2015 IEEE Symposium Series on Computational Intelligence

Constructing Probable Explanations of Nonconformity: A Data-aware and History-based Approach

Mahdi Alizadeh
Eindhoven University of Technology
Email: m.alizadeh@tue.nl

Massimiliano de Leoni
Eindhoven University of Technology
Email: m.d.leoni@tue.nl

Nicola Zannone
Eindhoven University of Technology
Email: n.zannone@tue.nl

***Abstract*—Auditing the execution of business processes is becoming a critical issue for organizations. Conformance checking has been proposed as a viable approach to analyze process executions with respect to a process model. In particular, alignments provide a robust approach to conformance checking in that they are able to pinpoint the causes of nonconformity. Alignment-based techniques usually rely on a predefined cost function that assigns a cost to every possible deviation. Defining such a cost function, however, is not trivial and is prone to imperfection that can result in inaccurate diagnostic information. This paper proposes an alignment-based approach to construct probable explanations of nonconformity. In particular, we show how cost functions can be automatically computed based on historical logging data and taking into account multiple process perspectives. We implemented our approach as a plug-in of the ProM framework. Experimental results show that our approach provides more accurate diagnostics compared to existing alignment-based techniques.**

I. Introduction

Risk management is a critical part of the business processes of any organization. Several regulations and guidelines (e.g., Sarbanes-Oxley Act, Basel III, COBIT) have been defined for operational risk management. These regulations require organizations to implement internal controls and ensure a transparent audit trail of IT-related activities. In particular, organizations are required to monitor their internal processes in order to detect operations that violate the prescribed behavior.

Conformance checking [1] has been proposed to analyze the events recorded by an IT system against a prescribed behavior described as a process model. In particular, alignments [2] provide a robust approach to conformance checking by allowing the detection and analysis of nonconformity between the observed and prescribed behavior. Alignment-based techniques construct alignments based on a cost function that assigns a cost to every possible deviation. In particular, they construct alignments that have the least cost with respect to the employed cost function. As a consequence, the quality of diagnostic information provided by alignment-based techniques depends on the cost function used to construct the alignments.

Alignment-based techniques usually expect cost functions to be defined by business analysts based on their background knowledge and beliefs. In particular, business analysts have to define the cost of every possible deviation that can occur, and this cost is often specific to the purpose of the intended analysis and only based on the activities executed. The definition of such a cost function is, thus, fully based on human judgment and prone to imperfections. These imperfections ultimately lead to alignments that are optimal, according to the provided cost function, but that may not provide accurate diagnostics.

To illustrate these issues, consider the process model in Fig. 1. The model describes a process for handling credit requests at a financial institution. After a client requests a loan (a), the client's financial data are verified (b). In the case the result of the verification is negative, the client is informed about the result (f or g) and the process terminates. In particular, if the initial of the client's name is A-L, activity g is performed; otherwise activity f is performed. If the verification is positive, either advanced (c) or simple (d) assessment is performed depending on the amount of the requested loan. If the request is approved, a credit loan is opened (h) and the client is informed about the result (f or g). If the assessment is negative, the client can renegotiate the loan (e). Several data attributes might be associated with these activities. During credit request, the client's name (R) and amount of the requested loan (A) should be provided. The result of the verification (V) and assessment (D) are also recorded by the system. After renegotiation of the request, the amount of the loan is updated.

Suppose an analyst has to verify the compliance of the process execution represented by the following event trace[1] with the process model in Fig. 1 and determine the causes of nonconformity if the trace is not compliant.

$$\sigma_1 = \langle (b, \{V = true\}), (h, \{\}), (g, \{\}) \rangle$$

It is easy to observe that σ_1 does not conform to the process model, i.e. the trace does not correspond to any path allowed by the model. Activity a was not executed. Moreover, a credit loan was opened without an assessment of the request. These deviations can have various explanations. A possible explanation is that the request was rejected and, thus, h should not have been performed. Another possible explanation is that the assessment (c or d) was skipped and the credit loan was opened correctly.

Which explanation is constructed by alignment-based techniques depends on the cost assigned to deviations. For instance, suppose that, for a certain analysis, the analyst assigns a cost to the suppression of c and d (i.e., these activities are not executed when they should have) higher than the cost of inserting h (i.e., executing the activity when it should not have been executed). In this case, the first explanation is returned. Later, the analyst

[1]Notation $(act, \{attr_1 = val_1, \ldots, attr_n = val_n\})$ is used to denote the occurrence of activity act in which data attributes $attr_1, \ldots, attr_n$ are assigned values $val_1, \ldots, val_n$ respectively.

978-1-4799-7560-0/15 $31.00 © 2015 IEEE
DOI 10.1109/SSCI.2015.194

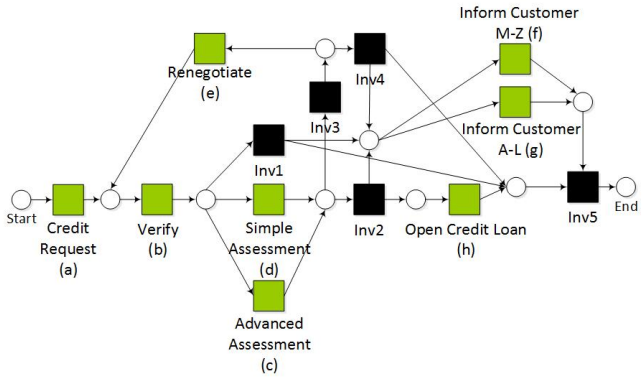


Figure 1: A process model for handling credit requests. Green boxes represent the transitions associated with process activities while black boxes represent invisible transitions. The text below the activities represents the label, which is shortened with a single letter as indicated inside the transitions.

defines a new cost function for a new type of analysis, this time assigning a lower cost to the suppression of c or d. In this case, the second explanation is returned, making the analyst uncertain on what actually happened.

This example reveals that determining what actually happened is independent from the purpose of the analysis. We advocate that the analysis of event traces should be divided into two phases: first nonconformity should be identified along with their root causes based on objective factors; then nonconformity should be analyzed with respect to the purpose of the analysis. This work focuses on the identification and explanations of nonconformity.

