−8352−6304]

[1] Dipartimento di Ingegneria, Università di Ferrara, Via Saragat 1, Ferrara, Italy {firstname.lastname}@unife.it
[2] Dipartimento di Informatica - Scienza e Ingegneria, Viale Risorgimento 2, Bologna, Italy {firstname.lastname}@unibo.it
[3] Dipartimento di Scienze Chimiche, Farmaceutiche ed Agrarie - Università di Ferrara, Via Borsari 46, Ferrara, Italy {firstname.lastname}@unife.it

Abstract. When dealing with real-world processes, it is essential to consider their inherent uncertainty to more accurately represent their nature. In this work, we consider cases in which some information in the log might be unreliable. We propose a novel semantics for probabilistic process traces, based on the Distribution Semantics from Probabilistic Logic Programming, which allows one to annotate event executions of an observed trace with a probability representing the uncertainty of the event as the degree of our belief in that event happening. Then, we propose a novel definition of probabilistic compliance of a probabilistic process trace w.r.t. a declarative process specification, and how to compute it using a probabilistic abduction proof-procedure. Experimental results on a real-world healthcare protocol are presented to evaluate the feasibility of the proposed semantics on conformance checking.

Keywords: Process Mining · Probabilistic compliance · Declarative language · Distribution Semantics

1 Introduction

In the Business Process Mining (BPM) community [24], the integration of probability into various aspects of process modeling is a growing research topic, reflecting an interest in enhancing process analysis under conditions of uncertainty in real-world domains. Uncertainty can be found at various levels, such as process constraints, events, event attributes, or traces.

For instance, in declarative process mining, [2] introduced the notion of probabilistic process constraints, by associating probabilities to DECLARE constraints with a frequency-based approach. A probabilistic constraint is satisfied over the log if the number of traces satisfying the constraint over the log cardinality achieves the mathematical relation established by the relational operator and the probability assigned. A constraint c_1 with assigned probability $(0.6, \geq)$ is satisfied if the number of traces complaint with c_1 is at least $\geq 60\%$ of the total traces. They propose how to discover such constraints and how to carry out monitoring and conformance checking.

In procedural process mining, [6], [15] and [14] handle probabilities at the level of traces or events in a trace. In [6], traces are supposed generated by a stochastic process model, which outputs a variety of possible sequences, each associated with a certain probability. Then, probabilistic trace alignment - i.e. the comparison between the model traces and the observed trace - is done by identifying the k model traces nearest to a particular observed log trace. [15] defines probability distributions over all possible realizations of an uncertain trace, where uncertainty may come from

partial timestamps, indeterminate events, and activity labels. Uncertainty is divided into two types: strong uncertainty, where the probability distributions of events are unknown, and weak uncertainty, where events follow known probability distributions. The conformance score is calculated as the cost of the optimal alignment between the trace and a Petri net model. [14] consider the analysis of a specific class of event logs, those containing uncertain events, i.e. recordings of executions of specific activities in a process which are enclosed with an indication of uncertainty in the event attributes. Specifically, attributes of an event are not recorded as a precise value but as a range or a set of alternatives, and they perform conformance checking by assessing upper and lower bounds on the conformance score for possible values of the attributes in an uncertain trace.

Differently from previous work, we propose a novel semantics to handle uncertainty at the level of *events* in a trace in a *declarative* process mining setting. In fact, some information in a process trace could be unreliable and a domain expert might be allowed to indicate a *degree of belief* on that information. Such degree is expressed by an *epistemic probability* value [20] which may be associated to an event execution. To better explain the idea, we provide an example based on a real medical protocol about elective colorectal surgery [23]. In the ERAS® (Enhanced Recovery After Surgery) guidelines, the administration of preoperative antibiotics is crucial for reducing infection risks. However, there might be cases where this administration is not logged correctly due to human error or system issues. For instance, if a surgeon recalls ordering antibiotics for a patient before surgery but later finds no documentation of this in the medical records, she might estimate, based on routine practice, interactions with the nursing staff, and protocol recommendations, that there was a 95% probability the antibiotics were administered as needed. This estimation is not based on quantitative data but rather on the surgeon's *confidence/belief* in her standard procedures.

From a formal point of view, we introduce the concept of Probabilistic Event and Probabilistic Trace starting from Probabilistic Logic Programming (PLP) and the Distribution Semantics [22]. In a PLP program, logic formulas are annotated with probabilities, defining a probability distribution over normal logic programs called worlds. These probabilities represent the likelihood of the formulas appearing in a normal logic program. Likewise, the probability of an event is treated as the probability that such event will appear (or not) in a world, similarly to how scenarios (or realizations) are treated in [15]. However, we consider uncertainty related to the event as a whole, without considering event attributes. According to our semantics, all worlds including one such event "inherit" its probability, while those not including the event take into account the complement to 1 of its probability. We also tackle the problem of probabilistic conformance checking of a probabilistic trace with DECLARE constraints: the conformance score is defined as the sum of the probabilities of the worlds where the trace is compliant.

This work makes the following contributions: we define a semantics for probabilistic events and traces in the setting of declarative process specifications, introduce a new concept of compliance, and show their application to the ERAS® protocol. Then, we exploit previous results, in particular (*i*) the mapping provided to DECLARE in terms of the SCIFF modeling language [13] and (*ii*) the extension of the Distribution Semantics to Abductive Logic Programming (ALP) in the SCIFF framework [3], in order to implement our proposed semantics and compute a compliance probability.

Experiments show that the approach is able to manage a large number of DECLARE constraints and probabilistic events.

The paper is organized as follows: Section 2 introduces background on declarative languages and the distribution semantics, Section 3 presents the proposed semantics, Section 4 describes how we perform probabilistic compliance. Section 5 describes experimental results and Section 6 concludes the paper.

