Aligning Event Logs and Process Models for Multi-Perspective Conformance Checking: An Approach Based on Integer Linear Programming

Massimiliano de Leoni and Wil M. P. van der Aalst

Eindhoven University of Technology, Eindhoven, The Netherlands
m.d.leoni,w.m.p.v.d.aalst@tue.nl

Abstract. Modern organizations have invested in collections of descriptive and/or normative process models, but these rarely describe the actual processes adequately. Therefore, a variety of techniques for *conformance checking* have been proposed to pinpoint discrepancies between modeled and observed behavior. However, these techniques typically focus on the control-flow and *abstract from data, resources and time*. This paper describes an approach that aligns event log and model while taking *all* perspectives into account (i.e., also data, time and resources). This way it is possible to quantify conformance and analyze differences between model and reality. The approach was implemented using ProM and has been evaluated using both synthetic event logs and a real-life case study.

1 Introduction

Today's organizations are challenged to make their processes more efficient and effective; costs and response times need to be reduced in all of today's industries. Process models are used to guide people, discuss process alternatives, and to automate parts of critical business processes. Often these process models are not enforced and people can deviate from them. Such flexibility is often desirable, but still it is good to analyze differences between modeled and observed behavior. This illustrates the relevance of *conformance checking* [1]. Conformance checking techniques take an event log and a process model and compare the observed traces with the traces possible according to the model. There are different dimensions for comparing process models and event logs. In this paper, we focus of the *fitness* dimension: a model with good fitness allows for most of the behavior seen in the event log. A model has *perfect* fitness if all traces in the log can be replayed by the model from beginning to end. Other quality dimensions are *simplicity*, *precision*, and *generalization* [1].

Various conformance checking techniques have been proposed in recent years [1–4]. Unfortunately, they focus on the control-flow, i.e. the ordering of activities, thereby ignoring the other perspectives, such as data, resources, and time. In a process model, each case, i.e. a process instance, is characterized by its case variables. Paths taken during the execution may be governed by guards and conditions defined over such variables. Process models define the domain of possible values to assign to each variable, along with modeling the variables that each activity is prescribed to write or update. In addition, process models describe which resources are allowed to execute which activities. An activity is typically associated with a particular role, i.e., a selected group of resources. There may also be additional rules such as the "four-eyes principle" which

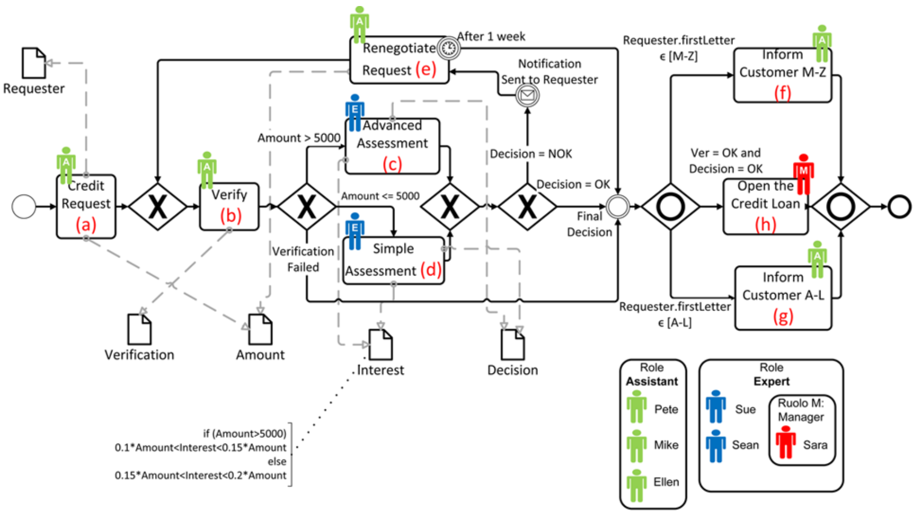


Fig. 1. BPMN diagram describing a process to handle credit requests. Besides the control-flow perspective, also the data perspective (see data objects and conditions), the resource perspective (see roles), and the time perspective (see timeout) are modeled. Dotted lines going from activities to data objects indicate the variables manipulated by each activity. Each activity requires a person having a particular role.

does not allow for the situation where the same resource executes two related tasks for the same case. Finally, there may be time-related constraints, e.g., a registration activity needs to be followed by decision activity within 30 days.

Since existing conformance checking techniques abstract from data, resources, and time, many deviations remain undetected. Let us consider the process model in Figure 1 (taken from [5]). The model describes a process to deal with loans requested by clients to buy small home appliances. After the credit request, the requester's financial data are verified and, if the verification is positive, the request is assessed. In case of a positive assessment, the credit is provided and the requester is informed. In case of a negative assessment, requesters can try to renegotiate the credit within one week or, otherwise, the request is definitely rejected. In the remainder, data objects are simply referred with the upper-case bold initials, e.g., ***V***=*Verification*, and activity names by the letter in boldface in brackets, e.g. **a**=*Credit Request*.

Let us also consider the following trace where variables are shortened with the initial letter and $\mathbf{E_x}$ and $\mathbf{T_x}$ denote the executor of x and the timestamp when x was executed:[1]

$$\begin{aligned}
&\langle(\mathbf{a}, \{\mathbf{A} = 1000, \mathbf{R} = \text{Mary}, E_a = \text{Pete}, \mathbf{T_a} = \texttt{03 Jan}\}), (\mathbf{b}, \{\mathbf{V} = \mathit{OK}, \mathbf{E_b} = \text{Sue}\}),\\
&(\mathbf{c}, \{\mathbf{I} = 150, \mathbf{D} = \mathit{OK}, \mathbf{E_c} = \text{Sue}, \mathbf{T_b} = \texttt{4 Jan}\}), (\mathbf{e}, \{\mathbf{E_e} = \text{Pete}, \mathbf{A} = 1000, \mathbf{T_e} = \texttt{15 Jan}\}),\\
&(\mathbf{c}, \{\mathbf{I} = 150, \mathbf{D} = \mathit{NOK}, \mathbf{E_c} = \text{Sue}, \mathbf{T_c} = \texttt{16 Jan}\}), (\mathbf{g}, \{\mathbf{E_g} = \text{Pete}, \mathbf{T_g} = \texttt{17 Jan}\}),\\
&(\mathbf{h}, \{\mathbf{E_h} = \text{Sara}, \mathbf{T_h} = \texttt{18 Jan}\})\rangle.
\end{aligned}$$

Conformance checking techniques only considering the control-flow perspective cannot find the following conformity's violations: *(i)* the requested amount cannot be 1000:

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

activity **d** should be executed, instead of **c**; *(ii)* for the considered credit loan, the interest is against the credit-institute's policy for large loans; *(iii)* 'Sue' is not authorized to execute activity **b** since she cannot play role *Assistant*; *(iv)* activity **e** is performed 11 days after the preceding **c** occurrence, whereas it should not be later than 7 days; *(v)* activity **h** has been executed and, hence, the last decision cannot be negative. The approach we propose is based on the principle of finding an *alignment* of event log and process model. The events in the log traces are mapped to the execution of activities in the process model. Such an alignment shows how the event log can be *replayed* on the process model. We allow costs to be assigned to every potential deviation: some deviations may be more severe than others.

This paper proposes a technique based on building a suitable ILP program to find a valid sequence of activities that is as close as possible to the observed trace, i.e., we aim to minimize the cost of deviations and create an optimal alignment. To assess the practical feasibility and relevance, the technique has also been implemented in ProM and tested using synthetic event logs and in a real-life case study. Experimental results show that conformance of the different perspectives can be checked efficiently.