In our previous work [3], we proposed an alternative way to define a cost function for the detection of nonconformity, where the human judgment is put aside and only objective factors are considered. The cost function is automatically constructed by looking at the logging data and, more specifically, at the past process executions that are compliant with the process model. The intuition behind is that one should look at the past history of process executions and learn from it how the process is typically executed. A probable explanation of nonconformity of a certain process execution can be obtained by analyzing the behavior observed for such a process execution in each and every state against the behavior observed for other conforming traces when they were in the same state.

However, in our previous work, we used a concept of state that only considers the activities that are executed. Suppose, in the example above, that the analyst has to determine whether advanced (c) or simple (d) assessment should have been performed. Knowledge of the amount of the requested loan provides an indication of the type of assessment that should have been executed. In fact, an advanced assessment is typically performed when a large amount is requested, whereas requests for a low amount are typically verified using a simple assessment. Thus, considering the data attributes manipulated by activities provides additional information about which activities could have been executed and, thus, can help determine more accurate explanations of nonconformity.

This paper extends the work in [3] by taking into account multiple process perspectives to compute a cost function for the construction of probable explanations of nonconformity. In particular, we extend the concept of state to incorporate the data attributes manipulated by activities, improving the accuracy of the cost function. It is worth noting that information about data attributes is often not provided with the process model. Therefore, our approach constructs alignments using only the activities executed, whereas data attributes are used to define a cost function that enables the construction of alignments which give probable explanations of nonconformity.

The new technique has been implemented as a plug-in of the ProM framework. An evaluation of our approach shows that it provides more accurate explanations of nonconformity compared to existing alignment-based techniques.

The remainder of the paper is structured as follows. The next section presents the basic notation and background about alignments. Section III introduces the notion of state representation. Section IV presents our approach to compute the cost of deviations based on logging data and construct probable alignments. Section V presents the results of our experiments. Finally, Section VI discusses related work, and Section VII concludes the paper with directions for future work.

II. Background and Notation

Process models describe how activities in the process must be performed. In this work, we represent process models in the form of Labeled Petri net. Our approach is extendable to any modeling language for which a translation to Labeled Petri [4] nets is available.

Definition 1 (Labeled Petri Net)**.** *A Labeled Petri net is a tuple* $(P, T, F, A, \ell, m_i, m_f)$ *where*

- P *is a set of places;*
- T *is a set of transitions;*
- $F \subseteq (P \times T) \cup (T \times P)$ *is the flow relation between places and transitions (and between transitions and places);*
- A *is the set of labels for transitions;*
- $\ell : T \rightarrow A$ *is a function that associates a label with every transition in* T*;*
- m_i *is the initial marking;*
- m_f *is the final marking.*

Hereafter, for simplicity, the term Petri net is used to refer to Labeled Petri Net. In a Petri net, transitions represent events and places represent states. Multiple transitions can have the same label. Such transitions are called duplicate transitions. We distinguish two types of transitions, namely invisible and visible transitions. Visible transitions are labeled with activity names. Invisible transitions are used for routing purposes or related to activities that are not observed by the IT system. Given a Petri net N, the set of activity labels associated with invisible transitions is denoted with $\mathsf{Inv}_N \subseteq A$.

A marking is a multiset of tokens and represents a state of the Petri net. Tokens reside in places. A transition is enabled if at least one token exists in each input place of the transition. By executing (i.e., firing) a transition, a token is consumed from each input places and a token is produced for each of its output places. A complete firing sequence is a sequence of transitions from the initial marking to the final marking. The

set of all complete firing sequences of a Petri net N is denoted by Ψ_N. Hereafter, we call process trace a firing sequence in which transitions are mapped to their activity label.

Definition 2 (Event, Event Trace, Event Log)**.** *Let $N = (P, T, F, A, \ell, m_i, m_f)$ be a labeled Petri net and $\mathsf{Inv}_N \subseteq A$ the set of invisible transitions in N. Let V be a set of data attributes. Let us denote the domain of a data attribute $v \in V$ with $U(v)$ and the union of all attribute domains with $\mathcal{U}$, i.e. $\mathcal{U} = \cup_{v \in V} U(v)$. An event $e = (a, \varphi_e)$ consists of an executed activity $a \in A \setminus \mathsf{Inv}_N$ and a partial function φ_e that assigns values to attributes: $\varphi_e \in V \nrightarrow \mathcal{U}$ s.t. $\forall v \in dom(\varphi_e)$ $\varphi_e(v) \in U(v)$.[2] The set of events is denoted by $\mathcal{E}$. An event trace $\sigma \in \mathcal{E}^*$ is a sequence of events. An event log $\mathcal{L} \in \mathbb{B}(\mathcal{E}^*)$ is a multiset of event traces.*[3]

Given an event $e = (a, \varphi_e)$, we use $act(e)$ to denote the activity label associated to e, i.e. $act(e) = a$. This notation extends to event traces. Given an event trace $\sigma = \langle e_1, \ldots, e_n \rangle \in \mathcal{E}^*$, $act(\sigma)$ denotes the sequence of activities obtained from the projection of the events in σ to their activity label, i.e. $act(\sigma) = \langle act(e_1), \ldots, act(e_n) \rangle$.

Not all event traces in an event log can be reproduced by a Petri net, i.e. not all event traces may correspond to a process trace. If an event trace perfectly fits the net, each "move" in the event trace (i.e., an event observed in the trace) can be mimicked by a "move" in the model (i.e., an activity in the net). After all events in the event trace are mimicked, the net reaches its final marking. In cases where deviations occur, some moves in the event trace cannot be mimicked by the net or vice versa. We explicitly denote "no move" by $\gg$.

Definition 3 (Legal move)**.** *Let $N = (P, T, F, A, \ell, m_i, m_f)$ be a Petri net, $S_L = \mathcal{E} \cup \{\gg\}$ and $S_M = A \cup \{\gg\}$. A legal move is a pair $(m_L, m_M) \in (S_L \times S_M) \setminus (\gg, \gg)$ such that*

- *(m_L, m_M) is a* synchronous move *if $m_L \in \mathcal{E}$, $m_M \in A$ and $act(m_L) = m_M$,*
- *(m_L, m_M) is a* move on log *if $m_L \in \mathcal{E}$ and $m_M = \gg$,*
- *(m_L, m_M) is a* move on model *if $m_L = \gg$ and $m_M \in A$.*

Σ_N denotes the set of legal moves for a Petri net N.