2 Background

2.1 Process Traces and DECLARE

In the BPM context, the starting point is observing a process execution through the activities that constitute the process. Each execution, or *process instance*, is referred to as a *trace*, with each activity execution known as an *activity instance*. Activities are identified by names and typically marked with a temporal timestamp, establishing an order within the same trace. Traces are often represented as sequences of activity names, ordered by their timestamps. However, depending on the context, timestamps can be omitted, as is the case in our approach.

Formally, we assume a finite alphabet of symbols Σ representing the activity names. A trace and a log then are defined as follows:

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

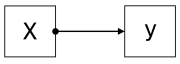
In a log of the process, each trace t represents a different process instance. Different process instances may have the same trace, although referring to different executions. It might make sense to count the number of appearances of a specific trace. Formally:

Definition 2 ($\mathcal{L}(\cdot)$ function). *With an abuse of notation, $\mathcal{L}(\cdot)$ will be used to indicate also a function that counts the occurrences of a trace t in the log $\mathcal{L}$.*

Example 1. Let us suppose that a process is made of activities a, b, c, and d. $\Sigma = \{\mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{d}\}$. An example of a log might be: $\mathcal{L} = \{t_1 = \langle \mathsf{a}, \mathsf{b}, \mathsf{c}\rangle, t_2 = \langle \mathsf{a}, \mathsf{b}, \mathsf{a}, \mathsf{d}\rangle, t_3 = \langle \mathsf{a}, \mathsf{a}, \mathsf{d}\rangle, t_4 = \langle \mathsf{a}, \mathsf{b}, \mathsf{c}\rangle\}$.

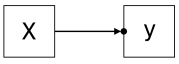
It holds that: $\mathcal{L}(t_1) = \mathcal{L}(t_4) = 2, \mathcal{L}(t_2) = 1, \mathcal{L}(t_3) = 1$.

The DECLARE modeling language, introduced by [16], counters the inflexibility of procedural models by focusing on defining key process properties rather than specific execution steps. It models processes through *constraints*, rules governing the activities within a trace with qualitative temporal relations among the activities. For instance, the constraint response(a, b) mandates that each instance of activity a be eventually followed by b, focusing on what must happen *after* a in a forward-looking way. DECLARE offers a variety of constraint patterns, such as response(x, y), init(x), requiring every process to start with activity x, and precedence(x, y), ensuring that an activity x has occurred *before* an activity y, in a backward-looking way. DECLARE also provides a graphical representation for these patterns (see Figure 1) and is based on two different logic formal semantics.



(a) The response template.

(b) The init template.



(c) The precedence template.

Fig. 1: Examples of the DECLARE graphical notation: x and y are placeholders that should be substituted with proper activity names.

In the original formulation in [16] the semantics was given using the *LTL* temporal logic; subsequent works have shown the feasibility of adopting the LTL_f logic [12]: for a recent recap see [7].

A second formal semantics has been proposed in [13], where the SCIFF language and ALP [1] has been exploited. Both the semantics exploit the idea that each DECLARE template can be mapped onto one or more logic formulas φ, and that logical entailment can be used to define the notion of *compliance/violation* of a trace t w.r.t. a constraint formula φ. We provide an intuitive definition of compliance/violation, where the meaning of the entailment symbol $\models$ should be referred to the chosen semantics (LTL_f or ALP).

Definition 3 (Compliance/violation of a trace versus a constraint). *A trace t is* compliant *with a DECLARE constraint if, named φ the corresponding logic formula modelling that constraint, it holds: $t \models \varphi$.*

A trace t violates *a DECLARE constraint if it does not entail the corresponding formula φ, i.e. if $t \not\models \varphi$.*

Definition 4 (Declarative Process Specification, from [7]). *A* declarative process specification *is a tuple DS=(REP, Σ, C) where:*

- *REP is a finite non-empty set of* templates*, where each* template *is a predicate $\text{C}(x_1, \ldots, x_m) \in \text{REP}$ on variables $x_1, \ldots, x_m$ (with $m \in \mathbb{N}$ the arity of C);*
- *Σ is a finite non-empty set of* activity names*;*
- *C is a finite set of* constraints*, obtained by instantiating templates from REP to Σ; we will compactly denote such constraints with $\text{C}(a_1, \ldots, a_m)$, $a_1, \ldots, a_m \in \Sigma$.*

Usually the constraints $\text{C}(a_1, \ldots, a_m)$ in a DS are considered as being in *logical conjunction*. The notion of compliance can be then lifted from a trace vs. a constraint towards a trace vs. a DS as follows:

Definition 5 (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 $c_i \in C$:*

$$t \models \varphi_1 \wedge \ldots \wedge \varphi_m$$

where m is the cardinality of C.

Example 2. Let us consider the log introduced in Example 1, and the following DS (REP and Σ are omitted for the sake of simplicity):

$$C = \{ \quad \begin{array}{l} \text{C}_1 = \mathsf{response(a,b)} \\ \text{C}_2 = \mathsf{init(a)} \end{array} \quad \}$$

Even without considering the corresponding formal semantics, we can notice that t_1 and t_4 are compliant with C_1; t_2 is not compliant with C_1 because the second occurrence of activity a is not followed by an occurrence of activity b; t_3 is not compliant with C_1 because two occurrences of a are not followed by an occurrence of b; $t_1, \ldots, t_4$ are all compliant with C_2.