When checking for conformance, pinpointing the deviations of every single trace is definitely useful, but it is not enough. Process analysts need to be provided with a helicopter view of the conformance of the model with respect to the entire log. Therefore, this paper also introduces some diagnostics to clearly highlight the most frequent deviations encountered during the process executions and the most common causes.

Our previous work [5] provides an initial approach for *multi-perspective conformance checking*. However, our previous technique could not deal with variables defined over infinite domains. The ILP-based approach presented in this paper also allows for numerical values. Our new approach is also several orders of magnitude faster. Finally, the work reported in [5] was limited to returning optimal alignments. In this paper, we provide enhanced diagnostics guiding the user in finding the root-cause of specific conformance problems.

Our conformance-checking technique is independent of the specific formalism used to describe the control-flow and data-flow perspectives. Therefore, BPMN, EPC or any other formalism can be employed to represent these perspectives. However, we need a simple modeling language with clear semantics to explain our technique. For this purpose we use *Petri nets with data*. The notation is briefly discussed in Section 2. Section 3 illustrates the basic concepts related to aligning a process and an event log. Section 4 details our new technique to compute optimal alignments and to provide enhanced diagnostics. Section 5 describes the implementation in ProM and reports on the experimental results. Finally, Section 6 concludes the paper, comparing this work with the state of the art and describing future research directions.

2 Petri Nets with Data

A *Petri net with data* (DPN-net) is a Petri net in which transitions can write variables. This formalism was introduced in [6] and, later, revisited in [7]. A transition modeling an activity performs *write operations* on a given set of variables and may have a data-dependent guard. A transition can fire only if its guard is satisfied and all input places are marked. A guard can be any formula over the process variables, using logical operators such as conjunction ($\wedge$), disjunction ($\vee$), and negation ($\neg$).

Definition 1 (DPN-net). *A Petri net with data (DPN-net) $N = (P, T, F, V, U, W, G)$ consists of:*

- *a Petri net (P, T, F);*
- *a set V of variables;*
- *a function U that defines the values admissible, i.e., for each variable $v \in V$, $U(v)$ is the domain of variable v;*
- *a write function $W : T \to 2^V$ that labels each transition with a set of* write operations*, i.e. with a s the set of variables whose value needs to be written/updated;*
- *a guard function $G : T \to \mathcal{G}_V$ that associates a guard with each transition.*[2]

When a variable $v \in V$ appears in a guard $G(t)$, it refers to the value just before the t occurrence. Nonetheless, if $v \in W(t)$, it can also appear as v' (i.e., with the prime symbol). In this case, it refers to the value after the t occurrence. Some transitions can be *invisible* and correspond to τ-steps: they do not represent actual pieces of work. Pictorially, they are represented as black boxes in the model.

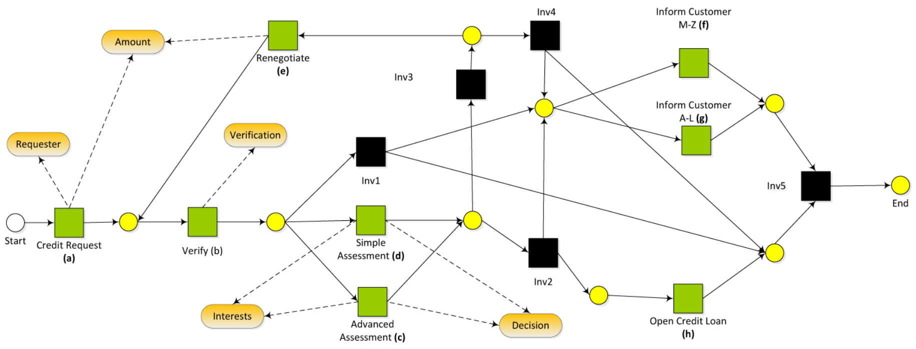
Example 1 *Figure 2 shows the DPN-net that models the same process as that modeled in Figure 1 through the BPMN notation. In particular, Figure 2(a) depicts the control-flow and the write operations. In addition to the variables depicted in the figure, there exists a set of variables to model the resource and time perspective, i.e., for each transition t, there are two variables E_t and T_t. Moreover, these two variables are associated with a write operation of t. Figure 2(b) enumerates the data-perspective guards $G_d(t)$ for each transition t. When defining guards, we assume that string values can be lexicographically ordered and, hence, it is also possible to use inequality operators (i.e., $<$ and $>$) for strings.*

To also model the resource and time perspective, a second guard $G_r(t)$ can be associated with each transition t (see Figure 2(c)). Formally, only one guard $G(t)$ can be assigned to t and, hence, we set $G(t) = G_d(t) \wedge G_r(t)$. Note the atom $E'_c \neq E_c$ in the guard of transition Simple Assessment *in Figure 2(c): it models the resource constraint that the* Simple Assessment *cannot be performed the i-th time by the same resource that performed it the $(i-1)$-th time within the same case (i.e., the "four-eyes" principle mentioned in Section 1). Formula $(T'_e \leq T_c + 7days \vee T'_e \leq T_d + 7days)$ in the guard of transition* Renegotiate Request *to model that it must occur within 7 days from the occurrence of the Assessment, Simple or Advanced. Conditions are never satisfied when involving variables that are not set.*

Space limitations prevent us from describing a complete operational semantics of DPN-net. Interested readers are referred to [7]. We only introduce the concepts needed later. The preset of a transition t is the set of its input places: ${}^\bullet t = \{p \in P \mid (p, t) \in F\}$. The postset of t is the set of its output places: $t^\bullet = \{p \in P \mid (t, p) \in F\}$. Definitions of pre- and postsets of places are analogous. A marking of a Petri net (with data) is a multiset of its places, i.e., a mapping $M : P \to \mathbb{N}$. We say the marking assigns to each place a number of tokens. A state is a pair (M, A) where M is a marking for Petri net (P, T, F) and A is a function that associates a value with each variable, i.e. $A : V \to D \cup \{\bot\}$, with $A(v) \in U(v) \cup \{\bot\}$.[3] Each DPN-net defines two special places $p_o, p_e \in P$, the initial and final place. The *initial state* is (M_0, A_0) where the initial place $p_0 \in P$ contains exactly one token, i.e. $M_0(p_0) = 1$, and any other $p \in P \setminus \{p_0\}$ contains no tokens, i.e. $M_0(p) = 0$. Moreover, $A_0(v) = \bot$ for each $v \in V$. A transition firing $s = (t, w)$ is *valid* in state (M, A) if each place in the preset of t contains at least one

[2] The guard is defined over (a sub set of) variables in V. If a transition t has no guard, we set $G(t) = \text{true}$.

[3] A special value $\bot$ is assigned to variables that have not been initialized.



(a) The control-flow and write operations. Transitions and places are represented as squares and circles, respectively. Each rounded orange rectangle identifies a different process variable. A dotted arrow from a transition t to a variable v is a pictorial representation of the fact that $v \in W(t)$.