A sequence σ' is a prefix of sequence σ'' if there exists a sequence σ''' such that $\sigma'' = \sigma' \oplus \sigma'''$, where $\oplus$ denotes the concatenation operator. Hereafter, $\sigma' \in \mathsf{prefix}(\sigma'')$ is used to denote such a relation.

Definition 4 (Alignment)**.** *Let Σ_N be the set of legal moves for a Petri net $N = (P, T, F, A, \ell, m_i, m_f)$. An alignment of an event trace σ and N is a sequence $\gamma \in \Sigma_N^*$ such that, ignoring all occurrences of $\gg$, the projection on the first element yields σ and the projection on the second element yields a process trace $\langle a_1, \ldots, a_n \rangle$ such that there exists a firing sequence $\psi' = \langle t_1, \ldots, t_n \rangle \in \mathsf{prefix}(\psi)$ for some $\psi \in \Psi_N$ where, for every $1 \leq i \leq n$, $\ell(t_i) = a_i$. If $\psi' \in \Psi_N$, γ is called a* complete alignment *of σ and N.*

Fig. 2 shows four alignments of event trace σ_1 and the net in Fig. 1. The first and second columns in the alignment, ignoring $\gg$, show the event trace and the process trace to which the event trace is aligned respectively.

[2]We denote the domain of a function f with $dom(f)$.

[3]$\mathbb{B}(X)$ represents the set of all multisets over X.

Event Trace	Process Trace
$\gg$	a
b, {V=true}	b
$\gg$	c
$\gg$	Inv_2
h, {}	h
g, {}	g
$\gg$	Inv_5

(a)

Event Trace	Process Trace
$\gg$	a
b, {V=true}	b
$\gg$	d
$\gg$	Inv_2
h, {}	h
g, {}	g
$\gg$	Inv_5

(b)

Event Trace	Process Trace
$\gg$	a
b, {V=true}	b
$\gg$	Inv_1
h, {}	$\gg$
g, {}	$\gg$
$\gg$	f
$\gg$	Inv_5

(c)

Event Trace	Process Trace
$\gg$	a
b, {V=true}	b
$\gg$	Inv_1
h, {}	$\gg$
g, {}	g
$\gg$	Inv_5

(d)

Figure 2: Examples of complete alignments of the net in Fig. 1 and $\sigma_1 = \langle (b, \{V = true\}), (h, \{\}), (g, \{\}) \rangle$

As shown in Fig. 2, there can be several (possibly infinite) explanations of nonconformity. The quality of an alignment is measured based on a cost function. A cost function defines the cost of every move that can occur in an alignment. The cost of an alignment is the sum of the cost of the moves in the alignment. An *optimal alignment* is an alignment that has the least cost according to the cost function.

As an example, consider a cost function that penalizes all moves on model for visible transitions and moves on log equally. This cost function does not penalize synchronous moves and moves on model for invisible transitions. If moves on model for invisible transitions are ignored, the alignments in Fig. 2a and Fig. 2b have two moves on model, the alignment in Fig. 2c has two moves on log and two moves on model, and the alignment in Fig. 2d has one move on log and one move on model. Thus, according to our example cost function, the alignments in Fig. 2a, Fig. 2b, and Fig. 2d are three optimal alignments of σ_1 and the process model in Fig. 1.

III. State Representation

The goal of this work is to construct alignments that give probable explanations of nonconformity based on the information in historical logging data. To this end, we aim at the definition of a cost function that accounts for the probability of executing (or never executing) a certain activity in the current state of the process execution based on the past process executions. In this section we define the state representation used to define such a cost function. In the next section, we define the cost function and present our approach for constructing probable alignments.

The state of a process represents its execution at a given time, i.e. the performed activities along with the value of data attributes. A state is formally defined as follows:

Definition 5 (State)**.** *Let A be the set of activities, V the set of attributes and $\mathcal{U}$ the attributes' domain. A state s for an event trace σ is a pair $(act(\sigma), \varphi_s)$ where φ_s denotes a function that associates a value to each attribute, i.e. $\varphi_s : V \to \mathcal{U} \cup \{\perp\}$ such that for all $v \in V$ $\varphi_s(v) \in U(v) \cup \{\perp\}$ (where $\perp$ indicates undefined). The initial state is denoted $s_I = (\langle\rangle, \varphi_I)$ where φ_I is the initial assignment of values to attributes.*

Trace	#
(a,{R='bob',A=1000}),(b,{V=false}),(g,{})	200
(a,{R='bob',A=1000}),(b,{V=true}),(d,{D=true}),(g,{}),(h,{})	600
(a,{R='bob',A=4100}),(b,{V=true}),(d,{D=false}),(g,{})	100
(a,{R='bob',A=4200}),(b,{V=true}),(d,{D=false}),(g,{})	100
(a,{R='bob',A=4500}),(b,{V=true}),(d,{D=false}),(g,{})	100
(a,{R="tim",A=5500}),(b,{V=false}),(f,{})	500
(a,{R='tim',A=5500}),(b,{V=true}),(c,{D=true}),(f,{}),(h,{})	200
(a,{R='tim',A=5500}),(b,{V=true}),(c,{D=false}),(f,{})	50
(a,{R='tim',A=5500}),(b,{V=true}),(c,{D=false}),(e,A=1000), (b,{V=false}),(f,{})	150

(a) Event traces

State						Next Activity	#
ID	Executed Activities	Data Attributes					
		R	A	V	D		
s_1	$\langle\rangle$	$\perp$	$\perp$	$\perp$	$\perp$	a	2000
s_2	$\langle a\rangle$	bob	1000	$\perp$	$\perp$	b	800
s_3	$\langle a\rangle$	bob	4100	$\perp$	$\perp$	b	100
s_4	$\langle a\rangle$	bob	4200	$\perp$	$\perp$	b	100
s_5	$\langle a\rangle$	bob	4500	$\perp$	$\perp$	b	100
s_6	$\langle a\rangle$	tim	5500	$\perp$	$\perp$	b	900
s_7	$\langle a,b\rangle$	bob	1000	false	$\perp$	g	200
s_8	$\langle a,b\rangle$	bob	1000	true	$\perp$	d	600
s_9	$\langle a,b\rangle$	bob	4100	true	$\perp$	d	100
s_{10}	$\langle a,b\rangle$	bob	4200	true	$\perp$	d	100
s_{11}	$\langle a,b\rangle$	bob	4500	true	$\perp$	d	100
s_{12}	$\langle a,b\rangle$	tim	5500	true	$\perp$	c	400
s_{13}	$\langle a,b\rangle$	tim	5500	false	$\perp$	f	500
s_{14}	$\langle a,b,g\rangle$	bob	1000	false	$\perp$	-	200
s_{15}	$\langle a,b,d\rangle$	bob	1000	true	true	g	600
...	...	...	...	...	...	...	...