2.2 Distribution Semantics and Probabilistic Logic Programming

Probabilistic Logic Programming (PLP) has recently garnered increasing attention due to its capability to incorporate probability into logic programming, thereby handling uncertain information

more effectively. Among the various PLP approaches, the one based on the distribution semantics [22] has become particularly popular, in fact it serves as the foundation for numerous other languages, including Probabilistic Logic Programs [8], Probabilistic Horn Abduction (PHA) [18], Independent Choice Logic (ICL) [17], pD [10], Logic Programs with Annotated Disjunctions (LPADs) [26], ProbLog [9] and CP-logic [25]. Such semantics is particularly appealing for its intuitiveness and because efficient inference algorithms were proposed.

A program in one of these languages defines a probability distribution over normal logic programs called *worlds*. We consider here the case of the distribution semantics for programs that do not contain function symbols (i.e., they have a finite set of worlds) for the sake of simplicity; for the treatment of function symbols see [21]. A survey of the distribution semantics in PLP can be found in [4]. In the following, the distribution semantics will be described with reference to LPADs.

Formally, a LPAD consists of a finite set of "annotated disjunctive clauses", where the head is a disjunction in which each atom is annotated with a probability. If the body holds true, only one of the atoms in the head will be true with the associated probability. An annotated disjunctive clause R_i is of the form

$$h_{i1} : p_{i1}; \ldots; h_{in_i} : p_{in_i} :- b_{i1}, \ldots, b_{im_i},$$

where $h_{i1}, \ldots, h_{in_i}$ are logical atoms and $\{p_{i1}, \ldots, p_{in_i}\}$ are real numbers in the interval $[0, 1]$ such that $\sum_{k=1}^{n_i} p_{ik} \leq 1$; $b_{i1}, \ldots, b_{im_i}$ is indicated with $body(R_i)$. 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 $1 - \sum_{k=1}^{n_i} p_{ik}$. We denote by $ground(T)$ the grounding of an LPAD T.

An *atomic choice* [17] is a triple (R_i, θ_j, k) where $R_i \in T$, θ_j is a substitution that grounds R_i and $k \in \{1, \ldots, n_i\}$ identifies one of the head atoms. (R_i, θ_j, k) means that, for the ground clause $R_i\theta_j$, 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 [17]. The *probability* $P(\kappa)$ *of a composite choice* κ is the product of the probabilities of the independent atomic choices, i.e. $P(\kappa) = \prod_{(R_i,\theta_j,k)\in\kappa} p_{ik}$.

A *selection* σ is a composite choice that, for each clause $R_i\theta_j$ in $ground(T)$, contains an atomic choice (R_i, θ_j, k). Let us indicate with S_T the set of all selections. A selection σ identifies a normal logic program w_σ defined as $w_\sigma = \{(h_{ik} \leftarrow body(R_i))\theta_j | (R_i, \theta_j, k) \in \sigma\}$. w_σ is called a (possible) *world* of T. Since selections are composite choices, we can assign a probability to worlds: $P(w_\sigma) = P(\sigma) = \prod_{(R_i,\theta_j,k)\in\sigma} p_{ik}$.

We denote the set of all worlds of T by W_T. $P(W_T)$ is a probability distribution over worlds, i.e., $\sum_{w\in W_T} P(w) = 1$. A composite choice κ identifies a set of worlds $w_\kappa = \{w_\sigma | \sigma \in S_T, \sigma \supseteq \kappa\}$. Similarly we can define the set of possible worlds associated to a set of composite choices K: $W_K = \bigcup_{\kappa\in K} w_\kappa$.

Example 3. Consider the following LPAD T encoding the effect of flu and hay fever on the sneezing symptom.

(R_1) $strong\_sneezing(X) : 0.3; moderate\_sneezing(X) : 0.5 : -flu(X).$
(R_2) $strong\_sneezing(X) : 0.2; moderate\_sneezing(X) : 0.6 : -hay\_fever(X).$
(R_3) $flu(bob).$
(R_4) $hay\_fever(bob).$

If somebody has the flu or hay fever, there is the possibility that he experiences sneezing symptoms with different intensity: if she has the flu, then she might show strong sneezing with probability 0.3, or moderate sneezing with probability 0.5; similarly for the second clause. She might not experience

any symptom with probability 0.2 in both cases. We know for sure that Bob has both the flu and hay fever. T has $3 \cdot 3 = 9$ worlds, as each probabilistic clause has one grounding $\theta_1 = \{X/bob\}$. Worlds are shown in Table 1.

Table 1: Worlds for Example 3. The probabilities of the worlds sum up to 1.

World id	World	$P(w)$
w_1	strong_sneezing(bob):-flu(bob). strong_sneezing(bob):-hay_fever(bob). flu(bob). hay_fever(bob).	0.3×0.2 = 0.06
w_2	strong_sneezing(bob):-flu(bob). moderate_sneezing(bob):-hay_fever(bob). flu(bob). hay_fever(bob).	0.3×0.6 = 0.18
w_3	strong_sneezing(bob):-flu(bob). flu(bob). hay_fever(bob).	0.3×0.2 = 0.06
w_4	moderate_sneezing(bob):-flu(bob). strong_sneezing(bob):-hay_fever(bob). flu(bob). hay_fever(bob).	0.5×0.2 = 0.1
w_5	moderate_sneezing(bob):-flu(bob). moderate_sneezing(bob):-hay_fever(bob). flu(bob). hay_fever(bob).	0.5×0.6 = 0.3
w_6	moderate_sneezing(bob):-flu(bob). flu(bob). hay_fever(bob).	0.5×0.2 = 0.1
w_7	strong_sneezing(bob):-hay_fever(bob). flu(bob). hay_fever(bob).	0.2×0.2 = 0.04
w_8	moderate_sneezing(bob):-hay_fever(bob). flu(bob). hay_fever(bob).	0.2×0.6 = 0.12
w_9	flu(bob). hay_fever(bob).	0.2×0.2 = 0.04