Transition	**Guard**
Advanced Assessment	$\mathit{Verification} =$ true $\wedge\ \mathit{Amount} > 5000 \wedge 0.1 < \mathit{Interest}' / \mathit{Amount} < 0.15$
Inv1	$\mathit{Verification} =$ false
Inv2	$\mathit{Decision} =$ true
Inv3	$\mathit{Decision} =$ false
Open Credit Loan	$\mathit{Verification} =$ true $\wedge\ \mathit{Decision} =$ true
Register Decision and Inform Customer M-Z	$\mathit{Requester} \geq$ "M"
Register Decision and Inform Customer A-L	$\mathit{Requester} \leq$ "L"
Renegotiate	$\mathit{Amount}' \leq \mathit{Amount}$
Simple Assessment	$\mathit{Verification} =$ true $\wedge\ \mathit{Amount} \leq 5000 \wedge 0.15 < \mathit{Interest}' / \mathit{Amount} < 0.2$

(b) The guards to encode the data-perspective constraints.

Transition	**Guard**
Credit Request	$E'_a \in \{$"$Pete$", "$Mike$", "$Ellen$"$\}$
Verify	$E'_b \in \{$"$Pete$", "$Mike$", "$Ellen$"$\}$
Simple Assessment	$E'_c \in \{$"Sue", "$Sean$", "$Sara$"$\} \wedge E'_c \neq E_c$
Advanced Assessment	$E'_d \in \{$"Sue", "$Sean$", "$Sara$"$\} \wedge E'_d \neq E_d$
Renegotiate Request	$E'_e \in \{$"$Pete$", "$Mike$", "$Ellen$"$\}$ $\wedge(T'_e \leq T_c + 7$days $\vee\ T'_e \leq T_d + 7$days$)$
Open Credit Loan	$E'_h =$ "$Sara$"
Register Decision and Inform Customer M-Z	$E'_f \in \{$"$Pete$", "$Mike$", "$Ellen$"$\}$
Register Decision and Inform Customer A-L	$E'_g \in \{$"$Pete$", "$Mike$", "$Ellen$"$\}$

(c) The guards to encode the constraints over resources and time. $E_i \in \{a_1, \dots, a_n\}$ is a shortcut for expression $E_i = a_1 \vee \dots \vee E_i = a_n$.

Fig. 2. The DPN-net of the working example.

token, i.e. iff $\forall p \in {}^\bullet t.\ M(p) > 0$, t writes all and only the variables that it is prescribed to and $G(t)$ evaluates true with respect to assignment A. We introduce the following functions to access to the components of s: $\#_{vars}(s) = w$ and $\#_{act}(s) = t$. Function $\#_{vars}$ is also overloaded such that $\#_{vars}(s, v) = w(v)$ if $v \in \mathsf{dom}(\#_{vars}(s))$, or $\#_{vars}(s, v) = \perp$ if $v \notin \mathsf{dom}(\#_{vars}(s))$.[4]

Firing a transition s in a state (M, A) leads to a state (M', A') where M' assigns a token less than M to the t's input places ${}^\bullet t$ and a token more that M to t's output places $t^\bullet$; the number of tokens in the other places remains unchanged. Moreover, for each $v \in V$, $A'(v) = A(v)$ if $\#_{vars}(s, v) = \perp$, or, otherwise, $A(v') = \#_{vars}(s, v)$. The **set of valid process traces** of a DPN-net N is denoted with $\mathcal{P}_N$ and consists of all firing sequences $\sigma \in \big(T \times (V \not\rightarrow U)\big)^*$ that, from an initial state (M_0, A_0), lead to a state (M_F, A_F) where $M_F(p_e) > 0$.

[4] The domain of a function f is denoted with $\mathsf{dom}(f)$.

A DPN-net is *data sound* iff, for each sequence of transitions yielding a token in the final place, there is a sequence of write operations for the transitions in the sequence such that the guard of every transition t in the sequence is satisfied when t fires.

Definition 2 (Data Soundness). *Let $N = (P, T, F, V, U, W, G)$ be a DPN-net. Let $N' = (P, T, F, \varnothing, U', W', G')$ be a DPN-net such that $\mathsf{dom}(U') = \mathsf{dom}(W') = \mathsf{dom}(G') = \varnothing$. DPN-net N is data sound iff, for each $\langle s'_1, \ldots, s'_n \rangle \in \mathcal{P}_{N'}$, there exists a sequence $\langle s_1, \ldots, s_n \rangle \in \mathcal{P}_N$ where, for all $1 \leq i \leq n$, $\#_{act}(s_i) = \#_{act}(s'_i)$.*

3 Alignments of Event Logs and Process Models

An event log contains events associated to cases, i.e., process instances. Each case follows a trace of events. Each trace records the execution of a process instance. Different instances may follow the same trace. Let S_N be the set of (valid and invalid) firing of transitions of a DPN-net N with S_N. An **event log** is a multi-set of traces: $\mathcal{L} \in \mathbb{B}(S_N^*)$.[5]

Conformance checking requires an *alignment* of event log $\mathcal{L}$ and process model $\mathcal{P}$: the events in the event log need to be related to model elements and vice versa. Such an alignment shows how the event log can be replayed on the process model. This is far from being trivial since the log may deviate from the model and not all activities may have been modeled and recorded.

We need to relate "moves" in the log to "moves" in the model in order to establish an alignment between a process model and an event log. However, it may be the case that some of the moves in the log cannot be mimicked by the model and vice versa. We explicitly denote "no move" by $\gg$. For convenience, we introduce the set $S_N^{\perp} = S_N \cup \{\gg\}$.

One move in an *alignment* is represented by a pair $(s', s'') \in (S_N^{\perp} \times S_N^{\perp}) \setminus \{(\gg, \gg)\}$ such that

- (s', s'') is a *move in log* if $s' \in S$ and $s'' = \gg$,
- (s', s'') is a *move in process* if $s' = \gg$ and $s'' \in S$,
- (s', s'') is a *move in both without incorrect write operations* if $s' \in S$, $s'' \in S$ and $\forall v \in V\ \#_{vars}(s', v) = \#_{vars}(s'', v)$,
- (s', s'') is a *move in both with incorrect write operations* if $s' \in S$, $s'' \in S$ and $\exists v \in V\ \#_{vars}(s', v) \neq \#_{vars}(s'', v)$.

Event-Log Trace	Proc
a $\{\mathbf{A} = 1000, \mathbf{R} = \text{Mary}\}$	**a** $\{\mathbf{A} = 5001, \mathbf{R} = \text{Mary}\}$
b $\{\mathbf{V} = OK)$	**b** $\{\mathbf{V} = OK\}$
c $\{\mathbf{I} = 150, \mathbf{D} = OK\}$	**c** $\{\mathbf{I} = 650, \mathbf{D} = NOK\}$
	Inv3
e $\{\mathbf{A} = 1000\}$	**e** $\{\mathbf{A} = 5001\}$
	b $\{\mathbf{V} = OK\}$
c $\{\mathbf{I} = 150, \mathbf{D} = OK\}$	**c** $\{\mathbf{I} = 650, \mathbf{D} = OK\}$
	Inv2
g $\{\}$	**g**
h $\{\}$	**h**
	Inv5

Table 1. A complete alignment.

$\mathcal{A}_N = (S_N^{\perp} \times S_N^{\perp}) \setminus \{(\gg, \gg)\}$ is the set of all *legal moves*.

The **alignment** of two execution traces $\sigma', \sigma'' \in S_N^*$ is a sequence $\gamma \in \mathcal{A}_N^*$ such that, ignoring all occurrences of $\gg$, the projection on the first element yields to σ' and the projection on the second yields σ''. In the remainder, σ' and σ'' are referred to as the log and the process projection of alignment γ. In particular, γ is a **complete alignment** if $\sigma'' \in \mathcal{P}_N$. Table 1 shows a complete alignment of the process model in Figure 2 and the log trace in Section 1.