(b) State representation

Figure 3: Sample event traces and their state representation

The initial attribute assignment φ_I represents the value of the attributes in V before the process is executed. Moreover, φ_I comprises the value of *case attributes*, i.e. attributes that characterize the event trace as a whole. In some cases, a value may not be (initially) defined for some attributes. We use symbol "$\perp$" to denote the undefined value.

The execution of an event trace changes the state of the process. We first define a state transition, which is a change of one state to another state due to the effect of an event, and then we extend this definition to event traces.

Definition 6 (State Transition)**.** *Let V be the set of attributes. Given an event $e = (a, \varphi_e)$ and the state $s = (act(\sigma), \varphi_s)$ for an event trace σ, e transforms s into a state $s' = (act(\sigma'), \varphi_{s'})$ such that $\sigma' = \sigma \oplus a$ and for every $v \in V$*

$$\varphi_{s'}(v) = \begin{cases} \varphi_e(v) & \text{if } v \in dom(\varphi_e) \\ \varphi_s(v) & otherwise \end{cases} \quad (1)$$

We denote $s \xrightarrow{e} s'$ the state transition given by e.

Intuitively, (1) states that an event can update some data attributes, leaving the other attributes in φ_s unchanged.

Definition 7 (Trace Execution)**.** *Given an event trace $\sigma = \langle e_1, ..., e_n\rangle \in \mathcal{E}^*$, σ transforms the initial state s_I into a state s if there exist states $s_0, s_1, \ldots, s_n$ such that*

$$s_I = s_0 \xrightarrow{e_1} s_1 \xrightarrow{e_2} \ldots \xrightarrow{e_n} s_n = s$$

We denote $\mathsf{state}(\sigma)$ *the state yielded by an event trace σ.*

Alignments are constructed based on the probability of performing (or never performing) a certain activity when the process is in a given state. To this end, we determine the states that are reached by executing the (prefix) event traces in the historical logging data (i.e., past process executions) together with their occurrence.

Example 1. *Fig. 3a shows a sample of event traces in the historical logging data for the net in Fig. 1 along with their occurrence. The states yielded by the prefixes of these traces, their occurrence and the activity executed after reaching that state are shown in Fig. 3b. Data attributes are initialized with the undefined value ($\perp$). Consider the (prefix) event trace $\sigma_2 = \langle(a, \{R = \text{'bob'}, A = 1000\}), (b, \{V = true\})\rangle$. This trace yields a state $s_{\sigma_2} = (\langle a, b\rangle, \varphi)$ s.t. $\varphi(R) = \text{'bob'}, \varphi(A) = 1000, \varphi(V) = true, \varphi(D) = \perp$. Based on the traces in Fig. 3a, s_{σ_2} was reached 600 times.*

We have now the machinery to analyze historical logging data. However, to compute probable explanations of nonconformity we also have to determine the state in which the process is supposed to be after the execution of a given sequence of events. Essentially, we have to determine the state yielded by an alignment. The projection of an alignment over the net gives us only the sequence of activities that were performed. We also need to determine the value of data attributes in the state yielded by the alignment.

However, this information might be missing. For instance, moves on model, which represent activities that are assumed to be executed, do not provide any information about the data attributes that are expected to be updated by the activity. The problem of missing attributes should be taken into account when the state of the process is determined.

Example 2. *Consider state $s = (\langle a, b, d, e\rangle, \varphi)$ s.t. $\varphi(V) = true$ (all other data attributes are equal to $\perp$) and the move on model $(\gg, b)$. If the problem of missing data attributes is not considered, we simply obtain state $s' = (\langle a, b, d, e, b\rangle, \varphi')$ with $\varphi' = \varphi$, i.e. the value of V is not updated. However, by executing activity b the value of attribute V, indicating the result of the verification, might be updated to a new value and, thus, the process could be in a state in which V could be either $true$ or $false$.*

As shown in the example above, missing data attributes introduces uncertainty on the reached state: different states could have been reached. To capture this, we introduce the notion of *state subsumption*. In particular, state subsumption is used to determine the possible states of the process that could have been yielded by a (non-fitting) event trace.

Definition 8 (State Subsumption)**.** *Given two states $s = (r_s, \varphi_s)$ and $s' = (r_{s'}, \varphi_{s'})$, we say that s subsumes s', denoted $s \succ s'$, if and only if (i) $r_s = r_{s'}$ and (ii) for all $v \in V$ s.t. $\varphi_s(v) \neq \perp$ $\varphi_{s'}(v) = \varphi_s(v)$.*

Let us provide an explanatory example:

Example 3. *Consider a state $s = (\langle a, b\rangle, \varphi)$ s.t. $\varphi(R) = \perp$, $\varphi(A) = \perp$, $\varphi(V) = true$, $\varphi(D) = \perp$. Given the states in Fig. 3b, state s subsumes states $s_8, \ldots, s_{12}$. Thus, we can conclude that there are 1300 event traces in Fig. 3a whose prefix yields a state subsumed by s.*

Now we show how a trace can be extracted from an

alignment. Hereafter, we use notation $\bar{\varphi}_a$ to indicate the set of attributes that are typically updated by an activity a. Such a set can be defined by a statistical analysis of the events in historical logging data, for instance by setting a threshold on the percentage of occurrences in which an attribute is updated by an activity.