Given a goal G, its probability $P(G)$ can be defined by marginalizing the joint probability of the goal and the worlds:

$$P(G) = \sum_{w \in W_T} P(G, w) = \sum_{w \in W_T} P(G|w)P(w) = \sum_{w \in W_T : w \models G} P(w) \tag{1}$$

The probability of a goal G given a world w is $P(G|w) = 1$ if $w \models G$ and 0 otherwise. $P(w) = P(w_\sigma) = P(\sigma)$, i.e. is the product of the annotations p_{ik} of the atoms selected in σ. Therefore, the probability of a goal G can be computed by summing the probability of the worlds where the goal is true. In practice, given a goal to solve, it is unfeasible to enumerate all the worlds where G is entailed. Inference algorithms, instead, find *explanations* for a goal: a composite choice κ is an *explanation* for G if G is entailed by every world of w_κ. In particular, algorithms find a covering set of explanations w.r.t. G, where a set of composite choices K is *covering* with respect to G if every program $w_\sigma \in W_T$ in which G is entailed is in w_K.

3 Probabilistic Process Traces under the Distribution semantics

In this Section, we propose a new semantics for handling uncertainty at the level of event executions, that is highly inspired by the Distribution Semantics introduced in Section 2.2.

The motivation behind this proposal relies in the fact that in certain domains it might not be possible to completely observe the evolution of a process instance; as a consequence, there is no certainty of the happening of some events in the observed trace. However, due to the domain's characteristics, it may be the case that some events happened with a certain probability. Such probability can be interpreted as an *epistemic probability*, i.e. as the degree of our belief in that event, as done in [20],[19].

Example 4. In a hospital department the log of the patients' events is usually updated by healthcare professionals. They might forget to log one or more events. A probabilistic trace would take in consideration the fact that, even if not logged, some events might have happened with a probability p. The trace:

$$t = \langle 0.7 : \mathsf{check\_fever}, \mathsf{drug\_prescription}, \mathsf{check\_fever}, 0.95 : \mathsf{check\_fever}, \mathsf{dimission} \rangle$$

describes the situation where the healthcare professional logged only the second event (drug_prescription), the third event (check_fever), and the fifth event (dimission). Two events of type check_fever were not logged, however it is probable that they happened anyway due to the patient's disease and treatment. In these cases we have a degree of belief 0.7 and 0.95 respectively in the occurrence of those events, i.e. we *believe* that those events happened with that probability.

We propose to deal with this scenario by applying the Distribution Semantics as follows.

Definition 6 (Probabilistic Event). *A* Probabilistic Event *is a couple:*

$$Prob : \mathit{EventDescription}$$

where EventDescription *is a symbol describing an event (*EventDescription $\in \Sigma$*), while* Prob $\in [0, 1]$ *is the epistemic probability that the event happened. A probability value of 1 means the event happened, and we will refer to it as "certain"; for ease of reading such probability will be omitted.*

For the sake of simplicity, we will adopt the notation $Prob : \mathsf{E}_i$ indicating that the generic event E appearing in the i-th position in the trace has a degree of belief $Prob$.

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

Recalling the Distribution Semantics, for each probabilistic event we can make an *atomic choice*, which determines whether a probabilistic event appears or not in the trace.

Definition 8 (Atomic choice and Composite choice). *An* Atomic Choice *is a pair* (E_i, k) *where* E_i *is a probabilistic event* E *appearing in the* i*-th position in a probabilistic trace and* $k \in \{0, 1\}$*.* k *indicates whether* E_i *is chosen to be included in a world with probability* p_i *(k=1), or not with probability* $1 - p_i$ *(k=0). We assume independence between the different atomic choices.*

A Composite Choice $\kappa(t)$ *is a consistent set of atomic choices over probabilistic events in* t*, i.e.,* $(\mathsf{E}_i, k) \in \kappa(t), (\mathsf{E}_i, m) \in \kappa(t) \Rightarrow k = m$ *(only one decision for each probabilistic event). The* probability $P(\kappa)$ of a composite choice κ *is* $P(\kappa) = \prod_{(E_i,1)\in\kappa} p_i \prod_{(E_i,0)\in\kappa} (1 - p_i)$*, where* p_i *is the probability associated with* E_i*.*

Note that here we do not need the substitution θ as in the definition of atomic choice in subsection 2.2, since, according to Def. 1, the events are ground.

Definition 9 (Selection over a trace t). *A* Selection $\sigma(t)$ *over a probabilistic trace* t *is a composite choice containing an atomic choice* (E_i, k) *for each probabilistic event in the trace* t*.*

A selection $\sigma(t)$ identifies a world w_σ in this way: $w_\sigma = \{E_i | (E_i, 1) \in \sigma\}$.

Let us indicate with $S(t)$ the set of all selections and with $W(t)$ the set of all worlds.

