Conformance Checking with Uncertainty via SMT

Paolo Felli[1], Alessandro Gianola[1], Marco Montali[1], Andrey Rivkin[1], and Sarah Winkler[1]

Free University of Bozen-Bolzano, Bolzano, Italy
{pfelli,gianola,montali,rivkin,winkler}@inf.unibz.it

Abstract. Logs of real-life processes often feature uncertainty pertaining the recorded timestamps, data values, and/or events. We consider the problem of checking conformance of uncertain logs against data-aware reference processes. Specifically, we show how to solve it via SMT encodings, lifting previous work on data-aware SMT-based conformance checking to this more sophisticated setting. Our approach is modular, in that it homogeneously accommodates for different types of uncertainty. Moreover, using appropriate cost functions, different conformance checking tasks can be addressed. We show the correctness of our approach and witness feasibility through a proof-of-concept implementation.

1 Introduction

Process mining is a well-established field of research at the intersection between BPM and data science. The vast majority of process mining tasks assumes that their input event data provide an accurate and complete digital footprint of reality [20]. In many settings, this is an unrealistic assumption: events may be missing or totally/partially wrongly recorded, due to various factors such as human errors, faulty loggers, errors in the acquisition of events (e.g., through sensors), etc. To mitigate this issue, two lines of research emerged lately. The first deals with methodologies and techniques to improve the quality of event data, thus handling uncertainty in the data preparation phase [21]. The second aims instead at incorporating the management of uncertainty within the process mining tasks themselves, leading to a new generation of process mining techniques where process models [13,18,4,1] and/or event logs [17,8] explicitly address different kinds of uncertainty.

Surprisingly enough, the latter has received much less attention from the community. In this work, we aim at contributing to the advancement of process mining on uncertain data, considering in particular the problem of conformance checking [7]. Specifically, our contribution is twofold:

1. We introduce a framework for data-aware conformance checking over uncertain logs, through a suitably extended notion of alignment. The framework employs Data Petri nets [14] for reference process models, and addresses event logs incorporating sophisticated forms of uncertainty, pertaining the recorded timestamps, data values, and/or events. Notably, the framework comes with a generic cost function whose components can be flexibly instantiated to homogeneously account for a variety of measures required for computing optimal alignments.

2. We devise a corresponding operational counterpart to effectively attack the problem of computing alignments and their costs. Instead of relying on ad-hoc algorithmic techniques, our approach builds on and extends [11] to encode the problem into the well-established automated reasoning framework of SMT. This allows us to employ state-of-the-art SMT solvers.

To handle uncertainty in the log, we follow the approach in [17], where the log is explicitly enriched with annotations reflecting the degree and nature of uncertainty. Such annotations may be derived from operational characteristics of the information system recording the event data (considering its logging precision and reliability), and/or by directly attaching them to the generated events. For instance, the log may be enriched with explicit details on the coarseness or precision of an automatic logging device (such as a sensor); alternatively, uncertainty-related annotations may be derived from domain knowledge on the precision and frequency of a specific human activity. In particular, our framework accounts for four main types of uncertain event data.

- *Uncertain events*: these are recorded in a log trace but come with a known *confidence value*, capturing the degree of (un)certainty about the fact that a recorded event actually happened at all during the process execution.
- *Uncertain timestamps*: due to coarseness of the logging activity, events are in general not totally ordered, but come with a fixed range of possible timestamp values. This calls for considering multiple possible orderings and treating a log trace as *a set* of events rather than a sequence.
- *Uncertain activities*: this pertains events whose reference activity is not certainly known. Hence, the event comes with a candidate set of possible activities (each with its own confidence value).
- *Uncertain data values*: in the execution of data-aware processes, for instance due to sensor precision, event data attributes may come with both coarseness and ambiguity. Specifically, the log may only record a set of possible values or an interval for a given attribute, requiring all possible values to be considered.

We stress that the notion of confidence used here should not be confused with that of probability: it measures the degree of trust in the recorded behaviour, which has nothing to do with the likelihood/frequency of such a behaviour.

To account for these different types of uncertain event data, we borrow from [17] and adapt to our data-aware setting the notion of *realization*. A realization of a log trace with uncertainty is an *ordered sequence* of events in which the uncertainty of all types of event data as above is resolved. Our task then concretely becomes as follows: given a Data Petri net and a log trace with uncertainty, find some realization of that trace that admits an *optimal alignment*, i.e., an alignment of minimal cost among all possible realizations for that log trace. Differently from [17], the confidence values of the original trace are used as an essential component for measuring the cost incurred in selecting realizations.

Crucially, since we are in a data-aware setting, a log trace may correspond to infinitely many possible realizations. This is handled symbolically thanks to our SMT-based approach.

The rest of the paper is organized as follows. First, in Sec. 2 we recall the required preliminaries. Then, in Sec. 3 we fix the shape of traces in event logs with uncertainty

and the notion of alignments. In Sec. 4 we detail the cost components that must be considered in the setting with uncertain even data and that we use to define the conformance checking task. We discuss separately one main cost component: the notion of data-aware alignment cost function (in Sec. 4.1). In Sec. 5 we illustrate our SMT-based encoding and we report on the implementation. We conclude in Sec. 6.

2 Preliminaries

In this section we recall data Petri nets (DPNs) and their execution semantics, and the main notions of the machinery behind our approach, namely SMT.

2.1 Data Petri Nets

We use Data Petri nets (DPNs) for modelling multi-perspective processes, adopting the same formalization as in [11,14]. For lack of space, in what follows we only recall the definitions and notation required for our technical development, referring the reader to [11,14] for further details.

Let V be a set of *process variables*, each with a type and an associated domain: booleans (type `bool`), integers (`int`), rationals (`rat`) or strings (`string`). We consider two disjoint sets of annotated variables $V^r = \{v^r \mid v \in V\}$ and $V^w = \{v^w \mid v \in V\}$ to be read and written by process activities, as explained below. Based on these, we define constraints according to the grammar for c:

$$c ::= v_b \mid b \mid n \geq n \mid r \geq r \mid r > r \mid s = s \mid c \wedge c \mid \neg c \qquad s ::= v_s \mid t$$
$$n ::= v_z \mid z \mid n + n \mid -n \qquad r ::= v_r \mid q \mid r + r \mid -r$$

where $v_b \in V_{\texttt{bool}}$, $b \in \mathbb{B}$, $v_s \in V_{\texttt{string}}$, $t \in \mathbb{S}$, $v_z \in V_{\texttt{int}}$, $z \in \mathbb{Z}$, $v_r \in V_{\texttt{rat}}$, and $q \in \mathbb{Q}$. Standard equivalences apply, hence disjunction (i.e., $\vee$) and comparisons $>$, $\neq$, $<$, $\leq$ can be used as well (`bool` and `string` only support (in)equality). The set of constraints over variables V is denoted $\mathcal{C}(V)$. These form the basis for expressing conditions on the values of variables that are read and written during the execution of process activities. For instance, a constraint $(v_1^r > v_2^r)$ dictates that the current value of variable v_1 is greater than the current value of v_2. Similarly, $(v_1^w > v_2^r + 1) \wedge (v_1^w < v_3^r)$ requires that the new value given to v_1 (i.e., assigned as a result of the execution of the activity to which this constraint is attached) is greater than the current value of v_2 plus 1, and smaller than v_3.

Definition 1 (DPN). *A tuple $\mathcal{N} = (P, T, F, \ell, A, V, guard)$ is a* Petri net with data *(DPN), where:*

- *(P, T, F, ℓ) is a Petri net with two non-empty disjoint sets of places P and transitions T, a flow relation $F : (P \times T) \cup (T \times P) \to \mathbb{N}$ and a labeling function $\ell : T \to A \cup \{\tau\}$, where A is a finite set of activity labels and τ is a special symbol denoting silent transitions;*
- *V is a set of typed process variables; and*
- *$guard\colon T \to \mathcal{C}(V)$ is a guard assignment (for $t \in T$ with $\ell(t) = \tau$ we assume that $guard(t)$ does not use variables in V^w).*

As customary, given $x \in P \cup T$, we use ${}^\bullet x := \{y \mid F(y,x) > 0\}$ to denote the *preset* of x and $x^\bullet := \{y \mid F(x,y) > 0\}$ to denote the *postset* of x.