Definition 9 (From Alignments to Event Traces)**.** *Let Σ_N be the set of legal moves for a Petri net N, Inv_N the set of invisible transitions in N, $\mathcal{E}$ the set of events and V the set of attributes. We define a function $\alpha : \Sigma_N^* \rightarrow \mathcal{E}^*$ that transforms an alignment into an event trace as follows:*

$$\alpha(\langle\rangle) = \langle\rangle$$
$$\alpha(\gamma \oplus (m_L, m_M)) = \begin{cases} \alpha(\gamma) & m_M \in \mathsf{Inv}_N \cup \{\gg\} \\ \alpha(\gamma) \oplus (m_M, \varphi_{(m_L,m_M)}) & otherwise \end{cases} \quad (2)$$

where the attribute assignment $\varphi_{(m_L,m_M)}$ in the event obtained from move (m_L, m_M) is defined for every $v \in V$ as follow:

$$\varphi_{(m_L,m_M)}(v) = \begin{cases} \varphi_{m_L}(v) & act(m_L) = m_M \text{ and } v \in dom(\varphi_{m_L}) \\ \perp & act(m_L) = m_M \text{ and } v \notin dom(\varphi_{m_L}) \text{ and } v \in \bar{\varphi}_{m_M} \\ \perp & m_L = \gg \text{ and } v \in \bar{\varphi}_{m_M} \end{cases} \quad (3)$$

Intuitively, (2) projects an alignment over the process model and (3) determines the data attributes updated by the event reconstructed from a move. In particular, (3) assigns the undefined value ($\perp$) to missing data attributes.

IV. Construction of probable alignments

This section presents our approach to construct probable alignments, namely alignments between a process model and a log trace that give likely explanations of deviations based on objective facts, i.e. the historical logging data. To construct these alignments, we use the A-star algorithm [5]. Section IV-A discusses the basic foundation of the A-star algorithm and how it can be employed to build these alignments, whereas Section IV-B focuses on how to automatically define a cost function that enables their construction.

A. Usage of A-star for constructing probable alignments

The A-star algorithm [5] aims to find a path in a graph Q from a given *source* node q_0 to any node $q \in Q$ in a target set. Every node q of graph Q is associated with a cost determined by an *evaluation* function $f(q) = g(q) + h(q)$, where

- $g : Q \rightarrow \mathbb{R}_0^+$ is a function that returns the smallest path cost from q_0 to q;
- $h : Q \rightarrow \mathbb{R}_0^+$ is a heuristic function that estimates the path cost from q to its preferred target node.

A-star is guaranteed to find a path with the overall lowest cost if h is admissible, i.e. if it returns a value that underestimates the distance of a path from a node q' to its preferred target node q'': $g(q') + h(q') \leq g(q'')$.

The A-star algorithm keeps a priority queue of nodes to be visited: higher priority is given to nodes with lower costs. At each step, the node with the lowest cost is chosen from the priority queue and is expanded with all of its successors in the graph. The search continues until a target node is reached.

We employ A-star to find the optimal alignments between an event trace $\sigma \in \mathcal{L}$ and a Petri net N. To apply A-star, an opportune search space needs to be defined. We associate every node $q \in Q$ to an alignment, which is a prefix of some complete alignment of σ and N. The source node is the empty alignment $q_0 = \langle\rangle$, and the set of target nodes includes every complete alignment of σ and N.

Given a node/alignment $\gamma \in Q$, the search-space successors of γ include all alignments $\gamma' \in Q$ obtained from γ by concatenating exactly one move. Given an alignment $\gamma \in Q$, the cost of a path from the initial node to node $\gamma \in Q$ is:

$$g(\gamma) = \|\alpha(\gamma)\| + K(\gamma)$$

where $\|\sigma\|$ denotes the length of a sequence σ and $K(\gamma)$ is the cost of alignment γ. Section IV-B discusses how the cost function is constructed.

It is easy to check that, given two complete alignments γ'_C and γ''_C, $K(\gamma'_C) < K(\gamma''_C)$ iff $g(\gamma'_C) < g(\gamma''_C)$ and $K(\gamma'_C) = K(\gamma''_C)$ iff $g(\gamma'_C) = g(\gamma''_C)$. Therefore, an optimal solution returned by A-star coincides with an optimal alignment. Various heuristic functions can be adopted. For the lack of space, we refer to [3] for an example of heuristic function.

B. Probability-based Cost Function

The cost of an alignment is equal to the sum of the costs of its constituent moves. A move on model $(\gg, m_M)$ indicates that activity m_M is expected to be executed but it is not. To account for this behavior, we compute the cost based on the probability of executing m_M as the next activity. On the other hand, a move on log $(m_L, \gg)$ indicates that event m_L was not supposed to occur and its occurrence is not conforming the process specification. Thus, we compute the cost of moves on log based on the probability of that an activity will never eventually occur in the subsequent part of the trace.

These probabilities can be computed by analyzing the past process executions recorded in an event log. Note that we only consider the traces in an event log $\mathcal{L}$ that fit the process model with respect to the control-flow, denoted $\mathcal{L}_{fit}$. This is because using non-fitting traces might lead to the construction of a cost function that is biased by behaviors that should not be permitted. Moreover, error correction for non-fitting traces is not trivial and can result in overfitting of the training data [6]. In practices, using only fitting traces as historical logging data is not an unrealistic assumption as, in many application domains, there is typically a sufficient amount of fitting traces. It is worth noting that we do not impose any constraint on the historical logging data with respect to data attributes. In fact, as we assume that information on data attributes is not provided in the process model, this information cannot be used to reason about the conformance of event traces.

Based on the observations above, we compute the conditional probabilities that activities occur or never eventually occur when being in a certain state:

Definition 10. *Let $\mathcal{L}$ be an event log and $\mathcal{L}_{fit} \subseteq \mathcal{L}$ the subset of traces that fit with a Petri net N. The probability that an activity a occurs after reaching state s with respect to $\mathcal{L}_{fit}$, denoted by $P_{\mathcal{L}_{fit}}(\bigcirc a|s)$, is the ratio between the number of event traces in $\mathcal{L}_{fit}$ in which a is executed after reaching a state subsumed by s and the total number of event traces in*

$\mathcal{L}_{fit}$ *that traverse a state subsumed by* s*:*

$$P_{\mathcal{L}_{fit}}(\bigcirc a|s) = \frac{|\{\sigma \in \mathcal{L}_{fit} : \exists \sigma' \in \mathsf{prefix}(\sigma).\ s \succ \mathsf{state}(\sigma') \wedge act(\sigma') \oplus \langle a \rangle \in \mathsf{prefix}(act(\sigma))\}|}{|\{\sigma \in \mathcal{L}_{fit} : \exists \sigma' \in \mathsf{prefix}(\sigma).\ s \succ \mathsf{state}(\sigma')\}|} \quad (4)$$