Example 5 (continued from previous). Given the trace t:

$$t = \langle 0.7 : \mathsf{check\_fever}_1,\ \mathsf{drug\_prescription}_2,\ \mathsf{check\_fever}_3,\ 0.95 : \mathsf{check\_fever}_4,\ \mathsf{dimission}_5 \rangle$$

where we explicitly add a pedix for each event due to the multiple appearance of the check_fever activity, the set $S(t)$ of possible selections $\sigma(t)$ is:

$$\begin{aligned}
\sigma_1(t) &= \{(\mathsf{check_fever}_1, 1), (\mathsf{check_fever}_4, 1)\} \\
\sigma_2(t) &= \{(\mathsf{check_fever}_1, 1), (\mathsf{check_fever}_4, 0)\} \\
\sigma_3(t) &= \{(\mathsf{check_fever}_1, 0), (\mathsf{check_fever}_4, 1)\} \\
\sigma_4(t) &= \{(\mathsf{check_fever}_1, 0), (\mathsf{check_fever}_4, 0)\}
\end{aligned}$$

The set $W(t)$ of possible worlds $w(t)$ is:

$$\begin{aligned}
w_{\sigma 1}(t) &= \langle \mathsf{check_fever}_1,\ \mathsf{drug_prescription}_2,\ \mathsf{check_fever}_3,\ \mathsf{check_fever}_4,\ \mathsf{dimission}_5 \rangle \\
w_{\sigma 2}(t) &= \langle \mathsf{check_fever}_1,\ \mathsf{drug_prescription}_2,\ \mathsf{check_fever}_3,\ \mathsf{dimission}_5 \rangle \\
w_{\sigma 3}(t) &= \langle \mathsf{drug_prescription}_2,\ \mathsf{check_fever}_3,\ \mathsf{check_fever}_4,\ \mathsf{dimission}_5 \rangle \\
w_{\sigma 4}(t) &= \langle \mathsf{drug_prescription}_2,\ \mathsf{check_fever}_3,\ \mathsf{dimission}_5 \rangle
\end{aligned}$$

Note that certain (non probabilistic) events always appear in the generated worlds.

Definition 10 (Probability of a Selection). *The probability of a selection $\sigma(t)$ is defined as:*

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

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

The probability of a world is obtained by multiplying the probabilities associated to each alternative (presence or absence of an event) as these are considered independent of each other. This gives a probability distribution $P(w(t))$ over the worlds, i.e. $\sum_{w(t)\in W(t)} P(w(t)) = 1$.

Example 6 (continued from previous).

The trace t has four worlds, corresponding to the selections listed in Example 5, and their probabilities are:

$$\begin{aligned}
P(\sigma_1(t)) = P(w_{\sigma 1}(t)) &= 0.7 * 0.95 &&= 0.665 \\
P(\sigma_2(t)) = P(w_{\sigma 2}(t)) &= 0.7 * (1 - 0.95) &&= 0.035 \\
P(\sigma_3(t)) = P(w_{\sigma 3}(t)) &= (1 - 0.7) * 0.95 &&= 0.285 \\
P(\sigma_4(t)) = P(w_{\sigma 4}(t)) &= (1 - 0.7) * (1 - 0.95) &&= 0.015
\end{aligned}$$

Note that 0.665+0.035+0.285+0.015=1.

Example 5 allows us to highlight some characteristics of the semantics proposed in this paper. First of all, thanks to the notion of *world*, we started from a probabilistic trace, and ended up with a set of regular (non-probabilistic) traces, that correspond to the worlds $w_{\sigma i}(t)$.

4 Probabilistic Compliance

In this Section we will define the notion of *compliance* of a probabilistic trace w.r.t. a Declarative Process Specification. Given a model M and a notion of compliance of a trace t w.r.t. M, we can extend this notion to a probabilistic trace by considering the compliance of each world $w_\sigma(t)$ generated by a trace, and by summing up the probabilities of the worlds compliant with M.

Definition 11 (Compliance of a probabilistic trace). *Given a model M, a probabilistic trace t, and a set $S(t)$ of all the selections $\sigma_i(t)$, we define the compliance Comp(M,t) of a probabilistic trace t w.r.t. M as the sum of the probabilities of the worlds $w_{\sigma i}(t)$ for which the compliance function returns true.*

$$Comp(M,t) = \sum_{w(t)\in W(t)} \begin{cases} P(w(t)) & \text{if } w(t) \text{ is compliant with } M, \\ 0 & \text{otherwise.} \end{cases}$$

Example 7 (continued from previous). Suppose we have a model M that prescribes that before any drug_prescription, a check_fever must be executed. Only worlds $w_{\sigma 1}(t)$ and $w_{\sigma 2}(t)$ are compliant with the prescribed model. Hence:

$$Comp(M,t) = P(w_{\sigma 1}(t)) + P(w_{\sigma 2}(t)) = 0.665 + 0.035 = 0.7$$

We will say that t is compliant w.r.t. model M with a probability of 0.7.

Definition 12 (Compliance of a log with probabilistic traces). *Given a model M and a log $\mathcal{L}$ with probabilistic traces t_i, we define the compliance of a log with probabilistic traces as the weighted sum of the compliance of t_i, where where $\mathcal{L}(t_i)$ counts the occurrences of trace ti in the log L (see Def. 2).*

$$Comp(M,\mathcal{L}) = \sum_{t_i\in\mathcal{L}} \frac{\mathcal{L}(t_i)}{|\mathcal{L}|} * Comp(M,t_i)$$

We might observe that, w.r.t. Def. 5, we move forward from a crisp boolean concept towards a *degree* of compliance: this is an expected consequence of introducing probabilities in the traces' events. The proposed semantics accommodates for both probabilistic and non-probabilistic traces: the latter are treated as traces with only certain events as usual in Process Mining, and the derived worlds w_{σ_i} will always contain them.

5 Implementation and Evaluation

In this Section we describe how we compute the compliance of a probabilistic trace w.r.t a Declarative Process Specification, by means of the framework implemented in [5]. This framework enables reasoning on ALP programs that incorporate "integrity constraints" (ICs) à la IFF [11], extended with the possibility of annotating them with a probability. The programs, known as IFF^{Prob}programs, define a probability distribution over IFF programs based on the distribution semantics. IFF^{Prob}is based on the $\mathcal{S}$CIFF proof-procedure [1]. In [5], the IFF sub-language is extended with a CHR constraint that represents the current explanation with its probability in the current derivation branch. Here, *expl* is a set of pairs (C_i, k), where C_i represents a (possibly) probabilistic integrity constraint and the Boolean value k determines whether C_i is included in *expl*.