To assign values to variables, we consider a *state variable assignment*, i.e., a total function α that assigns a value (of the right type) to each variable in V. A *state* in a DPN $\mathcal{N}$ is a pair (M, α) constituted by a marking $M\colon P \to \mathbb{N}$ for the underlying Petri net (P, T, F, ℓ), plus a state variable assignment α. Therefore, a state simultaneously accounts for the control flow progress and for the current values of all variables in V, as specified by α.

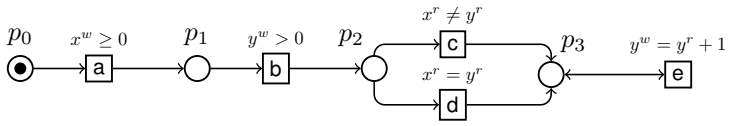
Given $\mathcal{N}$, we fix one state (M_I, α_0) as *initial*, where M_I is the initial marking of the underlying Petri net (P, T, F, ℓ) and α_0 specifies the initial value of all variables in V. Similarly, we denote the final marking as M_F, and call *final* any state of $\mathcal{N}$ of the form (M_F, α_F) for some α_F.

We now define when a Petri net transition may fire from a given state (M, α). Informally, a *transition firing* is a couple (t, β) where $t \in T$ and β is a function used to determine the new values of variables after the transition has fired. The step yields a new state (M', α'), and is denoted $(M, \alpha) \xrightarrow{(t_n, \beta_n)} (M', \alpha')$. A transition firing is *valid* in a state (M, α) when t is enabled in M and α satisfies the constraint associated to t. The formal definition can be found, e.g., in [11,14].

Based on this single-step transition firing, we say that a state (M', α') is *reachable* in a DPN with initial state (M_I, α_0) iff there exists a sequence of valid transition firings of the form $\mathbf{f} = \langle (t_1, \beta_1), \ldots, (t_n, \beta_n) \rangle$ such that $(M_I, \alpha_0) \xrightarrow{(t_1, \beta_1)} \ldots \xrightarrow{(t_n, \beta_n)} (M', \alpha')$. Moreover, such a sequence $\mathbf{f}$ is called a *process run* of $\mathcal{N}$ if $(M_I, \alpha_0) \xrightarrow{\mathbf{f}} (M_F, \alpha_F)$ for some α_F, i.e., if the run leads to a final state. As in [11,15], we restrict to DPNs where at least one final state is reachable.

We denote the set of transition firings of a DPN $\mathcal{N}$ by $\mathcal{F}(\mathcal{N})$, and the set of process runs by $Runs(\mathcal{N})$.

Example 1. Let $\mathcal{N}$ be as shown (with initial marking $[p_0]$ and final marking $[p_3]$). $Runs(\mathcal{N})$ contains, e.g., $\langle (\mathsf{a}, \{x^w \mapsto 2\}), (\mathsf{b}, \{y^w \mapsto 1\}), (\mathsf{c}, \{x^r \mapsto 2, y^r \mapsto 1\}) \rangle$ and $\langle (\mathsf{a}, \{x^w \mapsto 1\}), (\mathsf{b}, \{y^w \mapsto 1\}), (\mathsf{d}, \{y^r \mapsto 1, x^r \mapsto 1\}) \rangle$, for $\alpha_0 = \{x, y \mapsto 0\}$.



2.2 Satisfiability Modulo Theories (SMT)

The classic propositional satisfiability (SAT) problem amounts to, given a propositional formula φ, either find an assignment ν under which φ evaluates to true, or detect that φ is unsatisfiable. E.g., given the formula $(p \vee q) \wedge (\neg p \vee r) \wedge (\neg r \vee \neg q)$, a satisfying assignment is $\nu(p) = \nu(r) = \top$, $\nu(q) = \bot$. The SMT problem [3] is an extension of SAT that consists of establishing satisfiability of a formula φ whose language enriches propositional formulas with constants and operators from one or more theories $\mathcal{T}$ (e.g., arithmetics, bit-vectors, arrays, uninterpreted functions). In this paper, we only consider the theories of linear integer and rational arithmetic ($\mathcal{LIA}$ and $\mathcal{LQA}$). For instance, the

SMT formula $a > 1 \wedge (a + b = 10 \vee a - b = 20) \wedge p$, where a, b are integer and p is a propositional variable, is satisfiable by the assignment ν such that $\nu(a) = \nu(b) = 5$ and $\nu(p) = \top$. Another important problem studied in the area of SMT and relevant to this paper is the one of Optimization Modulo Theories (OMT) [19]. The OMT problem asks, given a formula φ, to find a satisfying assignment of φ that minimizes or maximizes a given objective expression. SMT-LIB [2] is an initiative aiming at providing an extensive on-line library of benchmarks and promoting the adoption of common languages and interfaces for SMT solvers. In this paper, we employ the SMT solvers Yices 2 [10] and Z3 [9].

3 Event Logs with Uncertainty and Alignments

Let ID be a finite set of event identifiers, A be a finite set of activity labels, and TS be a totally ordered set of possible timestamps (for simplicity, we use $\mathbb{N}$).

Definition 2. *An event with uncertainty is a tuple* $ue = \langle \text{ID}, conf, \text{LA}, \text{TS}, \alpha \rangle$ *s.t.*

- $\text{ID} \in ID$ *is an event identifier;*
- $0 < conf \leq 1$ *expresses the confidence that the event actually happened. We say that the event is an uncertain event whenever* $conf < 1$*;*
- $\text{LA} = \{b_1 : p_1, \ldots, b_n : p_n\}$ *is a finite, non-empty subset of activity labels* $b_i \in A$*, each associated to a confidence value* $0 < p_i \leq 1$ *so that* $\sum_{i=1}^{n} p_i = 1$*;*
- TS *is either a finite set of timestamps in* TS *or an interval over* TS*;*
- *with some abuse of notation,* α *is a (possibly partial) function returning for variables in* V *a* finite set *of values in the domain of* v *or an interval over such domain (if* v *is of type* `int` or `rat`).

Given an event $ue = \langle \text{ID}, conf, \text{LA}, \text{TS}, \alpha \rangle$, we denote its components by $\text{ID}(ue)$, $conf(ue)$, $\text{LA}(ue)$, $\text{TS}(ue)$ and $\alpha(ue)$, respectively.

Note that we do not associate confidence values to timestamps, along the lines of [17]. We also do not consider timestamp values following any kind of distribution, e.g., a normal distribution, as this would make the encoding in Section 5 computationally too challenging.

Definition 3. *A log* trace with uncertainty ***ue*** *is a finite set of events with uncertainty, such that all event identifiers are unique.*

Thus, there is no fixed order among the events in a trace with uncertainty. An *event log* L is a multiset of log traces with uncertainty.

Example 2. Consider $\mathcal{N}$ from Ex. 1. For simplicity, we use natural numbers for timestamps. The following are three possible traces with uncertainty:

$$\mathbf{ue}_1 = \{\langle \#_1, .25, \{\mathsf{a} : 1\}, [0\text{-}5], \{x \mapsto \{2, 3\}\}\rangle,\ \langle \#_2, .9, \{\mathsf{b} : .8, \mathsf{c} : .2\}, \{2\}, \{y \mapsto \{1\}\}\rangle$$

$$\mathbf{ue}_2 = \{\langle \#_3, 1, \{\mathsf{a} : 1\}, \{0\}, \{x \mapsto [1, 6.5]\}\rangle,\ \langle \#_4, 1, \{\mathsf{b} : 1\}, \{2\}, \{y \mapsto \{1\}\}\rangle,$$
$$\langle \#_5, 1, \{\mathsf{c} : 1\}, \{3\}, \emptyset\rangle\}$$

$$\mathbf{ue}_3 = \{\langle \#_6, 1, \{\mathsf{a} : 1\}, \{2\}, \{x \mapsto \{6\}\}\rangle,\ \langle \#_7, 1, \{\mathsf{b} : 1\}, \{2\}, \{y \mapsto \{1\}\rangle\}\}$$

For instance, $\mathbf{ue}_1$ has two events with uncertainty: $\#_1$ and $\#_2$. The former is uncertain (confidence 0.25), has event label a (with confidence 1), timestamp interval $[0, 5]$ and a variable assignment such that x is assigned to either 2 or 3. Also $\#_2$ is uncertain, has label b or c (with associated confidence values 0.8 and 0.2, respectively), timestamp 2 and variable assignment $y = 1$. Another example of an uncertain event is $\#_3$ in $\mathbf{ue}_2$, where x takes a value from the interval $[1, 6.5]$.