[5] $\mathbb{B}(X)$ the set of all multi-sets over X.

In order to define the severity of a deviation, we introduce a cost function on legal moves: $\kappa \in S_A \rightarrow \mathbb{R}_0^+$. The cost of each legal move depends on the specific model and process domain and, hence, cost function κ needs to be defined ad-hoc for every specific case. The cost function can be generalized to an alignment γ as the sum of the cost of each individual move: $\mathcal{K}(\gamma) = \sum_{(s',s'') \in \gamma} \kappa(s', s'')$.

However, we do not aim to find any complete alignment. Given a log trace $\sigma_L \in \mathcal{L}$, our goal is to find a complete alignment of σ_L and $\mathcal{P}$ which minimizes the cost. We refer to it as an optimal alignment. Let $\Gamma_{\sigma_L,\mathcal{P}}$ be the multi-set of all complete alignments of σ_L and $\mathcal{P}$. The alignment $\gamma \in \Gamma_{\sigma_L,\mathcal{P}}$ is an **optimal alignment** if $\forall \gamma' \in \Gamma_{\sigma_L,\mathcal{P}}\ \mathcal{K}(\gamma) \leq \mathcal{K}(\gamma')$. Note that there may exist several optimal alignments, i.e. several complete alignments having the same minimal cost.

Regarding the complexity, in [8] we discuss a polynomial-time reduction of SAT problems (i.e., simply satisfiability problems) to problems of finding an optimal alignment. The existence of this reduction implies that finding an optimal alignment is an NP-hard problem. Therefore, while it is clear that no deterministic algorithm can guarantee an optimal alignment to be computed in polynomial time, this paper attempts to reduce the average computation time.

4 The ILP-based Technique and Diagnostics

In order to find an optimal alignment of a DPN net $N = (P, T, F, V, U, W, G)$ and a log trace σ_L, we rely on existing techniques to build an alignment that only considers the control-flow perspective. Later, we construct a problem of Integer Linear Programming (ILP) to obtain an optimal alignment which also takes the other process perspectives into account.

The approach assumes four functions to be provided by process analysts. Functions $\kappa^l(t)$ and $\kappa^p(t)$ return a non-negative cost associated with a move in log or process for transition t. Function $k^v(v)$ returns a non-negative cost relative to a variable v and a move in both (s_i^L, s_i^P) where $\#_{vars}(s_i^L, v) \neq \perp$ and $\#_{vars}(s_i^L, v) \neq \#_{vars}(s_i^P, v)$ (i.e., s_i^L assigns to v a value that is out of the domain or incompatible with some guard). Function $k^n(v)$ returns a non-negative cost relative to a variable v and a move in both (s_i^L, s_i^P) where $\#_{vars}(s_i^L, v) = \perp$ and $\#_{vars}(s_i^P, v) \neq \perp$ (i.e., s_i^L does not perform a prescribed write operation for v). These four functions can be suitably composed to obtain cost functions κ as defined in Section 3. Our ILP-based technique comprises of three phases:

1. We employ the off-the-shelf techniques described in [9] to build an alignment $\gamma_C = \langle (s_1^L, s_1^P), \ldots, (s_n^L, s_n^P) \rangle$ between the Petri net (P, T, F) and the log trace σ_L using the cost functions k^l and k^m. Since such techniques only take the control-flow into account, the process projection of γ_C does not contain write operations, i.e. if $s_i^P \neq \gg$, $\mathsf{dom}(\#_{vars}(s_i^P)) = \varnothing$ for all $1 \leq i \leq n$. In the remainder, γ_C is called *control-flow alignment* and is not a complete alignment since its process projection is not a trace in $\mathcal{P}_N$.
2. We enrich the firings of transitions in the process projection σ_C of γ_C with the opportune write operations so as to minimize their difference with respect to the write operations observed in σ_L. Since it is a minimization problem, finding the opportune write operations can be formulated as solving a certain ILP problem: when a solution is found, the values of certain variables of the ILP-problem denote

those to be assigned to variables in the writing operations of σ_C. The ILP-problem objective function f is the alignment cost.

3. We compute the fitness value $\mathcal{F}(\sigma_L) \in [0,1]$. Adriansyah et al. [9] propose a fitness measurement where only the control-flow is considered. Let $\mathcal{F}_C(\sigma_L) \in [0,1]$ be this measure. Here, we propose a fitness with respect to all perspectives: $\mathcal{F}(\sigma_L) = \big(\mathcal{F}_D(\sigma_L) + \mathcal{F}_C(\sigma_L)\big)/2$ which is the mean of $\mathcal{F}_C(\sigma_L)$ and a certain quantity $\mathcal{F}_D(\sigma_L) \in [0,1]$ that considers the fitness with respect to any of the non-control-flow perspectives (data, resource, time):

$$\mathcal{F}_D(\sigma_L) = \frac{f_{\min}}{\sum_{(s^L,s^P)\in\gamma_O\,:\,s^P \neq \gg} \sum_{v \in \#_{vars}(s^P)} \max(k^d(v), k^n(v))}$$

where $f_{\min}$ is the value of the objective function for the solution found of the ILP problem. The denominator corresponds to the highest cost in term of deviations, i.e. for each move (s^L, s^P) in both, the deviations between the write operations of s^L and s^P have the highest cost.

Section 4.1 discusses how to build an ILP problem to obtain optimal solutions. To keep the discussion simple, each guard is assumed to be atomic (e.g., $Amount > 5000$). However, this limitation can be easily addressed, as discussed in [8]. The technique report also discusses how to convert date- and string-typed variables into integers and, hence, to support these types of variables. Section 4.2 discusses our proposal for a helicopter view where common deviations and their causes are shown.

4.1 Construction of the ILP problem

Given a DPN-net N and a log trace σ_L, the outcome of the first phase is a control-flow alignment $\gamma_C = \langle (s^L_1, s^P_1), \ldots, (s^L_n, s^P_n) \rangle$.

Example 2 *In order to maintain the example of a reasonable size, in the remainder, we only consider the data perspective (i.e., we ignore the guards in Figure 2(c)). Let us assume $\kappa^l(t) = \kappa^p(t) = 1$ for each transition $t \in T \setminus \{Inv1, Inv2, Inv3, Inv4\}$ and $\kappa^l(t) = \kappa^p(t) = 0$ for each transition $t \in \{Inv1, Inv2, Inv3, Inv4\}$. The latter transitions are invisible and, hence, by definition, they never appear in the log. Therefore, they are always associated with moves in process, without them being real deviations. Figure 3(a) is a possible control-flow alignment $\overline{\gamma}$ for the trace considered in Section 1, which is returned by the technique reported in [9]. This is the output of the first phase, which needs to be extended to obtain an optimal alignment.*

In order to build the ILP problem, we introduce a helper function $\#_V(\gamma, v)$ that returns the number of write operations that are prescribed to happen for variable v, considering the transitions fired in the process projection of γ.

For each write operation that is prescribed to happen for a variable $v \in V$, there exist an ILP variable v_i. The set of such variables is denoted with V_{ILP}: $V_{ILP} = \{v_i \,:\, v \in V \,\wedge\, 0 < i \leq \#_V(\gamma, v)\}$.