Definition 11. *Let* $\mathcal{L}$ *be an event log and* $\mathcal{L}_{fit} \subseteq \mathcal{L}$ *the subset of traces that fit with a Petri net* N*. The probability that an activity* a *will never eventually occur in a process execution after reaching state* s *with respect to* $\mathcal{L}_{fit}$*, denoted by* $P_{\mathcal{L}_{fit}}(\blacklozenge a|s)$*, is the ratio between the number of event traces in* $\mathcal{L}_{fit}$ *in which* a *is never eventually executed after reaching a state subsumed by* s *and the total number of event traces in* $\mathcal{L}_{fit}$ *that traverse a state subsumed by* s*:*

$$P_{\mathcal{L}_{fit}}(\blacklozenge a|s) = \frac{|\{\sigma \in \mathcal{L}_{fit} : \exists \sigma' \in \mathsf{prefix}(\sigma).\ s \succ \mathsf{state}(\sigma') \wedge \forall \sigma'' \in \mathcal{E}^* act(\sigma' \oplus \sigma'') \oplus \langle a \rangle \notin \mathsf{prefix}(act(\sigma))\}|}{|\{\sigma \in \mathcal{L}_{fit} : \exists \sigma' \in \mathsf{prefix}(\sigma).\ s \succ \mathsf{state}(\sigma')\}|} \quad (5)$$

To clarify the reader, we propose the following example:

Example 4. *Consider state* $s = (\langle a, b \rangle, \varphi)$ *s.t.* $\varphi(R) = \perp$, $\varphi(A) = \perp, \varphi(V) = true$, $\varphi(D) = \perp$ *and the history in Fig. 3a. As shown in Example 3, there are 1300 traces in the history whose prefix yields a state subsumed by* s*. In 400 of these traces, activity* c *is executed after reaching a state subsumed by* s*. Thus, the probability that* c *occurs after* s *is* 0.31*.*

The cost of an alignment move is defined as follow:

Definition 12 (Move Cost)**.** *Let* Σ_N *be the set of legal moves for a Petri net* N*,* $\mathcal{L}$ *an event log and* $\mathcal{L}_{fit} \subseteq \mathcal{L}$ *the subset of traces that fit with* N*. The cost of a legal move* $(m_L, m_M) \in \Sigma_N$ *with respect to a state* s *and* $\mathcal{L}_{fit}$ *is*

$$\kappa((m_L, m_M), s) = \begin{cases} 0 & act(m_L) = m_M \\ 0 & m_L = \gg \text{ and } m_M \in \mathsf{Inv}_N \\ 1 + log\left(\frac{1}{P_{\mathcal{L}_{fit}}(\bigcirc m_M | s)}\right) & m_L = \gg \text{ and } m_M \notin \mathsf{Inv}_N \\ 1 + log\left(\frac{1}{P_{\mathcal{L}_{fit}}(\blacklozenge act(m_L) | s)}\right) & m_M = \gg \end{cases} \quad (6)$$

The choice of using the formulation $1 + \log\left(\frac{1}{p}\right)$ is motivated in [3] where, through experiments, we show that this formulation provides more accurate results compared with other formulations.

The cost of an alignment is the sum of the cost of all moves in the alignment:

Definition 13 (Alignment Cost)**.** *Let* Σ_N *be the set of legal moves for a Petri net* N*. The cost of an alignment* $\gamma \in \Sigma_N^*$ *is*

$$K(\gamma) = \begin{cases} 0 & \gamma = \langle\rangle \\ \kappa((m_L, m_M), s_{\gamma'}) + K(\gamma') & \gamma = \gamma' \oplus (m_L, m_M) \end{cases} \quad (7)$$

where $s_{\gamma'}$ *is the state yielded by* γ'*:* $s_{\gamma'} = act\big(\alpha(\gamma')\big)$*.*

A *probable alignment* is an alignment with the least cost according to the cost function given in Definition 13. Below we present an example of how probable alignments are constructed.

Example 5. *Suppose that an analyst wants to construct the probable alignments of event trace* $\sigma_3 = \langle (b, \{V = true\}), (h, \{\}), (g, \{\}) \rangle$ *and the net in Fig. 1 where the traces in Fig. 3a are used as historical logging data, i.e.* $\mathcal{L}_{fit}$*. Assume that the algorithm constructed the partial alignment in Fig. 4a after analyzing the prefix trace* $\langle (b, \{V = true\}) \rangle$*. The corresponding state is presented in Fig. 4b. The alignment can be extended by move on log* $(h, \gg)$ *or by moves on model* $(\gg, c)$ *or* $(\gg, d)$*. The cost of these moves is shown in Fig. 4c. In the current state, move* $(\gg, d)$ *has the least cost. As there is no alignment with lower cost than* $\gamma \oplus (\gg, d)$*, this alignment is selected for the next iteration.*

Event Trace	Process
$\gg$	a
b, {V=true}	b

(a) Alignment γ

$$s_\gamma = (\langle a, b \rangle, \varphi) \text{ s.t. } \begin{cases} \varphi(V) = true \\ \varphi(R) = \perp \\ \varphi(A) = \perp \\ \varphi(D) = \perp \end{cases}$$

(b) State of alignment γ

$$\gamma' = \gamma \quad \oplus \quad \begin{cases} (h, \gg) & \kappa\big((h, \gg), s_\gamma\big) = 1.41 \\ (\gg, c) & \kappa\big((\gg, c), s_\gamma\big) = 1.51 \\ (\gg, d) & \kappa\big((\gg, d), s_\gamma\big) = 1.15 \\ \dots & \end{cases}$$

(c) Cost of alignment moves

Figure 4: Construction of the alignment between event trace $\sigma_3 = \langle (b, \{V = true\}), (h, \{\}), (g, \{\}) \rangle$ and the net in Fig. 1. The traces in Fig. 3a are used as $\mathcal{L}_{fit}$.

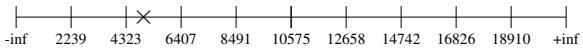
V. Experiments

We have implemented the proposed approach as a plugin of the ProM framework (http://www.promtools.org). The plug-in takes as inputs a process model and two event logs. It computes the probable alignments of each trace in the first event log and the process model based on the traces in the second event log (historical logging data).