An activity label $b \in A$ is *admissible* for an event with uncertainty *ue* iff it is consistent with $\text{LA}(ue)$, i.e., if there is some p such that $(b, p) \in \text{LA}(ue)$. Admissibility of timestamp and variable values is defined similarly.

Intuitively, given a log trace with uncertainty $\mathbf{ue}$, a *realization* of $\mathbf{ue}$ is a sequence $\mathbf{e} = \langle e_1, \ldots, e_n \rangle$ of events corresponding to a possible sequentialization of *a subset of* the events with uncertainty in $\mathbf{ue}$ that is consistent with their uncertain timestamps, and in which only one possible value is chosen for event labels and variable assignments. The remaining events with uncertainty in $\mathbf{ue}$ but not in $\mathbf{e}$ are simply discarded.

An event without uncertainty, or simply *event*, is a tuple $(\text{ID}, b, \hat{\alpha})$, where ID is again an event identifier, $b \in A$ is an activity label, and $\hat{\alpha}$ is a special variable assignment that assigns to each variable $v \in V$ a *single* value of the correct type. Given an event $e = (\text{ID}, b, \hat{\alpha})$, we denote its components by $\text{ID}(e)$, $lab(e)$ and $\hat{\alpha}(e)$, respectively. These events are akin to the standard notion of events in conformance checking literature, extended with variable assignments as in [11], with the addition of identifiers (which are needed to relate them to the corresponding event with uncertainty in the log, as explained later). The set of all possible such events is denoted by $\mathcal{E}$.

Definition 4 (Realization). *A sequence $\mathbf{e} = \langle e_1, \ldots, e_n \rangle$ of events as above is a* realization *of a log trace with uncertainty $\mathbf{ue}$ if there is a subset $\{ue_1, \ldots, ue_n\} \subseteq \mathbf{ue}$ and a sequence of timestamps $t_1 \leq t_2 \leq \cdots \leq t_n$ such that for each $i \in [1, n]$:*

(i) t_i is admissible for ue_i, hence defining an ordering on $\mathbf{e}$;
(ii) $\text{ID}(e_i) = \text{ID}(ue_i)$;
(iii) $lab(e_i) = b$ with b admissible for ue_i;
(iv) $\hat{\alpha}(e_i)(v) \in \alpha(ue_i)(v)$ for all v such that $\alpha(ue_i)(v)$ is defined.

Moreover, we impose that for every $ue \in \mathbf{ue}$ with $conf(ue) = 1$ there is an event $e \in \mathbf{e}$ with $\text{ID}(e_i) = \text{ID}(ue_i)$, namely a realization cannot discard events in the log that are not uncertain.

A realization of a trace with uncertainty $\mathbf{ue}$ is thus a possible sequentialization of (a subset of) the events with uncertainty in $\mathbf{ue}$ in which a single, admissible timestamp value, activity label and value for variables are selected from the corresponding event with uncertainty $ue \in \mathbf{ue}$ with $\text{ID}(e) = \text{ID}(ue)$. We denote that $\mathbf{e}$ is a realization of $\mathbf{ue}$ by writing $\mathbf{e} \in \mathcal{R}(\mathbf{ue})$. Events in a realization $\mathbf{e}$ are no longer associated with confidence values (which remain in $\mathbf{ue}$).

Note that $\mathcal{R}(\mathbf{ue})$ cannot be empty, as it is always possible to select $\{t_1, \ldots, t_n\}$ as in Def. 4: even if two events cannot be ordered because they admit the same single timestamp, both orderings are accounted for by different realizations. $\mathcal{R}(\mathbf{ue})$ can be infinite if data variables are assigned by $\mathbf{ue}$ to intervals over dense domains.

Example 3. Consider the trace with uncertainty $\mathbf{ue}_1$ in Ex. 2. It has 13 realizations, since the first event has two possible variable assignments, the second event has two possible labels; moreover, the two events can be ordered in both ways and in addition each event can also be removed (as they are uncertain).

Two possible realizations of $\mathbf{ue}_1$ are $\mathbf{e}' = \langle\langle \#_1, \mathsf{a}, \{x \mapsto 2\}\rangle, \langle \#_2, \mathsf{b}, \{y \mapsto 1\}\rangle\rangle$ and $\mathbf{e}'' = \langle\langle \#_2, \mathsf{c}, \{y \mapsto 1\}\rangle, \langle \#_1, \mathsf{a}, \{x \mapsto 3\}\rangle\rangle$. Note that these realizations differ in the order of the two events, label selection and variable assignments.

We focus on a conformance checking procedure to construct an *alignment* of a log trace $\mathbf{e}$ (that is a realization of a log trace with uncertainty $\mathbf{ue}$) w.r.t. the process model (i.e., the DPN $\mathcal{N}$), by matching event labels in the log trace against transition firings in the process runs of $\mathcal{N}$. However, when constructing an alignment, not every event in the log trace can always be put in correspondence with a transition firing, and vice versa. Therefore, as customary, we consider a special "skip" symbol $\gg$ and the extended set of events $\mathcal{E}^{\gg} = \mathcal{E} \cup \{\gg\}$ and, given $\mathcal{N}$, the extended set of transition firings $\mathcal{F}^{\gg} = \mathcal{F}(\mathcal{N}) \cup \{\gg\}$.

Given a DPN $\mathcal{N}$ and a set $\mathcal{E}$ of events (without uncertainty) as above, a pair $(e, f) \in \mathcal{E}^{\gg} \times \mathcal{F}^{\gg} \setminus \{(\gg, \gg)\}$ is called *move*. A move (e, f) is called: (i) *log move* if $e \in \mathcal{E}$ and $f = \gg$; (ii) *model move* if $e = \gg$ and $f \in \mathcal{F}(\mathcal{N})$; (iii) *synchronous move* if $(e, f) \in \mathcal{E} \times \mathcal{F}(\mathcal{N})$. Let $Moves_{\mathcal{N}}$ be the set of all such moves. We now show how moves can be used to define alignments of realizations.

For a sequence of moves $\gamma = \langle (e_1, f_1), \ldots, (e_n, f_n)\rangle$, the *log projection* $\gamma|_L$ of γ is the subsequence $\langle e'_1, \ldots, e'_i\rangle$ of $\langle e_1, \ldots, e_n\rangle$ that is in $\mathcal{E}^*$ and is obtained by projecting away from γ all $\gg$ symbols. Similarly, the *model projection* $\gamma|_M$ of γ is the subsequence $\langle f'_1, \ldots, f'_j\rangle$ of $\langle f_1, \ldots, f_n\rangle$ such that $\langle f'_1, \ldots, f'_j\rangle \in \mathcal{F}(\mathcal{N})^*$.

Definition 5 (Alignment). *Given $\mathcal{N}$, a sequence of moves γ is a* complete alignment *of a realization $\mathbf{e}$ if $\gamma|_L = \mathbf{e}$ and $\gamma|_M \in Runs(\mathcal{N})$.*

Example 4. Consider the realization $\mathbf{e}' = \langle\langle \#_1, \mathsf{a}, \{x \mapsto 2\}\rangle, \langle \#_2, \mathsf{b}, \{y \mapsto 1\}\rangle\rangle$ from Ex.3. The following are examples of possible complete alignments of $\mathbf{e}'$ with respect to the DPN from Ex. 1:

$\gamma^1_{\mathbf{e}'}$

$\#_1$	$\#_2$	$\gg$
a $x^w \mapsto 2$	b $y^w \mapsto 1$	c

$\gamma^2_{\mathbf{e}'}$

$\#_1$	$\#_2$	$\gg$
a $x^w \mapsto 5$	b $y^w \mapsto 1$	c

$\gamma^3_{\mathbf{e}'}$

$\#_1$	$\gg$	$\#_2$
a $x^w \mapsto 2$	b $y^w \mapsto 2$	d

We denote by $Align(\mathcal{N}, \mathbf{e}')$ the set of all complete alignments for $\mathbf{e}'$ w.r.t. $\mathcal{N}$.

As shown in Ex. 4, some alignments are more fitting than others: for instance, they can have mismatching variable assignments (e.g., in the first move of $\gamma^2_{\mathbf{e}'}$) and label matching (e.g., in the third move of $\gamma^3_{\mathbf{e}'}$). This will be captured by the cost function, described next.