To perform the experiments, firstly we described, in the CLIMB language [13], a DECLARE process model for elective colorectal surgery based on the ERAS® (Enhanced Recovery After Surgery) Society guidelines for perioperative care [23]. The model comprises 21 constraints regarding crucial perioperative events, from patient admission to post-surgery recovery. CLIMB (Computational Logic for the Verification and Modeling of Business constraints) is a rule-based language which employs forward rules to bind event occurrences with *expected* courses of interaction. An event execution status is denoted as happened (H), expected (E), or forbidden (though forbidden activities are not considered here). For example, the fact that an atomic activity a has happened at time T can be denoted by $H(\text{exec}(a), T)$, while $E(\text{exec}(a), T)$ states that a is expected to occur at time T. Being the body of integrity constraints universally quantified with scope the entire rule, every time a group of event occurrences makes its body true, then the rule triggers, generating the expectations contained in the head. An excerpt of our model is:

$$\text{true} \rightarrow E(\text{exec}(\text{program\_admission}), 0). \qquad (ic_1)$$

$$H(\text{exec}(\text{program\_admission}), T_1) \rightarrow E(\text{exec}(\text{counseling}), T_2) \wedge T_2 > T_1. \qquad (ic_2)$$

Integrity constraint ic_1 specifies that the process must always start with the event *program admission* of the patient, occurring at time 0 (init template). ic_2 represents the response constraint, indicating that whenever a *program admission* happens, the pre-operative *counseling* (patient education) must eventually follow at a time T2 later than T1.

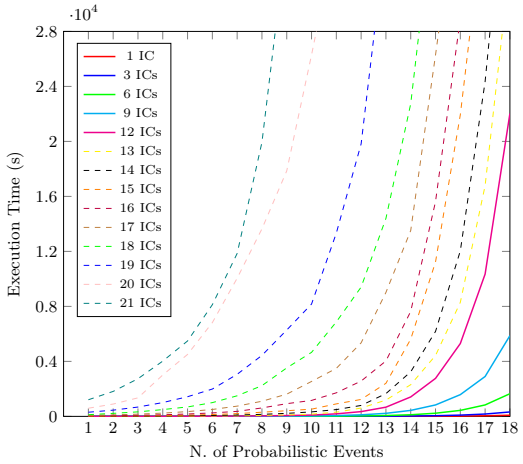
Secondly, we made one or more events probabilistic in a process trace by relying on the same syntax used for the constraints. Every event is represented as a happened (H) event in the head of a rule with body *true*, so that the head is always enacted. This is enabled by the $\mathcal{S}$CIFF-lite extension [13], where the head of the IC may include happened events. To assign a probability p to uncertain events, we place $p ::$ before the rule. An example of an execution trace composed of both certain and probabilistic events is the following:

$$true \longrightarrow H(event(program\_admission), 0).$$
$$0.75 :: true \longrightarrow H(event(counseling), 5).$$
$$0.95 :: true \longrightarrow H(event(fasting), 80).$$

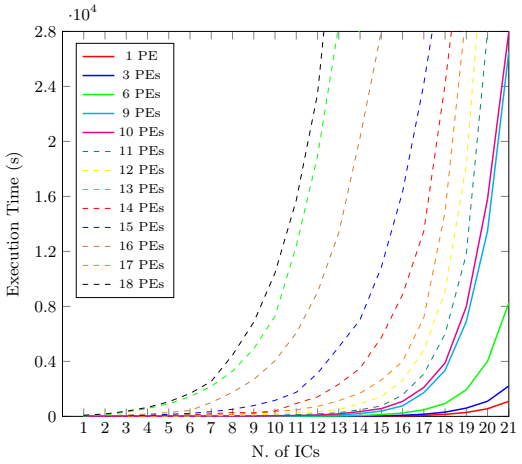
We created a process trace using 21 events, each one corresponding to a different activity provided by the protocol. Of these, we decided to keep the main three as certain given their meaning: `program_admission`, `start_surgery`, and `end_surgery`, while all the remaining events were kept as certain or were assigned a probability value of 0.5, representing the degree of belief associated with each event's occurrence (see in the following for more details). For our purposes, there is no difference in associating 0.5 or any other probability value.

We ran two scalability tests to evaluate how the execution time for computing the probability of compliance of a probabilistic trace w.r.t a DS scales, (a) by increasing the number of probabilistic events (PEs) in the trace, given a fixed number of ICs, or (b) by increasing the number of ICs in the model, given a fixed number of probabilistic events.

Tests were executed on a Linux machine equipped with an Intel® Xeon® E5-2630v3 processor, at 2.40 GHz, 20 GB of RAM and a maximum execution time of 8 hours for each job. In test (a), the number of probabilistic events in the trace was increased from 1 to 18 at steps of 1, while the number of ICs was increased in multiples of 3, from 1 to 12. Starting from 13 ICs, constraints were added at steps of 1 up to 21 ICs, in order to identify the configurations leading to timeout. Here, the "larger" configuration (in terms of number of ICs + number of PEs) reaching timeout was 13 ICs



(a) Execution time as the number of probabilistic events varies with different numbers of ICs in the Declarative Process Specification.



(b) Execution time as the number of ICs varies with different numbers of probabilistic events (PEs).