By analyzing the control-flow alignment, it is possible to determine whether the i-th write operation for v has or has not occurred. In remainder, constant $\underline{v_i}$ denotes the actual value observed in the log trace , with $\underline{v_i} = \perp$ if the i-th write operation has not been observed. For instance, considering the control-flow alignment in Figure 3(a) and, specifically, the first move, it can be observed that $\underline{A_1} = 1000$ and $\underline{R_1} = 1000$, whereas, considering the 6th move (i.e., for transition **b**), $\underline{V_2} = \perp$. In addition to the variables in

Event-Log Trace	Proc
a $\{\mathbf{A} = 1000, R = \text{Mary}\}$	**a**
b $\{\mathbf{V} = OK)$	**b**
c $\{\mathbf{I} = 150, \mathbf{D} = OK\}$	**c**
	Inv3
e $\{\mathbf{A} = 1000\}$	**e**
	b
c $\{\mathbf{I} = 150, \mathbf{D} = NOK\}$	**c**
	Inv2
g $\{\}$	**g**
h $\{\}$	**h**
	Inv5

(a) Control-flow Alignment.

$$\begin{aligned}
&\min 10 + \widehat{V_1} + \widehat{I_1} + \widehat{D_1} + \widehat{A_1} \\
&\quad + \widehat{A_2} + \widehat{I_2} + \widehat{D_2} + \widehat{R_1} \\
&V_1 = 1 \\
&I_1 > 0.1\ A_1 \\
&I_1 < 0.15\ A_1 \\
&D_1 = 0 \\
&A_2 < 0.6\ A_1 \\
&I_2 > 0.1\ A_2 \\
&I_2 < 0.1\ A_2 \\
&I_2 < 0.15\ A_2 \\
&D_2 = 0 \\
&R_1 < \text{“}M\text{”} \\
&\mathbf{A_1 = 1000 \Leftrightarrow \widehat{A_1} = 0} \\
&\mathbf{R_1 = \text{“Mary”} \Leftrightarrow \widehat{R_1} = 0} \\
&\mathbf{I_1 = 150 \Leftrightarrow \widehat{I_1} = 0} \\
&\mathbf{A_2 = 1000 \Leftrightarrow \widehat{A_1} = 0} \\
&\mathbf{I_2 = 150 \Leftrightarrow \widehat{I_2} = 0} \\
&\mathbf{D_2 = 1 \Leftrightarrow \widehat{D_2} = 0}
\end{aligned}$$

(b) The ILP problem to find an optimal alignment.

Event-Log Trace	Proc
a $\{\mathbf{A} = 1000, \mathbf{R} = \text{Mary}\}$	**a** $\{\mathbf{A} = A_1^*, \mathbf{R} = R_1^*\}$
b $\{\mathbf{V} = OK)$	**b** $\{\mathbf{V} = V_1^*\}$
c $\{\mathbf{I} = 150, \mathbf{D} = OK\}$	**c** $\{\mathbf{I} = I_1^*, \mathbf{D} = D_1^*\}$
	Inv3
e $\{\mathbf{A} = 1000\}$	**e** $\{\mathbf{A} = A_2^*\}$
	b $\{\mathbf{V} = V_2^*\}$
c $\{\mathbf{I} = 150, \mathbf{D} = NOK\}$	**c** $\{\mathbf{I} = I_2^*, \mathbf{D} = D_2^*\}$
	Inv2
g $\{\}$	**g**
h $\{\}$	**h**
	Inv5

(c) Optimal Alignment obtained as solution of an ILP problem: v_i^* denotes the value assigned to variable v_i in the ILP-problem solution that is found.

Fig. 3. The technique applied to the working example. First, a control-flow alignment is built, which, later, is used to build an ILP problem, whose solution allows for extending the control-flow alignment to obtain the write operations of the process projection. The constraints in bold in Figure 3(b) are non-linear. Nevertheless, as discussed in the text, each can be transformed into two equivalent linear constraints.

V_{ILP}, the ILP problem also includes a boolean variable $\widehat{v_i}$, for each $v_i \in V_{ILP}$ such that $\underline{v_i} \neq \bot$. For the solution found for the ILP problem, variable $\widehat{v_i}$ is assigned value 1 if variable v_i is not assigned value $\underline{v_i}$, i.e. there is a deviation relative to the i-th write operation for variable v. Otherwise, $\widehat{v_i}$ is given value 0.

We create a set Φ_{γ_C} of ILP-problem constraints as follow. For each prefix $\gamma'_C = \langle (s_1^L, s_1^P), \ldots, (s_i^L, s_i^P) \rangle$ of control-flow alignment γ_C, there exists a constraint $\phi \in \Phi_{\gamma_C}$ if $s_i^P \neq \gg$. Constraint ϕ is obtained starting from $G(\#_{act}(s_i^P))$ and replacing, for each $v \in V$, all occurrences of v with $v_{k-1} \in V_{ILP}$ and all occurrences of v' (i.e., with the prime symbol) with $v_k \in V_{ILP}$, where $k = \#_V(\gamma'_C, v)$.

Example 3 *By analyzing the control-flow in Figure 3(a), variables $\mathbf{V}, \mathbf{I}, \mathbf{D}$ need to be written twice each and variable $\mathbf{R}$ once. Therefore, following variables $V_1, V_2, I_1, I_2, D_1, D_2$ and R_1 are introduced to the ILP problem. Moreover, besides the write operation for $\mathbf{V}$ associated with the 6th move of the control-flow alignment (i.e., the second execution of transition **b**), they have all occurred. Therefore the following boolean variables $\widehat{V_1}, , \widehat{I_1}, \widehat{I_2}, \widehat{D_1}, \widehat{D_2}, \widehat{R_1}$ needs to be introduced, i.e. $\widehat{V_2}$ is excluded. Figure 3(b) shows the ILP problem to find the optimal alignment: the constraints not in bold are those in $\Phi_{\overline{\gamma}}$, i.e. relative to the transitions' guards. Each occurrence v and v' is replaced with the respective variable v_{i-1} and v_i of ILP problem, as described above. To enhance the example's comprehension, string constants are not converted into their respective numeric representations. Nonetheless, the conversion is necessary to be able to solve the ILP problem.*

Once these elements are introduced, the structure of the entire ILP problem can be described. Let $CN(\gamma_C, k^n)$ be the cost relative to missing write operations:

$$CN(\gamma_C, k^n) = \sum_{(s_i^L, s_i^P) \in \gamma_C \text{ s.t. } s_i^P \neq \gg} \Big(\sum_{v \in (\mathsf{dom}(\#_{vars}(s_i^P)) \setminus \mathsf{dom}(\#_{vars}(s_i^L)))} k^n(v) \Big)$$

The objective function to minimize is the cost associated with deviations of any perspective different from the control-flow:

$$\min \Big(CN(\gamma_C, k^n) + \sum_{v_i \in V_{ILP} : \underline{v_i} \neq \perp} \big(k^d(v) \cdot \widehat{v_i} \big) \Big) \tag{1}$$

subject to the constraints in Φ_{γ_C} and the following ones:

$$\forall v_i \in V_{ILP} \text{ s.t. } \underline{v_i} \neq \perp .\ v_i = \underline{v_i} \Leftrightarrow \widehat{v_i} = 0 \tag{2}$$

and

$$\forall v_i \in V_{ILP}.\ v_i \in U(v); \quad \forall v_i \in V_{ILP} \text{ s.t. } \underline{v_i} \neq \perp .\ \widehat{v_i} \in [0,1]$$

The constraints in Equation 2 are clearly not linear. Nonetheless, each of these constraints can also be written as a pair of linear constraints:

$$v_i - M\widehat{v_i} \leq \underline{v_i}; \quad -v_i - M\widehat{v_i} \leq -\underline{v_i}$$

where M is a sufficiently large number (e.g., the maximum machine-representable number). The equivalence of each pair of these constraints can be easily observed: in the solution of the ILP problem, if $\widehat{v_i} = 0$, then $v_i = \underline{v_i}$ must hold; otherwise any value can be assigned to v_i. Nonetheless, we aim to minimize the objective function in Equation 1; hence, $\widehat{v_i} = 1$ only if value $\underline{v_i}$ cannot be assigned to v_i.