4 Costs and Optimal Alignments

In this paper we do not wish to restrict to specific cost functions, and therefore fix only a *cost schema* which leaves several elements arbitrary. We however illustrate the the cost components and describe one possible instantiation of said schema, which we use in the encoding in Sec. 5. The overall cost schema for alignments is shown in Fig. 1.

$$\mathfrak{K}(\gamma_{\mathbf{e}}, \mathbf{ue}) = \overbrace{\sum_{i \in [1,n]} \underbrace{\kappa(e_i, f_i)}_{\substack{\text{data-aware} \\ \text{alignment cost (Sec. 4.1)}}} \otimes \underbrace{\theta(e_i, \mathbf{ue})}_{\text{confidence cost}}}^{\text{alignment cost } \kappa_A(\gamma_{\mathbf{e}}, \mathbf{ue})} + \overbrace{\sum_{e \in \mathbf{ue}, e \notin \mathbf{e}} \kappa_{\mathbf{ue}}(e)}^{\text{event removal cost } \kappa_R(\mathbf{e}, \mathbf{ue})}$$

Fig. 1. Structure of the cost of an alignment $\gamma_{\mathbf{e}} = \langle (e_1, f_1), \ldots, (e_n, f_n) \rangle$ of a realization $\mathbf{e}$ of a trace with uncertainty $\mathbf{ue}$. The cost associated to the selection of $\mathbf{e}$ is given by $\kappa_R(\mathbf{e}, \mathbf{ue})$ plus, at each step, the additional penalty given by $\theta(e_i, \mathbf{ue})$ according to $\otimes$.

We first give the intuition. The general idea is that, as we are not merely interested in finding a cost-minimal alignment for an arbitrary realization as in [17], i.e., without considering the confidence associated to the selection of realizations, we impose a confidence cost on realizations *in addition* to the cost of aligning them, as illustrated in Fig. 1. As a result, the cost $\mathfrak{K}(\gamma_{\mathbf{e}}, \mathbf{ue})$ of an alignment $\gamma_{\mathbf{e}}$ with respect to an uncertain trace $\mathbf{ue}$ is the sum of two costs:

1) The alignment cost $\kappa_A(\gamma_{\mathbf{e}}, \mathbf{ue})$ measures the quality of the alignment $\gamma_{\mathbf{e}}$ for the realization $\mathbf{e}$. As customary in the conformance checking literature, it is based on a mapping $\kappa \colon Moves_{\mathcal{N}} \to \mathbb{R}^+$ that assigns a cost to every move $(e_i, f_i) \in \gamma_{\mathbf{e}}$. In Sec. 4.1 we will discuss in more detail how this function κ can be defined.

In addition, for synchronous moves and log moves, this cost is combined with a confidence penalty that depends on $conf(e_i)$ and on the confidence value p associated to the activity label $b = lab(e_i)$ according to the event with uncertainty *ue* so that $\text{ID}(e_i) = \text{ID}(ue)$, i.e., $(b, p) \in \text{LA}(ue)$. Intuitively, this imposes a penalty for selecting b as the activity chosen for e_i in the realization $\mathbf{e}$ of $\mathbf{ue}$.

We do not fix a specific calculation of this penalty, but keep it parametric and denote it as $\theta(e_i, \mathbf{ue})$. The cost of an alignment $\gamma_{\mathbf{e}}$ can then be defined as:

$$\kappa_A(\gamma_{\mathbf{e}}, \mathbf{ue}) = \textstyle\sum_{i=1}^{n} \kappa(e_i, f_i) \otimes \theta(e_i, \mathbf{ue})$$

where $\otimes$ denotes an arbitrary operator to combine the two costs.

For instance, in Sec. 5 we assume, for a realization $\mathbf{e}$ of $\mathbf{ue}$ and alignment $\gamma_{\mathbf{e}} = \langle (e_1, f_1), \ldots, (e_n, f_n) \rangle$:

$$\kappa(e_i, f_i) \otimes \theta(e_i, \mathbf{ue}) = \begin{cases} \kappa(e_i, f_i) & \text{if } e_i = \gg, \text{ otherwise:} \\ \theta(e_i, \mathbf{ue}) & \text{if } \kappa(e_i, f_i) = 0 \\ \kappa(e_i, f_i) \cdot (1 + \theta(e_i, \mathbf{ue})) & \text{if } \kappa(e_i, f_i) > 0 \end{cases}$$

in which we fix $\theta(e_i, \mathbf{ue}) = (1 - conf(e_i)) + (1 - p)$, where b is the label of e_i, i.e., $b = lab(e_i)$, and p is the confidence value associated to b, i.e., $(b, p) \in \text{LA}(ue)$.

Intuitively, in this definition of $\kappa_A(\gamma_{\mathbf{e}}, \mathbf{ue})$, the cost of model moves is simply (a data-aware extension of) the usual alignment cost, which we define in Sec. 4.1. Otherwise, the cost includes a penalty for having selected $lab(e_i)$ in the realization $\mathbf{e}$ of $\mathbf{ue}$. Such penalty decreases the more we are confident about the selected activity among the possible activities associated to the event with uncertainty. Other definitions of θ and $\otimes$ are however possible.

2) The event removal cost $\kappa_R(\mathbf{e}, \mathbf{ue})$ measures the cost of selecting the subsets of the events in $\mathbf{ue}$ that appear in $\mathbf{e}$, discarding the remaining (uncertain) events. Although we do not wish to restrict to a specific function κ_R, a reasonable option is to assume it to be based on a mapping $\kappa_{\mathbf{ue}} \colon \mathcal{E} \to \mathbb{R}_{\geq 0}$ that assigns a removal cost to each event, proportionally to the confidence value $conf(ue)$ for $ue \in \mathbf{ue}$ so that $\text{ID}(e) = \text{ID}(ue)$. Hence, the total event removal cost can be computed as:

$$\kappa_R(\mathbf{e}, \mathbf{ue}) = \textstyle\sum_{e \in \mathbf{ue}, e \notin \mathbf{e}} \kappa_{\mathbf{ue}}(e)$$

For instance, in Sec. 5 we will take $\kappa_{\mathbf{ue}}(e)$ to be precisely $conf(ue)$, for ue as above, when such a confidence value is less than 1, and equal to infinity otherwise (to prevent events that are not indeterminate to be discarded from realizations). Other definitions of κ_R are however possible. Again, according to these expressions, the cost of selecting $\mathbf{e}$ as a realization of $\mathbf{ue}$ results from $\kappa_R(\mathbf{e}, \mathbf{ue})$ for removed events plus, at each step, a penalty $\theta(e_i, \mathbf{ue})$ for not having discarded e_i but having selected one admissible label among those associated to the uncertain event in $\mathbf{ue}$ with the same ID.

Example 5. Consider again the trace with uncertainty $\mathbf{ue}_1$ from Example 3:

$\mathbf{ue}_1 = \{\langle \#_1, .25, \{\mathsf{a} : 1\}, [0\text{-}5], \{x \mapsto \{2, 3\}\}\rangle,\ \langle \#_2, .9, \{\mathsf{b} : .8, \mathsf{c} : .2\}, \{2\}, \{y \mapsto \{1\}\}\rangle\}$
and three of its possible realizations $\mathbf{e}_1 = \langle\langle \#_1, \mathsf{a}, \{x \mapsto 3\}\rangle\rangle$, $\mathbf{e}_2 = \langle\langle \#_2, \mathsf{b}, \{y \mapsto 1\}\rangle\rangle$ and $\mathbf{e}_3 = \langle\langle \#_2, \mathsf{c}, \{y \mapsto 1\}\rangle\rangle$, where in all cases one of the two events was removed. If we adopt the specific implementation of cost functions exemplified above (and used in our encoding in Section 5), we have that $\kappa_R(\mathbf{e}_2, \mathbf{ue}_1) > \kappa_R(\mathbf{e}_1, \mathbf{ue}_1)$ since $conf(\#_2) > conf(\#_1)$. Similarly, the difference between $\mathbf{e}_2$ and $\mathbf{e}_3$ is only in the activity chosen for $\#_2$, therefore the cost of selecting $\mathbf{e}_2$ is smaller than that for $\mathbf{e}_3$, because the confidence associated to activity b is greater than the one associated to c; hence $\theta(\langle \#_2, \mathsf{b}, \{y \mapsto 1\}\rangle, \mathbf{ue}_1) < \theta(\langle \#_2, \mathsf{c}, \{y \mapsto 1\}\rangle, \mathbf{ue}_1)$.