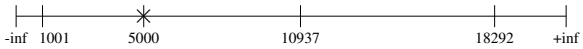
We have evaluated the proposed approach using synthetic data. The aim of the evaluation is to assess the influence of data attributes in determining the *correct* explanation of nonconformity. To this end, in the experiments we artificially added noise to the traces and assessed the ability of the approach to reconstruct the original traces.

Setting: Based on the process model in Fig. 1, we generated 20000 traces with 136138 events using the CPN tools (http://cpntools.org). We introduced different percentages of noise (10%, 20%, 30%, 40%) to 20% of the traces in the event log by adding or removing some events. The remaining traces were used as history logging data to compute the cost function. We computed the probable alignments for the manipulated traces. Then, by projecting alignments over the process model, we reconstructed the traces. To evaluate the accuracy of the approach, we considered the percentage of reconstructed traces that coincide with the original traces and the Levenshtein distance [7] between the original and reconstructed traces. The Levenshtein distance between two sequences computes the minimal number of changes required to transform one sequence into the other, indicating to what extent the approach is able to reconstruct the original traces.

In the experiments we compared our approach with two existing alignment-based techniques: one technique [3] constructs optimal alignments based on historical logging data as the approach proposed in this work but only considers the control-flow to construct optimal alignments; the other



(a) Partition of the domain into 10 equal-width bins



(b) Intervals obtained using the supervised discretization method

Figure 5: Discretization of data attribute *amount* using supervised and unsupervised methods. Symbol "×" is used to represent the interval cut used for the generation of event traces.

	Supervised Discretization		Unsupervised Discretization		[3]		[2]	
Noise	CA	LD	CA	LD	CA	LD	CA	LD
10%	99	18	97	80	89	298	86	344
20%	94	223	91	310	81	571	78	635
30%	82	735	80	827	69	1132	64	1256
40%	70	1334	67	1424	58	1778	54	1854

Table I: Results of experiments on synthetic data. CA indicates the percentage of correct alignments; LD indicates the Levenshtein distance between the original traces and the projection of the alignments over the process.

technique [2] penalizes all deviations equally and thus returns an alignment with the minimum number of deviations. In our case study, we considered continuous data attributes. This may have an impact on the state representation and, thus, bias the computation of the probabilities used for the definition of the cost function. To this end, we used various data discretization techniques [8] to partition the domain of continuous attributes into a finite number of intervals. In particular, we used the unsupervised and supervised methods provided by Weka (http://www.cs.waikato.ac.nz/ml/weka/). For unsupervised discretization, we used the equal-width method by partitioning the domain of continuous data attributes into 10 equal-width bins. Fig. 5 shows the discretization of the loan amount obtained using the equal-width and the supervised discretization methods of Weka.[4]

Results: Table I reports the average results over five runs for different levels of noise. When a supervised method was used for the discretization of data attributes, our approach, on average, computed the correct alignments for 12% and 18.25% more traces than the techniques proposed in [3] and [2] respectively. In addition, on average, the Levenshtein distance is improved 54% compared to [3] and 57% compared to [2]. These results show that the performed activities not only depend on the activities previously executed but also on data attributes. Thus, considering data attributes provides more accurate explanations of nonconformity. In particular, when the amount of noise is low, our approach is able to reconstruct the correct alignments for most of the traces.

The results also show that data discretization has a significant impact on the accuracy of the constructed explanations of nonconformity. Intuitively, if interval cuts are defined in such a way that event traces with similar behavior yield the same state, the approach can construct alignments more accurately. For the generation of the event log used for the experiments, we assumed that loan requests with an amount lower than 5000 are more likely assessed using simple assessment while the other requests are more likely assessed using advanced assessment. From Fig. 5, we can observe that the equal-width discretization method does not partition the domain of the data attribute into intervals that capture the behavior of the process properly. In fact, loan requests with an amount between 4323 and 6407 fall in the same interval. On the other hand, the supervised discretization method was able to identify the correct interval cut for the data attribute. Thus, event traces with similar behaviors are grouped more precisely, resulting in more accurate alignments compared to the case where the equal-width method was used for data discretization.

[4]The discretization is performed w.r.t. the minimal and maximal value for the data attribute observed in the training dataset. However, for the sake of generality, in Fig. 5 we replace them with $-inf$ and $+inf$ respectively.

VI. Related Work

Several approaches have been proposed in the literature for conformance checking [2], [9], [10], [11]. For instance, token-based replay techniques [9], [11] measure the conformance of an event log and a Petri net based on the number of missing and remaining tokens after replaying traces. However, these techniques fail to pinpoint the causes of nonconformity. This limitation is overcome by alignment-based techniques [2], [12], [13], [14], [15]. However, these techniques typically require analysts to define a cost function based on their background knowledge and beliefs, which can result in imperfections leading to wrong or unlikely explanations of nonconformity. Similar to this work, Alizadeh et al. [3] propose to compute the cost function automatically based on the analysis of past process executions. However, the technique in [3] only uses the control-flow perspective to compute the cost function whereas other process perspectives are not considered. As shown by our experiments, this provides less accurate explanations of nonconformity compared to the case where the data attributes manipulated by activities are also used.

This is not the first work that uses multi process perspectives for conformance checking. Some techniques [16], [17], [18] use rules to restrict a process execution using various process perspectives. These techniques, however, can only identify whether a process execution violates a particular rule. Instead of using a set of compliance rules, our approach uses an imperative process model that explicitly specifies how tasks should be executed, providing complete diagnostic information. Banescu et al. [19] propose an approach based on sequence distance metrics in which multi perspectives (e.g., data accessed by an activity, the user performing the activity) are used for the detection and quantification of privacy deviations. However, this approach is not efficient and does not support process models which allow infinitely many traces, e.g. process models with loops.