Example 4 *Let us suppose that $k^n(v) = 10$ and $k^d(v) = 1$ for each $v \in V$. In Figure 3(b), the constraints in bold are those which are introduced to enforce that $\widehat{v_i} = 0$ iff $v_i = \underline{v_i}$, i.e. the constraints of the type described in Equation 2. Figure 3(c) shows the optimal alignment in function of the solution of the ILP problem. It contains write operations of form $v = v_i^*$ where v_i^* is the value assigned to variable v_i in the solution found for the ILP problem. Note the objective function has an added constant 10, which is relative to the only missing write operation, which is for V.*

The following theorem discusses the admissibility of the ILP problems constructed as described above (see [8] for a sketch of the proof):

Theorem 1 (Admissibility of the ILP problem). *Let $N = (P, T, F, V, U, W, G)$ be a data-sound DPN-Net and σ_L be a log trace. The ILP problem constructed as mentioned above to find an optimal alignment of σ_L and N is always admissible.*

4.2 A Helicopter View on the Optimal Alignments

Alignments can be projected on the process model to obtain a helicopter view. Transitions and variables are colored based on their *level of conformance*: a value between 0 and 1. Extreme values 1 and 0 identify the situations in which, according to the behavior in an event log $\mathcal{L}$, the executions of a transition or the write operations for a variable are always or never conforming with the model. When projecting deviations on the model, transitions and variables are colored according to their level of conformance: if the level is 1 or 0, a white or black color is used, with intermediate values associated with intermediate shades of color, including different intensities of yellow, orange, red, purple and brown. Readers are referred to [8] for more details on how levels of conformance are computed.

Each transition t is also associated with a decision tree that relates the deviations for t (e.g., moves in log) to the typical causes, e.g. the typical DPN-net states (M, A) when such deviations occurred. Decision trees classify instances by sorting them down in a tree from the root to some leaf node. Each non-leaf node specifies a test of some classification feature and each branch descending from that node corresponds to a range of possible values for the feature. Leaf nodes are relative to the value for the classification feature.

Decision trees are constructed starting from a set of training instances. In our case, a different training instance $\overrightarrow{o}$ is associated with each prefix $\gamma' = \langle (s_1^L, s_1^P), \ldots, (s_i^L, s_i^P) \rangle$ of every optimal alignment $\gamma \in \Gamma$. Instance $\overrightarrow{o}$ is used to train the decision tree for transition $\#_{act}(s_i^P)$, if $s_i^P \neq \gg$, or $\#_{act}(s_i^L)$, if $s_i^P = \gg$. The possible value of the classification feature in $\overrightarrow{o}$ corresponds to one of 4 types of moves, i.e. moves in process, in log, as well as moves in both with or without incorrect write operations. Let σ'_L be the log projection of alignment prefix $\langle (s_1^L, s_1^P), \ldots, (s_{i-1}^L, s_{i-1}^P) \rangle$, i.e., ignoring the last move in γ'. If γ occurs multiple times in Γ, i.e. $\#_\Gamma(\gamma) > 1$, each γ prefix generates $\#_\Gamma(\gamma)$ training instances, which are giving the same values to all classification features.

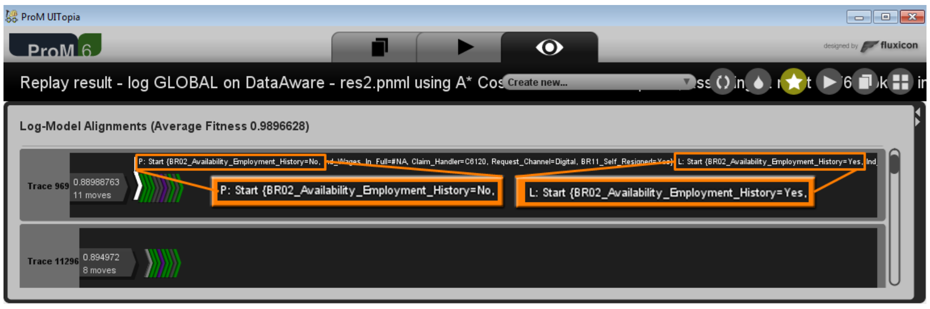
For each variable $v \in V$, there exist two classification features: v and v'. The value of the classification feature v is the value of the last write operation for v in log trace σ'_L. If there is no write operation for v in σ'_L, no value is assigned to the feature. As a matter of fact, decision-tree construction algorithms can deal with missing values of features. The value of the classification feature v' is the value assigned to v by log event s_i^L, i.e. $\#_{vars}(v, s_i^L)$. If $s_i^L = \gg$ or $\#_{vars}(v, s_i^L) = \perp$, no value is assigned to the feature. We also add an additional feature $\#t$ for each transition $t \in T$. The value for feature $\#t$ is the number of firings of transition t in σ'_L, i.e. the number of execution of t before the last move of γ'.

5 Implementation and Experiments on Real Life Event Logs

Our *multi-perspective conformance checking approach* is realized through two software plug-ins of ProM, a generic open-source framework for implementing process mining tools in a standard environment.[6] A first ProM plug-in, the *Data-aware Conformance Checker*, takes a process model in form of a DPN-net and an event log as input and operationalizes the techniques described in Section 4, including the extensions for non-atomic formulas. The output is a set of optimal alignments, one for each trace in the event log. A second plug-in, the *Data-aware Conformance Projector*, projects the optimal alignments onto the process model, operationalizing the approach described in Section 4.2. To solve ILP problems our implementation uses the *lp_solve* library, which is based on the revised simplex method combined with a branch-and-bound method for the integers.[7] To construct decision trees, we leverage on the implementation of the C4.5 algorithm in the *WEKA* toolkit.[8]

To assess the practical feasibility of the approach, the two ProM plug-ins have been tested on a real-life case study involving a Dutch insurance institute. We used an event log containing 12319 traces (i.e. process instances), where, on average, each trace is composed by around 7 events, with a minimum of 4 events and a maximum of 11

[6] http://www.promtools.org/
[7] http://lpsolve.sourceforge.net/
[8] http://weka.sourceforge.net


Replay result - log GLOBAL on DataAware - res2.pnml using A* Cos

Log-Model Alignments (Average Fitness 0.9896628)

Trace 969 0.88988763 11 moves

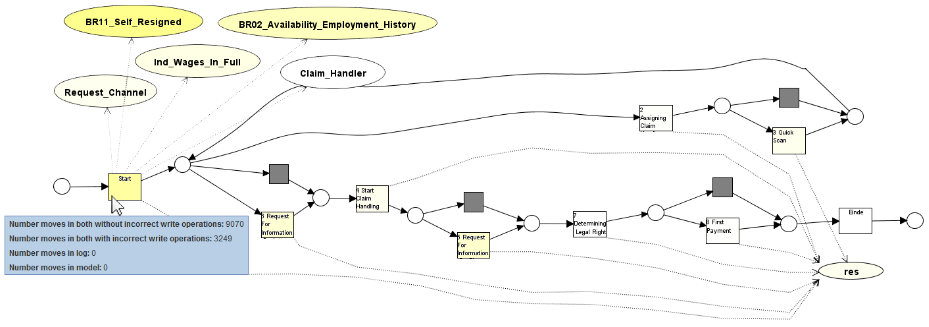
P: Start {BR02_Availability_Employment_History=No, Ind_Wages_In_Full=#NA, Claim_Handler=C6120, Request_Channel=Digital, BR11_Self_Resigned=Yes} L: Start {BR02_Availability_Employment_History=Yes, Ind

P: Start {BR02_Availability_Employment_History=No,

L: Start {BR02_Availability_Employment_History=Yes,

Trace 11296 0.894972 8 moves

Fig. 4. The ProM User Interface to show the optimal alignments.