Definition 6 (Cost of alignments). *Fixed the two arbitrary cost functions κ_A and κ_R introduced above, given $\mathcal{N}$, a trace with uncertainty $\mathbf{ue}$ that has realization $\mathbf{e} = \langle e_1, \ldots, e_n\rangle$ and an alignment $\gamma_{\mathbf{e}} = \langle (e_1, f_1), \ldots, (e_n, f_n)\rangle \in Align(\mathcal{N}, \mathbf{e})$, the* cost *of $\gamma_{\mathbf{e}}$ w.r.t. $\mathbf{ue}$, denoted $\mathfrak{K}(\gamma_{\mathbf{e}}, \mathbf{ue})$, is obtained as shown in Figure 1:*

$$\mathfrak{K}(\gamma_{\mathbf{e}}, \mathbf{ue}) = \kappa_A(\gamma_{\mathbf{e}}, \mathbf{ue}) + \kappa_R(\mathbf{e}, \mathbf{ue}).$$

An alignment $\gamma_{\mathbf{e}}$ is *optimal for* $\mathbf{e}$ if $\kappa_A(\gamma_{\mathbf{e}}, \mathbf{ue})$ is minimal among all complete alignments for $\mathbf{e}$, i.e., there is no $\gamma'_{\mathbf{e}} \in Align(\mathcal{N}, \mathbf{e})$ with $\kappa_A(\gamma'_{\mathbf{e}}, \mathbf{ue}) < \kappa_A(\gamma_{\mathbf{e}}, \mathbf{ue})$. Similarly, given $\mathcal{N}$ and a trace with uncertainty $\mathbf{ue}$, we say that $\gamma_{\mathbf{e}}$ is *optimal for* $\mathbf{ue}$ if $\mathfrak{K}(\gamma_{\mathbf{e}}, \mathbf{ue})$ is minimal among all possible realizations of $\mathbf{ue}$, i.e., there is no other realization $\mathbf{e}' \in \mathcal{R}(\mathbf{ue})$ and alignment $\gamma_{\mathbf{e}'} \in Align(\mathcal{N}, \mathbf{e}')$ so that $\mathfrak{K}(\gamma_{\mathbf{e}'}, \mathbf{ue}) < \mathfrak{K}(\gamma_{\mathbf{e}}, \mathbf{ue})$.

Definition 7 (Conformance checking). *Given $\mathcal{N}$, the* conformance checking task *for a trace with uncertainty $\mathbf{ue}$ is to find a realization $\mathbf{e}$ of $\mathbf{ue}$ and an alignment $\gamma_{\mathbf{e}}$ that is optimal for $\mathbf{ue}$.*

Multiple realizations $\mathbf{e}$ and optimal alignments $\gamma_{\mathbf{e}}$ may exist for $\mathbf{ue}$, though the minimal cost is unique for a given cost function. The *conformance checking task for an unordered log* consists of the conformance checking task for all its traces.

Note that we can easily formulate the task of finding the lower-bound on the cost of possible alignments among all realizations (as in [17]), given $\mathbf{ue}$, by simply imposing $\kappa_R(\mathbf{e}, \mathbf{ue}) = 0$, $\theta(e, \mathbf{ue}) = 1$ and by taking $\otimes$ as product: this corresponds to impose no cost for selecting an arbitrary realization, thus simply returning one that has minimal alignment cost κ.

In the remainder, we discuss separately the definition of alignment cost κ.

4.1 Data-aware Alignment Cost Function

We use a generalized form of a cost function to measure the conformance between a realization and a process run in $Runs(\mathcal{N})$, i.e., to define $\kappa\colon Moves_{\mathcal{N}} \to \mathbb{R}_{\geq 0}$ used in Def. 6. As in [11], we parameterize this by three penalty functions:

$$P_L\colon \mathcal{E} \to \mathbb{N} \qquad P_M\colon \mathcal{F}(\mathcal{N}) \to \mathbb{N} \qquad P_=\colon \mathcal{E} \times \mathcal{F}(\mathcal{N}) \to \mathbb{N}$$

called *log move penalty*, *model move penalty* and *synchronous move penalty*, respectively. Intuitively, $P_L(e)$ gives the cost that has to be paid for a log move e; $P_M(f)$ penalizes a model move f; and $P_=(e, f)$ expresses the cost to be paid for a synchronous move of e and f. By suitably instantiating $P_=$, P_L, and P_M, one can obtain conventional cost functions [11]: the Levenshtein distance [5,6], standard cost function for multi-perspective conformance checking [15,14].

Then, the data-aware cost function $\kappa\colon Moves_{\mathcal{N}} \to \mathbb{R}_{\geq 0}$ we adopt in Def. 6 is simply defined as $\kappa(e, f) = P_L(e)$ if $f = \gg$, $\kappa(e, f) = P_M(f)$ if $e = \gg$, and $\kappa(e, f) = P_=(e, f)$ otherwise.

Data-aware Cost Component of $P_=$. Crucially, for DPNs we typically consider a data-aware extension of the usual distance-based cost function for synchronous moves. Indeed, given an event $e = (\text{ID}, b, \hat{\alpha})$ of a realization and a transition firing $f = (t, \beta)$, we want $P_=(e, f)$ to compare also the values assigned to variables by $\hat{\alpha}$ and β. For instance, in Ex. 4, the alignment $\gamma^2_{\mathbf{e}_1}$ is so that its first (synchronous) move has a mismatch between the value assigned to variable x by the event $\#_1$ (i.e., $\hat{\alpha}(\#_1)(x) = 2$) and transition firing $(a, \{x^w \mapsto 5\})$. Various data-aware realizations of $P_=$ have been already addressed in the literature [15,11].

Example 6. Consider again the trace with uncertainty $\mathbf{ue}_1$ from Ex. 5, i.e., $\mathbf{ue}_1 = \{\langle \#_1, .25, \{\mathsf{a}:1\}, [0\text{-}5], \{x \mapsto \{2,3\}\}\rangle,\ \langle \#_2, .9, \{\mathsf{b}:.8, \mathsf{c}:.2\}, \{2\}, \{y \mapsto \{1\}\}\rangle\}$. Assume to fix P_M, P_L to be as usual in the standard cost function, as illustrated in [11], namely $P_L(b, \alpha) = 1$; $P_M(t, \beta) = 0$ if t is silent (i.e., $\ell(t) = \tau$) and $P_M(t, \beta)$ equal to 1 plus the number of variables written by $guard(t)$ otherwise. For $P_=$, assume a data-aware extension (of the $P_=$ used to match the standard cost function [11]) defined as: $P_=(\langle \text{ID}, b, \hat{\alpha}\rangle, (t, \beta)) = |\{v \mid \hat{\alpha}(v) \neq \beta(v^w)\}| \,/\, |V|$ if b is the label of t, i.e. $b = \ell(t)$, and $P_=(\langle \text{ID}, b, \hat{\alpha}\rangle, (t, \beta)) = \infty$ otherwise. Then, if we instantiate cost functions as in Ex. 5 (also used in our encoding in Sec. 5), the optimal alignment of $\mathbf{ue}_1$ w.r.t. the DPN $\mathcal{N}$ depicted in Ex. 1 is $\gamma^1_{\mathbf{e}'}$ as shown in Ex. 4 (of cost 2.05).

Further, if we consider the task of finding the lower-bound on the cost of optimal alignments for any realization of $\mathbf{ue}_1$ (as discussed below Def. 7), then this is 1 and it is given as well by the realization $\mathbf{e}'$ and $\gamma^1_{\mathbf{e}'}$.

5 Encoding

In this section we describe our SMT encoding, obtained as the result of 4 steps:
(1) represent the process run, the trace realization, and the alignment symbolically by a set of SMT variables;
(2) set up constraints Φ that express optimality of the alignment;
(3) solve Φ to obtain a satisfying assignment ν;
(4) decode the process run, trace realization, and optimal alignment γ from ν.
The same procedure was followed in [11], with important differences. In step (1), we now need to represent both the process run and also the trace realization, which is complicated by the fact that the order of the events is not fixed. Moreover, the cost functions are defined differently, as described in Sec. 4. These changes also affect the decoding in step (4).

Similarly to earlier SAT-based approaches [6,11], we aim to construct a symbolic representation of both a process run and an alignment, that are subsequently concretized using an SMT solver. Since the symbolic representation depends on a finite set of initial variable declarations (and thus must be finite), we need to fix upfront an upper bound on the size of the process run. This upper bound, and even its existence, depends on the cost function of choice. The Lemma below shows how a (coarse) upper bound can be established for the cost model from Sec. 4, where the cost function is the standard one as in Ex. 6.