Fig. 2: Execution time for computing the compliance of a probabilistic trace against a Declarative Process Specification, with varying probabilistic events (a) and ICs (b). Dashed curves indicate that the T.O. was reached.

and 18 PEs. Results are shown in Fig. 2a. In test (b) the number of PEs was increased in multiples of 3, from 1 up to 9, then at steps of 1 from 10 to 18. The number of ICs was increased from 1 to 21 at steps of 1. The larger configuration that led to timeout was the one with 10 PEs and 21 ICs. Figure 2b illustrates the results.

In both cases the framework is able to manage a remarkable number of PEs and constraints in the model at the same time, even if it shows an exponential trend in execution times as the number of constraints or of probabilistic events increases. In particular, it is able to compute the compliance probability within about one minute with, for instance, 4 PEs and 15 ICs, or 9 PEs and 12 ICs, 11 PEs and 9 ICs, 13 PEs and 6 ICs (plus the 3 main certain events in all cases). We also computed, for case (a), the number of explanations generated (see Table 2), which initially grows exponentially with few ICs and increasing number of probabilistic events, and then becomes constrained as more ICs need to be fulfilled. The exponential number of explanations results from each probabilistic event having two atomic choices, present or absent, in a world.

6 Conclusions and Future Work

In this paper, we introduced a new semantics for handling uncertainty at the level of events in declarative process traces. A probabilistic trace is able to incorporate probabilistic events where the probability represents the degree of belief in such events. The semantics is inspired by the Distribution Semantics from Probabilistic Logic Programming. We then propose to compute the compliance probability of a probabilistic trace against a DECLARE process model. The computation of the compliance relies on an existing algorithm. Tests show that, even if the execution time increases exponentially with additional probabilistic events and constraints, the approach used is able to manage combinations of a large number of model constraints and probabilistic events. Future research will focus on expanding experimental validation, extending the new semantics to manage uncertainty at the level of traces or the log itself, defining the compliance of probabilistic

Table 2: N. of explanations corresponding to each experiment of test (a). "T.O." indicates that the execution timed out.

	Number of Probabilistic Events																	
#IC	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	2	4	8	16	32	64	128	256	512	1024	2048	4096	8192	16384	32768	65536	131072	262144
3	1	2	4	8	16	32	64	128	256	512	1024	2048	4096	8192	16384	32768	65536	131072
6	1	2	4	8	8	8	8	16	32	64	128	256	512	1024	2048	4096	8192	16384
9	1	1	1	1	1	1	1	2	4	8	16	32	64	128	256	512	1024	2048
12	1	1	1	1	1	1	1	1	1	1	2	4	8	16	32	64	128	256
13	1	1	1	1	1	1	1	1	1	1	1	2	4	8	16	32	64	T.O.
14	1	1	1	1	1	1	1	1	1	1	1	1	2	4	8	16	32	T.O.
15	1	1	1	1	1	1	1	1	1	1	1	1	2	4	8	16	T.O.	T.O.
16	1	1	1	1	1	1	1	1	1	1	1	1	1	2	4	T.O.	T.O.	T.O.
17	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2	T.O.	T.O.	T.O.
18	1	1	1	1	1	1	1	1	1	1	1	1	1	1	T.O.	T.O.	T.O.	T.O.
19	1	1	1	1	1	1	1	1	1	1	1	1	T.O.	T.O.	T.O.	T.O.	T.O.	T.O.
20	1	1	1	1	1	1	1	1	1	1	T.O.	T.O.	T.O.	T.O.	T.O.	T.O.	T.O.	T.O.
21	1	1	1	1	1	1	1	1	T.O.	T.O.	T.O.	T.O.	T.O.	T.O.	T.O.	T.O.	T.O.	T.O.

trace(s) w.r.t *probabilistic* DECLARE models, and studying the profiles of energy consumption when computing the probability of compliance.

7 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. Research partly funded by the Italian Ministry of University and Research through PNRR - M4C2 - Investimento 1.3 (Decreto Direttoriale MUR n. 341 del 15/03/2022), Partenariato Esteso PE00000013 - "FAIR - Future Artificial Intelligence Research" - Spoke 8 "Pervasive AI" - CUP J33C22002830006, funded by the European Union under the NextGeneration EU programme". This work was realized by Daniela Loreti with a research contract co-financed by the European Union (PON Ricerca e Innovazione 2014-2020 art. 24, comma 3, lett. a), della Legge 30/12/2010, n. 240 e s.m.i. e del D.M. 10/08/2021 n. 1062).

References

1. Alberti, M., Chesani, F., Gavanelli, M., Lamma, E., Mello, P., Torroni, P.: Verifiable agent interaction in abductive logic programming: The SCIFF framework. ACM Trans. Comput. Log. **9**(4), 29:1–29:43 (2008). https://doi.org/10.1145/1380572.1380578
2. Alman, A., Maggi, F.M., Montali, M., Peñaloza, R.: Probabilistic declarative process mining. Inf. Syst. **109**, 102033 (2022). https://doi.org/10.1016/J.IS.2022.102033
3. Azzolini, D., Bellodi, E., Ferilli, S., Riguzzi, F., Zese, R.: Abduction with probabilistic logic programming under the distribution semantics. Int. J. Approx. Reason. **142**, 41–63 (2022). https://doi.org/10.1016/J.IJAR.2021.11.003