Number moves in both without incorrect write operations: 9070

Number moves in both with incorrect write operations: 3249

Number moves in log: 0

Number moves in model: 0

Fig. 5. The process model relative to the case study of a Dutch insurance institute. The alignments have been projected on the model to pinpoint where deviations occur more often.

events. The event log has been generated through XESame, which is bundled in ProM. XESame allowed us to extract the event log from the database of the insurance institute. A control-flow process model has been designed in collaboration with a process analyst of the institute. Figure 5 shows the DPN-net for the case study. Each transition t modeling an activity t is associated with a guard $G(t) = G_r(t) \wedge G_d(t)$ where formulas $G_r(t)$ and $G_d(t)$ encode the constraints on the resource and data perspective, respectively. We have derived $G_r(t)$ for every transition t after a number of talks with the process analyst; formulas $G_d(t)$ have automatically been mined through the Decision Miner [7] and validated with the process analyst. Although the event log contains 32 data attributes, only five are actually involved in the guards of the activities. Therefore, we did not include the others in the process model to preserve the model's readability. These five attributes are written once with the *Start* activity when a insurance claim is submitted and never updated. Here, the *Data-aware Conformance Checker* plug-in is used to evaluate whether this process model is a good representation of the real behavior observed in the event log. The process analyst could not identify some deviations as more severe than others. Therefore, the four cost functions were defined so as to return 1 for any control-flow and data-flow deviation.

Visualization of the Optimal Alignments in ProM. Figure 4 shows how the optimal alignments are visualized in ProM: the optimal alignment of each log trace is shown

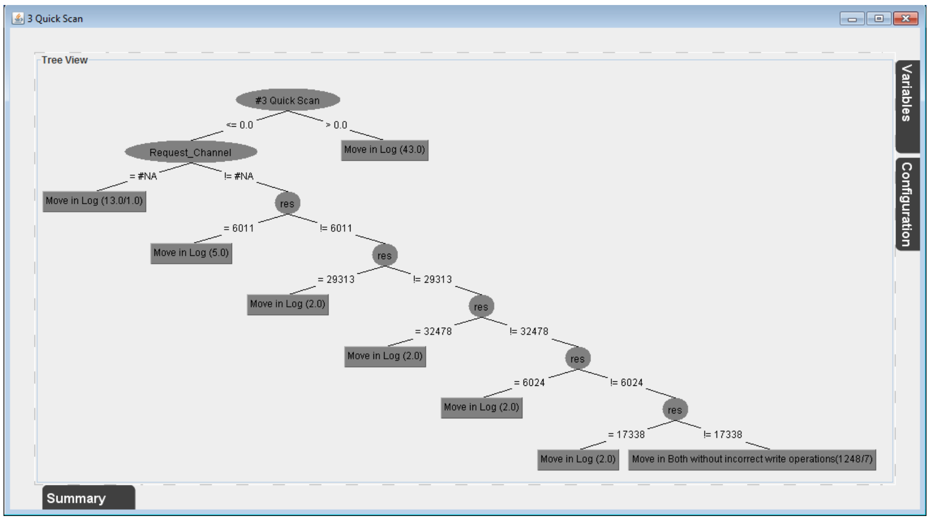


Fig. 6. The decision tree associated with activity/transition *Quick Scan*. Each tree leaf corresponds to one of the possible types of moves: in log, in process and in both, with or without incorrect write operations.

as a sequence of triangles, each representing an alignment's move. Each triangle is colored according to the move that it represents. The green and white colors are used to identify moves in both without or with incorrect write operations, respectively; yellow and purple are for moves in the log or in the process, respectively. Finally, the gray is used for moves for invisible transitions. When the user passes over a triangle with the mouse, the plug-in highlights the two transition firings s^L and s^P associated with the move (s^L, s^P). Specifically, the figure refers to a move in both with an incorrect write operation for variable *Availability_Employment_History*. The value next to the optimal alignment for every trace σ_L is the fitness value $\mathcal{F}(\sigma_L)$. On the top of the screen view, the average fitness of all traces is shown.

Enhanced Diagnostics in ProM. Figure 5 illustrates the output of the *Data-aware Conformance Projector* in ProM. Activities and variables are colored according to their level of conformance, as discussed in Section 4.2. Activity *Start* is the most involved in deviations, since the rectangle of respective transition is filled with the darkest color. Similarly, $BR11\_Self\_Resigner$ is the variables for which there is the highest number of incorrect write operations. When passing over a transition/activity or variable with the mouse, more information is given: the figure shows that activity *Start* is 3249 moves in both with incorrect write operations and 9070 moves without incorrect write operations. It is worthy observing that variable res is filled with a white color, which implies activities are generally performed by authorized resources. When clicking on a transition's rectangle, as discussed in Section 4.2, an associated decision tree is shown which classifies the types of deviations relative to the transition as function of the process state when the deviations occurred. Figure 6 shows the decision tree associated with transition *Quick Scan*. In brackets, each tree leaf shows the number of moves in the computed optimal alignments that are classified in that leaf. Some moves can be incorrectly classified: the type of moves may be different from what the values of the classification features would suggest. In these cases, after the slash, a second number is shown, which is the number of wrongly classified moves. The labels around the decision tree, i.e. *Variables*, *Configuration* and *Summary*, allow users, on the one hand, to access information about the quality of the decision tree and, on the other hand, to set the configuration parameters and the classification features to consider when constructing the decision tree. By analyzing the tree in Figure 6, one can observe that activity *Quick Scan* is usually correctly executed if it has never been previously executed (i.e., $\#3QuickScan = 0$) and the claim's request channel is known and the previous activity is performed by any resource different from #6011, #29313, #32478, #6024, #17338. This statement is not confirmed in only 7 out of 1248 moves as indicated by the label "Move in Both without incorrect write operation (1248/7)" associated with the tree's leaf. In any different state of the DPN-net when *Quick Scan* was performed, a move in log is expected.

Event Log & Model	**New**	**Previous**
Dutch Insurance Institute	3+7.02	$>$ 2 hs
Synthetic Log ($n = 4$)	0.17+0.38	13.3
Synthetic Log ($n = 5$)	0.2+0.21	48
Synthetic Log ($n = 6$)	0.2+0.44	205

Table 2. Execution time comparison between the new technique and the technique in [5] for some log-model combinations. When unspecified, the time units are seconds.

Execution-Time Analysis. As mentioned in the introduction, our new technique is several orders of magnitude faster than it predecessor described in [5]. To support this claim, we performed a number of experiments with different combinations of event logs and process models. The experimental results are summarized in Table 2: the second column is relative to the results for our new technique, i.e. the ILP-based technique described in this paper, whereas the third refers to the previous technique. For the new technique, the execution time is reported as $x + y$ where x and y are, respectively, the execution times to compute the control-flow alignments and to complement them to consider the other perspectives.