Lemma 1. *Let $\mathcal{N}$ be a DPN and $\boldsymbol{u}\mathbf{e}$ a trace with uncertainty that has m_1 certain and m_2 uncertain events. Let $\langle f_1, \ldots, f_n \rangle$ be a run of $\mathcal{N}$ such that $c = \sum_{j=1}^{n} P_M(f_j)$ is minimal, and k the length of the longest acyclic sequence of silent transitions in $\mathcal{N}$. Then there is an optimal alignment γ for $\boldsymbol{u}\mathbf{e}$ such that the length of $\gamma|_M$ is at most $(4m_1 + 2m_2 + c) \cdot k$.*

The proof of this lemma can be found in [12]. Note that, in case the model admits loops that entirely consist of silent transitions, then there can be infinitely many optimal alignments that are not bounded in length (as such loops can be repeated arbitrarily many times without incurring in any additional penalty on the alignment cost). Thus, the above lemma shows only *existence* of an optimal alignment within that bound, but in general the bound does not apply to *all* optimal alignments.

5.1 Encoding the Process Run

Assuming that the process run in the optimal alignment has length at most n, we use the following SMT variables to represent this run:
(a) transition step variables $\mathtt{S}_i$ for $1 \leq i \leq n$ of type integer; if $T = \{t_1, \ldots, t_{|T|}\}$ then it is ensured that $1 \leq \mathtt{S}_i \leq |T|$, so that $\mathtt{S}_i$ is assigned j iff the i-th transition in the process run is t_j;
(b) marking variables $\mathtt{M}_{i,p}$ of type integer for all i, p with $0 \leq i \leq n$ and $p \in P$, where $\mathtt{M}_{i,p}$ is assigned k iff there are k tokens in place p at instant i;
(c) data variables $\mathtt{X}_{i,v}$ for all $v \in V$ and i, $0 \leq i \leq n$; the type of these variables depends on v, with the semantics that $\mathtt{X}_{i,v}$ is assigned r iff the value of v at instant i is r; we also write $\mathtt{X}_i$ for $(\mathtt{X}_{i,v_1}, \ldots, \mathtt{X}_{i,v_k})$.

Note that variables (a)–(c) encode all information required to capture a process run of a DPN with n steps. They will be used to represent the model projection of the alignment γ. To encode the process run, we use the constraints

$$\varphi_{run} = \varphi_{init,fin} \wedge \varphi_{trans} \wedge \varphi_{enabled} \wedge \varphi_{mark} \wedge \varphi_{data}$$

where the subformulas above reflect requirements to the solution as follows:

- The initial and final markings M_I and M_F, and the initial assignment α_0 are respected:

$$\textstyle\bigwedge_{p \in P} \mathtt{M}_{0,p} = M_I(p) \wedge \bigwedge_{v \in V} \mathtt{X}_{0,v} = \alpha_0(v) \wedge \bigwedge_{p \in P} \mathtt{M}_{n,p} = M_F(p) \qquad (\varphi_{init,fin})$$

- Transitions correspond to transition firings in the DPN:

$$\textstyle\bigwedge_{1 \le i \le n} 1 \le \mathtt{S}_i \le |T| \qquad (\varphi_{trans})$$

- Transitions are enabled when they fire:

$$\textstyle\bigwedge_{1 \le i \le n} \bigwedge_{1 \le j \le |T|} (\mathtt{S}_i = j) \rightarrow \bigwedge_{p \in {}^\bullet t_j} \mathtt{M}_{i-1,p} \ge |{}^\bullet t_j|_p \qquad (\varphi_{enabled})$$

  where $|{}^\bullet t_j|_p$ denotes the multiplicity of p in the multiset ${}^\bullet t_j$.
- We encode the token game:

$$\bigwedge_{1 \le i \le n} \bigwedge_{1 \le j \le |T|} (\mathtt{S}_i = j) \rightarrow \bigwedge_{p \in P} \mathtt{M}_{i,p} - \mathtt{M}_{i-1,p} = |t_j{}^\bullet|_p - |{}^\bullet t_j|_p \qquad (\varphi_{mark})$$

  where $|t_j{}^\bullet|_p$ is the multiplicity of p in the multiset $t_j{}^\bullet$.
- The transitions satisfy the constraints on data:

$$\bigwedge_{1 \le i < n} \bigwedge_{1 \le j \le |T|} (\mathtt{S}_i = j) \rightarrow guard(t_j)\chi \wedge \bigwedge_{v \notin write(t_j)} \mathtt{X}_{i-1,v} = \mathtt{X}_{i,v} \qquad (\varphi_{data})$$

  where the substitution χ uniformly replaces V^r by $\mathtt{X}_{i-1}$ and V^w by $\mathtt{X}_i$. Above, $write(t)$ denotes the set of variables that are written by $guard(t)$.

5.2 Trace Realization Constraints

Next, we describe how an admissible realization for a given trace with uncertainty $\mathbf{ue}$ is encoded. To this end, additional variables are needed. Let $\mathbf{ue} = \{ue_1, \ldots, ue_m\}$ such that $ue_i = \langle \text{ID}, conf, \text{LA}, \text{TS}, \alpha \rangle$ for each $1 \le i \le m$, with $\text{LA} = \{b_1 : p_1, \ldots, b_{N_i} : p_{N_i}\}$. We use the following sets of variables for all i:

(d) a boolean *drop variable* $\mathtt{drop}_{ue_i}$ expressing whether the event is absent in the realization; it must satisfy $\mathtt{drop}_{ue_i} \implies (ue_i.conf < 1)$, i.e., it can only be assigned true for uncertain events with confidence below 1,

(e) an integer *activity variable* $\mathtt{A}_{ue_i}$ that expresses which of the labels $b_1, \ldots, b_{N_i}$ is taken, so it must satisfy $1 \le \mathtt{A}_{ue_i} \le N_i$, and

(f) *trace data variables* $\mathtt{D}_{v,ue_i}$ of suitable type for all $v \in V$ that satisfy either that $\bigvee_{c \in ue.\alpha} \mathtt{D}_{v,ue_i} = c$ if $\alpha(ue)$ is a set, or $l \le \mathtt{D}_{e_i} \le u$ if $\alpha(ue) = [l, u]$ is an interval.

If each uncertain event in $\mathbf{ue}$ has a single, distinct timestamp, we call $\mathbf{ue}$ *sequential*, and assume it is ordered by time as $\langle ue_1, \ldots, ue_n \rangle$. If $\mathbf{ue}$ is not sequential, we need the following additional variables: For all i, $1 \leq i \leq m$:

(g) a *time stamp variable* $\mathtt{T}_{ue_i}$ to express when event ue_i happened, with the constraint $\bigvee_{t \in \mathrm{TS}} \mathtt{T}_{ue_i} = t$ if $\mathrm{TS}(ue_i)$ is a set, or $l \leq \mathtt{T}_{e_i} \leq u$ if $\mathrm{TS}(ue_i) = [l, u]$ is an interval,

(h) an integer *position variable* $\mathtt{P}_{ue_i}$ to fix the position of ue_i in the realization,

(i) an integer *item variable* $\mathtt{L}_j$ that indicates the j-th element in the realization, i.e., $\mathtt{L}_j$ has value $\mathrm{ID}(ue_i)$ if and only if the j-th event in the trace with uncertainty is ue_i; we thus issue the constraint $\bigvee_{i=1}^{m} \mathtt{L}_j = \mathrm{ID}(ue_i)$ to fix the range of $\mathtt{L}_j$, for all $1 \leq j \leq m$.

The formula φ_{trace} consists of the range constraints in (d)-(i), in addition to

$$\bigwedge_{i=1}^{m} \bigwedge_{j=1}^{m} (\mathtt{P}_{ue_i} < \mathtt{P}_{ue_j} \implies \mathtt{T}_{ue_i} \leq \mathtt{T}_{ue_j}) \wedge (\mathtt{T}_{ue_i} < \mathtt{T}_{ue_j} \implies \mathtt{P}_{ue_i} < \mathtt{P}_{ue_j})$$
$$\bigwedge_{i=1}^{m} \bigwedge_{j=1}^{m} \mathtt{L}_i = \mathrm{ID}(ue_j) \iff \mathtt{P}_{ue_j} = i$$

so as to require that, first, the positions assigned to uncertain events by $\mathtt{P}_{ue_j}$ is compatible with the time stamps assigned by $\mathtt{T}_{ue_j}$ and, second, that the $\mathtt{P}_{ue_j}$ variables work as an "inverse function" of the $\mathtt{L}_i$.