4. Bellodi, E.: The distribution semantics in probabilistic logic programming and probabilistic description logics: a survey. Intelligenza Artificiale **17**(1), 143 – 156 (2023). https://doi.org/10.3233/IA-221072
5. Bellodi, E., Gavanelli, M., Zese, R., Lamma, E., Riguzzi, F.: Nonground abductive logic programming with probabilistic integrity constraints. Theory and Practice of Logic Programming **21**(5), 557–574 (2021). https://doi.org/10.1017/S1471068421000417
6. Bergami, G., Maggi, F.M., Montali, M., Peñaloza, R.: Probabilistic trace alignment. In: 2021 3rd International Conference on Process Mining (ICPM). pp. 9–16 (2021). https://doi.org/10.1109/ICPM53251.2021.9576856
7. Ciccio, C.D., Montali, M.: Declarative process specifications: Reasoning, discovery, monitoring. In: van der Aalst, W.M.P., Carmona, J. (eds.) Process Mining Handbook, LNBIP, vol. 448, pp. 108–152. Springer (2022). https://doi.org/10.1007/978-3-031-08848-3_4
8. Dantsin, E.: Probabilistic logic programs and their semantics. In: Russian Conference on Logic Programming. LNCS, vol. 592, pp. 152–164. Springer (1991)
9. De Raedt, L., Kimmig, A., Toivonen, H.: ProbLog: A probabilistic Prolog and its application in link discovery. In: Veloso, M.M. (ed.) 20th International Joint Conference on Artificial Intelligence (IJCAI 2007). vol. 7, pp. 2462–2467. AAAI Press (2007)
10. Fuhr, N.: Probabilistic datalog: Implementing logical information retrieval for advanced applications. Journal of the American Society for Information Science **51**, 95–110 (2000)
11. Fung, T.H., Kowalski, R.A.: The IFF proof procedure for abductive logic programming. Journal of Logic Programming **33**(2), 151–165 (1997). https://doi.org/10.1016/S0743-1066(97)00026-5
12. Giacomo, G.D., Vardi, M.Y.: Linear temporal logic and linear dynamic logic on finite traces. In: Rossi, F. (ed.) IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013. pp. 854–860. IJCAI/AAAI (2013), http://www.aaai.org/ocs/index.php/IJCAI/IJCAI13/paper/view/6997
13. Montali, M.: Specification and Verification of Declarative Open Interaction Models - A Logic-Based Approach, LNBIP, vol. 56. Springer (2010). https://doi.org/10.1007/978-3-642-14538-4
14. Pegoraro, M., van der Aalst, W.M.: Mining uncertain event data in process mining. In: 2019 International Conference on Process Mining (ICPM). pp. 89–96 (2019). https://doi.org/10.1109/ICPM.2019.00023
15. Pegoraro, M., Bakullari, B., Uysal, M.S., van der Aalst, W.M.P.: Probability estimation of uncertain process trace realizations. In: Munoz-Gama, J., Lu, X. (eds.) Process Mining Workshops. pp. 21–33. Springer International Publishing, Cham (2022)
16. Pesic, M.: Constraint-based workflow management systems : shifting control to users. Phd thesis 1 (research tu/e / graduation tu/e), Industrial Engineering and Innovation Sciences (2008). https://doi.org/10.6100/IR638413, proefschrift.
17. Poole, D.: The Independent Choice Logic for modelling multiple agents under uncertainty. Artificial Intelligence **94**, 7–56 (1997)
18. Poole, D.: Logic programming, abduction and probability - a top-down anytime algorithm for estimating prior and posterior probabilities. New Generation Computing **11**(3), 377–400 (1993)
19. Riguzzi, F., Bellodi, E., Lamma, E., Zese, R.: Epistemic and statistical probabilistic ontologies. In: Bobillo, F., Carvalho, R., da Costa, P.C.G., Fanizzi, N., Laskey, K.B., Laskey, K.J., Lukasiewicz, T., Martin, T., Nickles, M., Pool, M. (eds.) Proceedings of the 8th International Workshop on Uncertain Reasoning for the Semantic Web (URSW2012), Boston, USA, 11 November 2012. pp. 3–14. No. 900 in CEUR Workshop Proceedings, Sun SITE Central Europe, Aachen, Germany (2012)
20. Riguzzi, F., Bellodi, E., Lamma, E., Zese, R.: Reasoning with probabilistic ontologies. In: Yang, Q., Wooldridge, M.J. (eds.) Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015. pp. 4310–4316. AAAI Press (2015), http://ijcai.org/Abstract/15/613
21. Riguzzi, F., Swift, T.: Well-definedness and efficient inference for probabilistic logic programming under the distribution semantics. Theory and Practice of Logic Programming **13**(2), 279–302 (2013). https://doi.org/10.1017/S1471068411000664

22. Sato, T.: A statistical learning method for logic programs with distribution semantics. In: Sterling, L. (ed.) Logic Programming, Proceedings of the Twelfth International Conference on Logic Programming, Tokyo, Japan, June 13-16, 1995. pp. 715–729. MIT Press (1995)
23. 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**(3), 659–695 (Mar 2019). https://doi.org/10.1007/s00268-018-4844-y
24. van der Aalst, et al.: Process mining manifesto. In: Daniel, F., Barkaoui, K., Dustdar, S. (eds.) Business Process Management Workshops - BPM 2011 International Workshops, Clermont-Ferrand, France, August 29, 2011, Revised Selected Papers, Part I. LNBIP, vol. 99, pp. 169–194. Springer (2011). https://doi.org/10.1007/978-3-642-28108-2_19
25. Vennekens, J., Denecker, M., Bruynooghe, M.: CP-logic: A language of causal probabilistic events and its relation to logic programming. Theory and Practice of Logic Programming **9**(3), 245–308 (2009). https://doi.org/10.1017/S1471068409003767
26. Vennekens, J., Verbaeten, S., Bruynooghe, M.: Logic programs with annotated disjunctions. In: Demoen, B., Lifschitz, V. (eds.) 20th International Conference on Logic Programming (ICLP 2004). LNCS, vol. 3131, pp. 431–445. Springer (2004). https://doi.org/10.1007/978-3-540-27775-0_30