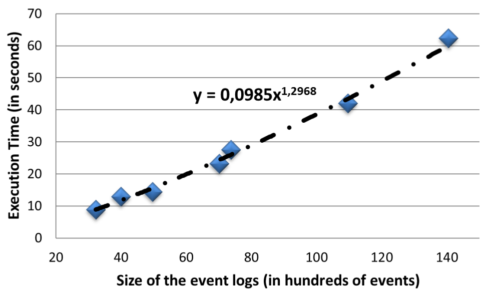


Fig. 7. Scalability of the approach with event logs of different sizes.

The first row is relative to the real-life event log of the Dutch insurance institute used before to showcase the implementation. Along with the real-life event log, we have employed the same process models that were used for execution-time analysis reported in Section 5 of paper [5]. In particular, a process model has been considered with n parallel activities. Each of n parallel activities performs a write operation for a different integer variable, which can be assigned a value between 0 and 42. After the n activities, a different activity is performed which is associated with a guard involving the n variables. Further details on the model are given in [5]. To perform the comparison, we have conducted experiments for $n = 4, 5$ or 6. For each of these three values, we have generated an event log that contained 60 traces with data-related deviations on the write operations. Finally, we have employed both of techniques and compared the execution time to find the 60 optimal alignments. Comparing the results reported in Table 2, the improvements of the new ILP-based technique are evident.

As further evaluation, we performed detailed experiments to see how the approach scales up with logs of different sizes. To this purpose, we generated 7 events logs by simulating the model in Figure 2 with CPNTools[9]. Each event log contained a different number of events while the number of traces were always 3000 (i.e., the average length of traces was different in each of 7 event logs). To generate event logs of different sizes, we instructed the simulation in a way that, on average, each loan request required a different number of renegotiations (i.e., the loop was executed a large number of times in each trace). After the generation of the logs, 20% of events were moved to a different position in the respective trace to introduce deviations. Figure 7 shows the the execution time required to compute the optimal alignments for the traces in the 7 event logs. The dotted line indicates the general trend. This shows that optimal alignments can be computed efficiently (at least, for the considered example): the execution time grows almost linearly with event logs of increasing sizes.

6 Conclusion

Various conformance checking techniques have been proposed in recent years. As mentioned in Section 1, they only focus on the control-flow thereby ignoring, e.g., incorrect routing decisions, incorrect values assigned to variables, delayed activities, and unqualified resources executing activities.

This paper presents a technique that consider data, resources and time when checking for process conformance. The proposed technique using state-of-the-art techniques to first create control-flow alignments. Such alignments are extended to incorporate the other perspectives. To extend alignments with other perspectives, an additional ILP problem is constructed and solved for each log trace. The conformance-checking technique discussed in this paper has been implemented in ProM and was tested with various synthetic log-model combinations and, also, using a real-life case study. This way we were able to demonstrate the practical relevance and feasibility of the technique.

Our approach goes much further than existing techniques for data-aware behavioral compliance checking [10, 11]. The setting considered in [10, 11] is different from ours: a set of compliance rules rather than a multi-set of log traces is checked. There also exist efficient algorithms to perform sequence alignments (e.g., the algorithms of Needleman-Wunsch and Smith-Waterman). Similarly, in Process Mining, J.C. Bose et al. [12] have proposed techniques to efficiently align pairs of log traces. Unfortunately, they cannot be applied to find an alignment between a log trace and a process model. In our setting, we do not know a priori the process trace to align with the log trace; conversely, the process trace needs to be chosen, thus minimizing the severity of the deviations. Moreover, sequence and trace alignments only focus on the activity names, i.e. the control-flow perspective, ignoring the other perspectives.

[9] http://cpntools.org/

Montali [13] developed some techniques to check the conformity of running process instances with respect to a set of temporal- and data-related constraints. Certainly, these techniques can also be applied for a-posteriori analysis, but they would not be able to pinpoint where the deviations exactly occur, which we aim to do. For this purpose, one could also leverage on existing techniques in the field of distributed systems for system debugging [14, 15]. Unfortunately, they would also be limited to alert deviations, without highlighting where they actually occur.

References

1. van der Aalst, W.M.P.: Process Mining - Discovery, Conformance and Enhancement of Business Processes. Springer (2011)
2. Rozinat, A., van der Aalst, W.M.P.: Conformance Checking of Processes Based on Monitoring Real Behavior. Information Systems **33** (March 2008) 64–95
3. Weidlich, M., Polyvyanyy, A., Desai, N., Mendling, J.: Process Compliance Measurement based on Behavioural Profiles. In: Proceedings of the 22nd International Conference on Advanced Information Systems Engineering. CAiSE'10, Springer-Verlag (2010) 499–514
4. Cook, J., Wolf, A.: Software Process Validation: Quantitatively Measuring the Correspondence of a Process to a Model. ACM Transactions on Software Engineering and Methodology (TOSEM) **8** (April 1999) 147–176
5. de Leoni, M., van der Aalst, W.M.P., van Dongen, B.F.: Data- and Resource-Aware Conformance Checking of Business Processes. In: 15th International Conference on Business Information Systems. Volume 117 of LNBIP., Springer Verlag (2012) 48–59
6. Sidorova, N., Stahl, C., Trčka, N.: Soundness Verification for Conceptual Workflow Nets With Data: Early Detection of Errors With the Most Precision Possible. Information Systems **36**(7) (2011) 1026–1043
7. de Leoni, M., van der Aalst, W.M.P.: Data-Aware Process Mining: Discovering Decisions in Processes Using Alignments. In: Proc. of the 28th ACM symposium on Applied Computing (SAC'13), ACM (2013)
8. de Leoni, M., van der Aalst, W.M.P.: Aligning Event Logs and Process Models for Multi-Perspective Conformance Checking: An Approach Based on Integer Linear Programming (2013) BPM Center Report BPM-13-05.
9. Adriansyah, A., van Dongen, B.F., van der Aalst, W.M.: Conformance Checking Using Cost-Based Fitness Analysis. In: IEEE International Enterprise Distributed Object Computing Conference, IEEE Computer Society (2011) 55–64
10. Ly, L., Rinderle-Ma, S., Knuplesch, D., Dadam, P.: Monitoring business process compliance using compliance rule graphs. In: Proceedings of OnTheMove Federated Conferences & Workshops: OTM 2011. Volume 7044 of LNCS. Springer (2011) 82–99
11. Belardinelli, F., Lomuscio, A., Patrizi, F.: Verification of GSM-Based Artifact-Centric Systems through Finite Abstraction. In: Proceedings of the 10th International Conference on Service-Oriented Computing (ICSOC'12). Volume 7636 of LNCS. Springer (2012) 17–31
12. Jagadeesh Chandra Bose, R.P., van der Aalst, W.M.P.: Process Diagnostics Using Trace Alignment: Opportunities, Issues, and Challenges. Information Systems **37**(2) (2012)
13. Montali, M.: Specification and Verification of Declarative Open Interaction Models - A Logic-Based Approach. Volume 56 of Lecture Notes in Business Information Processing. Springer (2010)
14. Reynolds, P., Killian, C., Wiener, J.L., Mogul, J.C., Shah, M.A., Vahdat, A.: Pip: detecting the unexpected in distributed systems. In: Proceedings of the 3rd conference on Networked Systems Design & Implementation - Volume 3, USENIX Association (2006) 115–128
15. Xu, W., Huang, L., Fox, A., Patterson, D., Jordan, M.I.: Detecting large-scale system problems by mining console logs. In: Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles, ACM (2009) 117–132