5.3 Encoding the Cost Function

To encode the alignment and its cost we use, additionally:

(j) distance variables $\mathtt{d}_{i,j}$ of type integer for $0 \leq i \leq m$ and $0 \leq j \leq n$, where $\mathtt{d}_{i,j}$ is the alignment cost of the prefix $\mathbf{e}|_i$ of the log trace realization $\mathbf{e}$ and prefix $\mathbf{f}|_j$ of the process run $\mathbf{f}$, both of which are yet to be determined.

The search for an optimal alignment is based on a notion of *edit distance*, similar as in [11,6]. More precisely, we assume that the data-aware alignment cost $\kappa(e_i, f_i)$ in Fig. 1 can be encoded using a *distance-based* cost function with *penalty functions* P_L, P_M, and $P_=$ as discussed in Sec. 4.1. Recall that $P_=$ is assumed to be data-aware, i.e., to take into account the mismatching variable assignments between the events in realizations and transition firings in process runs. Intuitively, such functions assess the degree of "closeness" between a process run and a log trace. We assume that there are SMT encodings of these penalty functions that use variables (a)–(i), denoted as $[P_=]_{i,j}$, $[P_M]_j$, and $[P_L]_i$.

Moreover, we assume that there are encodings of the event removal cost function $[\kappa_{\mathbf{ue}}]_i$ and the confidence cost function $[\theta_{\mathbf{ue}}]_i$, defined for the i-th element of the log trace realization. We then consider the following constraints for $i, j > 0$:[1]

$$\mathtt{d}_{0,0} = 0 \qquad \mathtt{d}_{i,0} = \min([P_L]_i \cdot [\theta_{\mathbf{ue}}]_i, [\kappa_{\mathbf{ue}}]_i) + \mathtt{d}_{i-1,0} \qquad \mathtt{d}_{0,j} = [P_M]_j + \mathtt{d}_{0,j-1}$$

$$\mathtt{d}_{i,j} = \min \begin{cases} ite([P_=]_{i,j} = 0, [\theta_{\mathbf{ue}}]_i, [P_=]_{i,j} + [P_=]_{i,j} \cdot [\theta_{\mathbf{ue}}]_i) + \mathtt{d}_{i-1,j-1} \\ [P_L]_i \cdot [\theta_{\mathbf{ue}}]_i + \mathtt{d}_{i-1,j} \\ [\kappa_{\mathbf{ue}}]_i + \mathtt{d}_{i-1,j} \\ [P_M]_j + \mathtt{d}_{i,j-1} \end{cases} \qquad (\varphi_\delta)$$

[1] We assume that P_L is always positive, otherwise, a case distinction using ite is also required in the second line.

This encoding constitutes an *operational way* for computing the cost function represented in Fig. 1, where the components $\kappa_{\mathbf{ue}}$ and θ are distributed to single moves, which at the same time allows us to use the encoding schema based on the edit distance. The inductive case $\mathtt{d}_{i,j}$ is computed so as to locally choose the move with minimal cost. In particular, the first and the second line of the case distinction correspond exactly to the specific instantiation of the expression $\kappa(e_i, f_i) \otimes \theta(e_i, \mathbf{ue})$ exemplified in Sec. 4. For instance, the cost penalty $\kappa(e_i, f_i) \cdot (1 + \theta(e_i, \mathbf{ue}))$ in case $\kappa(e_i, f_i) > 0$ (see Sec. 4) corresponds here, in the ite construct, to the cost penalty $[P_{=}]_{i,j} + [P_{=}]_{i,j} \cdot [\theta_{\mathbf{ue}}]_i$ in the *else* statement. The expression $\mathtt{d}_{m,n}$ encodes then the cost of the complete alignment, which will thus be used as the minimization objective.

The encodings of the penalties, as well as $[\kappa_{\mathbf{ue}}]_i$ and $[\theta_{\mathbf{ue}}]_i$, also depend on the choice of the respective functions. For those exemplified in Sec. 4, one can define $[\kappa_{\mathbf{ue}}]_i$ as a (nested) case distinction on the element from $\mathbf{ue}$ that is chosen for the i-th position (represented with variable L_i – see Sec. 5.2):

$$\begin{aligned}[\kappa_{\mathbf{ue}}]_i = & ite(L_i = \text{ID}(ue_1) \wedge \mathtt{drop}_{ue_1}, conf(ue_1), \dots \\ & ite(L_i = \text{ID}(ue_m) \wedge \mathtt{drop}_{ue_m}, conf(ue_m), \infty) \dots)\end{aligned}$$

A similar case distinction can be done for $[\theta_{\mathbf{ue}}]_i$, also exemplified in Sec. 4.

5.4 Solving and Decoding

We use an SMT solver to obtain a satisfying assignment ν for the following constrained optimization problem:

$$\varphi_{run} \wedge \varphi_{trace} \wedge \varphi_{\delta} \quad \text{minimizing} \quad \mathtt{d}_{m,n} \tag{Φ}$$

For a satisfying assignment ν for (Φ), we construct the process run $\mathbf{f}_\nu = \langle f_1, \dots, f_n \rangle$ where $f_i = (t_{\nu(\mathtt{s}_i)}, \beta_i)$, assuming that the set of transitions T consists of $t_1, \dots, t_{|T|}$ in the ordering already used for the encoding. The transition variable assignment β_i is obtained as follows: Let the state variable assignments α_j, $0 \leq j \leq n$, be given by $\alpha_j(v) = \nu(\mathtt{X}_{j,v})$ for all $v \in V$. Then, $\beta_i(v^r) = \alpha_{i-1}(v)$ and $\beta_i(v^w) = \alpha_i(v)$ for all $v \in V$. Moreover, we construct a realization $\mathbf{e}_\nu = \langle e_1, \dots, e_k \rangle$ by ordering the events in $\mathbf{ue}$ according to $\mathtt{T}_{ue_i}$, dropping those where $\mathtt{drop}_{ue_i}$ is true, and fixing the label and data values to $\mathtt{A}_{ue_i}$ and $\mathtt{D}_{ue_i}$, respectively. Finally, let the (partial) alignments $\gamma_{i,j}$ be defined as follows, for $i, j > 0$:

$$\gamma_{0,0} = \epsilon \qquad \gamma_{0,j+1} = \gamma_{0,j} \cdot (\gg, f_{j+1})$$

$$\gamma_{i+1,0} = \begin{cases} \gamma_{i,0} \cdot (e_{i+1}, \gg) & \text{if } \nu(\delta_{i+1,0}) = \nu([P_L]_{i+1} \cdot [\theta_{\mathbf{ue}}]_{i+1} + \delta_{i,0}) \\ \gamma_{i,0} & \text{if } \nu(\delta_{i+1,0}) = \nu([\kappa_{\mathbf{ue}}]_{i+1} + \delta_{i,0}) \end{cases}$$

$$\gamma_{i+1,j+1} = \begin{cases} \gamma_{i,j+1} \cdot (e_{i+1}, \gg) & \text{if } \nu(\delta_{i+1,j+1}) = \nu([P_L]_{i+1} \cdot [\theta_{\mathbf{ue}}]_{i+1} + \delta_{i,j+1}) \\ \gamma_{i,j+1} & \text{if } \nu(\delta_{i+1,j+1}) = \nu([\kappa_{\mathbf{ue}}]_{i+1} + \delta_{i,j+1}) \\ \gamma_{i+1,j} \cdot (\gg, f_{j+1}) & \text{if otherwise } \nu(\delta_{i+1,j+1}) = \nu([P_M]_{j+1} + \delta_{i+1,j}) \\ \gamma_{i,j} \cdot (e_{i+1}, f_{j+1}) & \text{otherwise} \end{cases}$$

5.5 Correctness

The next results state that the constructed alignment satisfies our conformance checking task as in Def. 7. The formal proofs are omitted for reasons of space, but can be found in the extended version [12]. It is however easy to see that our encoding matches the same definitions as in Sec. 2 and 3, and the cost functions in Sec. 4.

Lemma 2. *For any satisfying assignment ν to (Φ), (a) $\mathbf{f}_\nu$ is a process run, and (b) $\mathbf{e}_\nu$ is a realization of $\boldsymbol{u}\mathbf{e}$.*

This lemma shows that the decoding provides both a valid process run and a trace realization. Next we demonstrate that the decoded alignment is optimal.