Few approaches [20], [21] extend alignment-based techniques to support conformance checking based on multi process perspectives. In [20], alignments are built by considering only the control-flow and, then, other process perspectives are used to refine the computed alignments. Mannhardt et al. [21] propose a multi perspective approach to construct optimal alignments. Both techniques represent process models as Petri nets with data and rely on guards to construct alignments. In the case guards are not provided together with the process

model, they have to be inferred. In [22], de Leoni et al. propose an approach to mine the guards of process activities. In principle, the guard-discovery approach is based on the probability of an activity to be executed in a given state: if the guard does not hold, the probability is 0; otherwise, it is 1. The main drawback of such an approach is that guards are supposed to be disjoint: in a given state, only one guard is true and, hence, only one activity is enabled. In many real-life cases, in a certain state multiple activities are possible with different probabilities; in those cases, the guard-discovery approach would fail to discover any reliable rules, making it impossible to compute a probable alignment. Rather than constructing alignments based on multi process perspectives, our approach constructs alignments on the basis of the control-flow, whereas other process perspectives are used to define the cost function. This provides the flexibility necessary to deal with situations in which well-defined guards are not provided with the process model and/or cannot be inferred.

VII. Conclusion

This paper has presented an alignment-based technique to determine probable explanations of nonconformity when a process execution deviates from the prescribed process behavior. The approach exploits information about both the control-flow and data perspective of past process executions to define a cost function that enables the construction of probable alignments. An evaluation using synthetic data shows that our approach provides more accurate explanations of nonconformity compared to alignment-based techniques that only rely on the control-flow. For future work, we plan to perform more extensive experiments to verify whether our conclusions also hold for real-life logs.

In this work, we only focused on the identification of the causes of nonconformity. To support analysts in the analysis of event traces, our approach should be complemented with methods and metrics for the analysis and possibly the quantification of the identified causes of nonconformity with respect to the application domain of interest and purpose of the analysis. Moreover, alignments only show low level deviations, i.e. moves on model and log. While these deviations indicate where the process deviates, assessing the criticality of deviations requires understanding the actual high level deviations which occurred. To this end, we are studying how low level deviations can be correlated and combined into high level deviations.

Acknowledgments: This work has been partially funded by the NWO CyberSecurity programme under the PriCE project, the Dutch national program COMMIT under the THeCS project and the European Community's Seventh Framework Program FP7 under grant agreement n. 603993 (CORE).

References

[1] W. van der Aalst, *Process Mining: Discovery, Conformance and Enhancement of Business Processes*. Springer, 2011.

[2] A. Adriansyah, B. F. van Dongen, and W. van der Aalst, "Memory-efficient alignment of observed and modeled behavior," BPMcenter.org, BPM Center Report BPM-03-03, 2013.

[3] M. Alizadeh, M. de Leoni, and N. Zannone, "History-based Construction of Log-Process Alignments for Conformance Checking: Discovering What Really Went Wrong," in *Proceedings of the 4th International Symposium on Data-driven Process Discovery and Analysis*, ser. CEUR Workshop Proceedings 1293. CEUR-ws.org, 2014, pp. 1–15.

[4] W. van der Aalst and C. Stahl, *Modeling Business Processes: A Petri Net-Oriented Approach*. The MIT Press, Cambridge, MA, 2011.

[5] R. Dechter and J. Pearl, "Generalized best-first search strategies and the optimality of A*," *Journal of the ACM*, vol. 32, pp. 505–536, 1985.

[6] S. Geman, E. Bienenstock, and R. Doursat, "Neural networks and the bias/variance dilemma," *Neural Comput.*, vol. 4, no. 1, pp. 1–58, 1992.

[7] G. Navarro, "A guided tour to approximate string matching," *ACM Comput. Surv.*, vol. 33, no. 1, pp. 31–88, 2001.

[8] S. Garcia, J. Luengo, J. A. Sáez, V. López, and F. Herrera, "A survey of discretization techniques: Taxonomy and empirical analysis in supervised learning," *TKDE*, vol. 25, no. 4, pp. 734–750, 2013.

[9] S. Banescu, M. Petkovic, and N. Zannone, "Measuring privacy compliance using fitness metrics," in *Business Process Management*, ser. LNCS. Springer, 2012, vol. 7481, pp. 114–119.

[10] J. E. Cook and A. L. Wolf, "Software Process Validation: Quantitatively Measuring the Correspondence of a Process to a Model," *TOSEM*, vol. 8, pp. 147–176, 1999.

[11] A. Rozinat and W. van der Aalst, "Conformance checking of processes based on monitoring real behavior," *Information Systems*, vol. 33, no. 1, pp. 64–95, 2008.

[12] W. M. P. van der Aalst, A. Adriansyah, and B. F. van Dongen, "Replaying history on process models for conformance checking and performance analysis," *Wiley Interdisc. Rew.: Data Mining and Knowledge Discovery*, vol. 2, no. 2, pp. 182–192, 2012.

[13] A. Adriansyah, B. F. van Dongen, and N. Zannone, "Controlling break-the-glass through alignment," in *Proceedings of International Conference on Social Computing*. IEEE, 2013, pp. 606–611.

[14] ——, "Privacy analysis of user behavior using alignments," *it - Information Technology*, vol. 55, no. 6, pp. 255–260, 2013.

[15] S. Suriadi, C. Ouyang, W. van der Aalst, and A. H. ter Hofstede, "Root cause analysis with enriched process logs," in *Business Process Management Workshops*. Springer, 2013, pp. 174–186.

[16] A. Awad, M. Weidlich, and M. Weske, "Specification, verification and explanation of violation for data aware compliance rules," in *Service-Oriented Computing*. Springer, 2009, pp. 500–515.

[17] E. Ramezani, D. Fahland, and W. van der Aalst, "Where did i misbehave? diagnostic information in compliance checking," in *Business Process Management*. Springer, 2012, pp. 262–278.

[18] E. R. Taghiabadi, V. Gromov, D. Fahland, and W. van der Aalst, "Compliance checking of data-aware and resource-aware compliance requirements," in *OTM*. Springer, 2014, pp. 237–257.

[19] S. Banescu and N. Zannone, "Measuring privacy compliance with process specifications," in *Proceedings of International Workshop on Security Measurements and Metrics*. IEEE, 2011, pp. 41–50.

[20] M. de Leoni and W. van der Aalst, "Aligning event logs and process models for multi-perspective conformance checking: An approach based on integer linear programming," in *Business Process Management*. Springer, 2013, pp. 113–129.

[21] F. Mannhardt, M. de Leoni, H. Reijers, and W. van der Aalst, "Balanced multi-perspective checking of process conformance," *Computing*, 2015.

[22] M. de Leoni and W. van der Aalst, "Data-aware process mining: discovering decisions in processes using alignments," in *Proceedings of Symposium on Applied Computing*. ACM, 2013, pp. 1454–1461.