Theorem 1. *Let $\mathcal{N}$ be a DPN, $\boldsymbol{u}\mathbf{e}$ a log trace with uncertainty and ν a solution to (Φ) as in Sec. 5.4. Then $\gamma_{m,n}$ is an optimal alignment for $\boldsymbol{u}\mathbf{e}$.*

Moreover, as explained in Sec. 4 (after Def. 7), we can easily capture the additional task of computing the lower-bound on the optimal cost of alignments of realizations for a given trace with uncertainty, as considered in [17]. By taking advantage of the modularity of our framework, this simply amounts to set $\kappa_{\mathbf{ue}} = 0$ and $\kappa(e_i, f_i) \otimes \theta(e_i, \mathbf{ue}) = \kappa(e_i, f_i)$, thus ignoring all confidence values specified in $\mathbf{ue}$. This allows us to freely select, without any penalty, the realization of $\mathbf{ue}$ that has the minimal alignment cost.

Lemma 3. *For $\mathcal{N}$, $\boldsymbol{u}\mathbf{e}$ as above and $\gamma_{m,n}$ the alignment decoded from a satisfying assignment ν for (Φ) as in Sec. 5.4, there is no realization $\mathbf{e}$ of $\boldsymbol{u}\mathbf{e}$ and alignment γ for $\mathbf{e}$ such that $\kappa(\gamma) < \kappa(\gamma_{m,n})$.*

Note that in contrast to the approach in [17], our approach entirely avoids any explicit construction of realizations, which is a huge benefit for the overall performance.

5.6 Implementation

As a proof of concept, the uncertainty conformance checking approach described in this paper was implemented in `cocomot` – a Python command line tool that was originally designed for data-aware conformance checking without uncertainties [11]. It uses `pm4py` (https://pm4py.fit.fraunhofer.de/) to perform parsing tasks, and the SMT solvers Yices 2 [10] and Z3 [9].

The tool takes as input two files: a DPN in `.pnml` format and a log in `.xes`, specified using the XES extension for uncertain data described in [16]. The command line option `-u` triggers the use of the uncertainty module, and the tool outputs the optimal alignment as well as its cost. Based on the the encoding in Sec. 5, the tool employs the two cost functions mentioned in Ex. 6 to achieve two different tasks: Using the first cost function that takes confidence values into account, the cost of the optimal alignment can be interpreted as an expectation value of the best alignment cost for all realizations (parameter `-u fit`). Using the second cost function, a lower bound on the cost of the optimal alignment among all realizations is computed (parameter `-u min`). More information on the tool usage, the format for specifying uncertain logs, execution options

and further details, together with the source code, can be found on the tool website: https://github.com/bytekid/cocomot.

Although the presented encoding shows that the overall theoretical complexity of our approach does not change with respect to the one reported in [11] (that is, the problem of finding the optimal alignment for logs with uncertainty is NP-complete), experimental evaluations are required so as to assess the feasibility of the encoding in practical scenarios. More specifically, we plan to enrich publicly available logs for multi-perspective conformance checking [15] with uncertainty information, as done in [17].

6 Conclusions

In this work we have proposed an extension of the foundational framework for alignment-based conformance checking of data-aware processes studied in [11], to support logs with different types of uncertainties in events, timestamps, activities and other attributes. To account for all possible combinations of uncertainties in a trace, we rely on a notion of realization to fix one of its possible *certain* variants. However, given that there are potentially infinitely many realizations, performing the conformance checking task on each of them is not feasible.

To attack this problem, we considered a version the conformance checking task aimed at searching for the best alignment among all possible realizations. This has been achieved by introducing an involved cost model that incorporates traditional alignment-related penalties together with extra costs accounting for the selection of specific realizations. Although these cost components are presented as arbitrary and can in fact be tailored to specific settings and assumptions, we have provided a concrete instantiation and its corresponding encoding.

Thanks to the modularity of our conformance cost definition, we have also shown how we can accommodate different conformance checking tasks for logs with uncertainty, including those studied in the literature [17].

The theoretical underpinning of our approach is SMT solving. Our work is the first one to employ techniques based on *satisfiability* of formulae modulo suitable logical theories for solving data-aware conformance checking tasks with uncertainty, and to leverage well-established solvers to handle them. The approach was implemented in the `cocomot` tool.

In future work, we plan to investigate further more involved notions of uncertain logs, and conduct an experimental evaluation of our approach and implementation. To this end, instead of considering artificially generated logs, one first step is to compile a benchmark for data-aware conformance checking of uncertain logs, which is currently not available.

Acknowledgments. This research has been partially supported by the UNIBZ projects VERBA, MENS, WineID, SMART-APP and by the PRIN 2020 project PINPOINT.

References

1. A. Alman, F. M. Maggi, M. Montali, and R. Peñaloza. Probabilistic declarative process mining. *Inf. Syst.*, 2022.

2. C. Barrett, P. Fontaine, and C. Tinelli. The SMT-LIB Standard: Version 2.6. Technical report, Available at: http://smtlib.cs.uiowa.edu/language.shtml, 2018.
3. C. W. Barrett and C. Tinelli. Satisfiability modulo theories. In *Handbook of Model Checking*, pages 305–343. Springer, 2018.
4. G. Bergami, F. M. Maggi, M. Montali, and R. Peñaloza. Probabilistic trace alignment. In *Proc. of ICPM 2021*, pages 9–16. IEEE, 2021.
5. M. Boltenhagen, T. Chatain, and J. Carmona. Encoding conformance checking artefacts in SAT. In *Proc. BPM Workshops 2019*, pages 160–171, 2019.
6. M. Boltenhagen, T. Chatain, and J. Carmona. Optimized SAT encoding of conformance checking artefacts. *Computing*, 103:29–50, 2021.
7. J. Carmona, B. F. van Dongen, A. Solti, and M. Weidlich. *Conformance Checking - Relating Processes and Models*. Springer, 2018.
8. F. Chesani, P. Mello, R. De Masellis, C. Di Francescomarino, C. Ghidini, M. Montali, and S. Tessaris. Compliance in business processes with incomplete information and time constraints: a general framework based on abductive reasoning. *Fundam. Informaticae*, 161(1-2):75–111, 2018.
9. L. de Moura and N. Bjørner. Z3: an efficient SMT solver. In *Proc. TACAS 2008*, pages 337–340, 2008.
10. B. Dutertre. Yices 2.2. In *Proc. CAV 2014*, pages 737–744, 2014.
11. P. Felli, A. Gianola, M. Montali, A. Rivkin, and S. Winkler. Cocomot: Conformance checking of multi-perspective processes via SMT. In *Proc. BPM 2021*, pages 217–234. Springer, 2021.
12. P. Felli, A. Gianola, M. Montali, A. Rivkin, and S. Winkler. Conformance checking with uncertainty via SMT (extended version). Technical report, arXiv.org, 2022. Available at: https://arxiv.org/abs/2206.07461v1.
13. S. J. J. Leemans, W. M. P. van der Aalst, T. Brockhoff, and A. Polyvyanyy. Stochastic process mining: Earth movers' stochastic conformance. *Inf. Syst.*, 102:101724, 2021.
14. F. Mannhardt. *Multi-perspective Process Mining*. PhD thesis, Technical University of Eindhoven, 2018.
15. F. Mannhardt, M. de Leoni, H. Reijers, and W. van der Aalst. Balanced multi-perspective checking of process conformance. *Computing*, 98(4):407–437, 2016.
16. M. Pegoraro. Process mining on uncertain event data (extended abstract). In *Proc. ICPM-D 2021*, pages 1–2. CEUR, 2021.
17. M. Pegoraro, M. S. Uysal, and W. M. P. van der Aalst. Conformance checking over uncertain event data. *Inf. Syst.*, 102:101810, 2021.
18. A. Polyvyanyy and A. A. Kalenkova. Conformance checking of partially matching processes: An entropy-based approach. *Inf. Syst.*, 106:101720, 2022.
19. R. Sebastiani and S. Tomasi. Optimization modulo theories with linear rational costs. *ACM Trans. Comput. Log.*, 16(2):12:1–12:43, 2015.
20. W. M. P. van der Aalst *et al.* Process mining manifesto. In *Proc. of BPM Workshops 2011*, pages 169–194. Springer, 2011.
21. M. T. Wynn and S. W. Sadiq. Responsible process mining - A data quality perspective. In *Proc. BPM 2020*, pages 10–15. Springer, 2019.