Data- and Resource-Aware Conformance Checking of Business Processes

Massimiliano de Leoni, Wil M.P. van der Aalst, and Boudewijn F. van Dongen

Eindhoven University of Technology, Eindhoven, The Netherlands
{m.d.leoni,w.m.p.v.d.aalst,b.f.v.dongen}@tue.nl

Abstract. Process mining is not restricted to process discovery and also includes *conformance checking*, i.e., checking whether observed behavior recorded in the event log matches modeled behavior. Many organizations have descriptive or normative models that do not adequately describe the actual processes. Therefore, a variety of techniques for conformance checking have been proposed. However, *all of these techniques focus on the control-flow and abstract from data and resources*. This paper describes an approach that aligns event log and model while taking all perspectives into account (i.e., also data and resources). This way it is possible to quantify conformance and analyze differences between model and reality. The approach has been implemented in ProM and evaluated using a variety of model-log combinations.

1 Introduction

Modern organizations are centered around the processes needed to deliver products and services in an efficient and effective manner. Organizations that operate at a higher process maturity level use formal/semiformal models (e.g., UML, EPC, BPMN and YAWL models) to document their processes. In some case these models are used to configure process-aware information systems (e.g., WFM or BPM systems). However, in most organizations process models are not used to enforce a particular way of working. Instead, process models are used for discussion, performance analysis (e.g., simulation), certification, process improvement, etc. However, reality may deviate from such models. People tend to focus on idealized process models that have little to do with reality. This illustrates the importance of *conformance checking* [1,2,3].

An important enabler for conformance checking is the availability of event data in modern organizations. Even though processes are typically not enforced by a process-aware information system, still most events are recorded. Consider for example a hospital. Medical doctors are not controlled by some BPM system. However, many events are recorded, e.g., blood tests, X-ray images, administered drugs, surgery, etc. all result in events that can be linked to a particular patient. Digital data is everywhere – in every sector, in every economy, in every organization, and in every home – and will continue to grow exponentially. MGI estimates that enterprises globally stored more than 7 exabytes of new data on disk drives in 2010, while consumers stored more than 6 exabytes of new data on devices such as PCs and notebooks [4]. The growing availability of event data is an important enabler for conformance checking.

W. Abramowicz et al. (Eds.): BIS 2012, LNBIP 117, pp. 48–59, 2012.

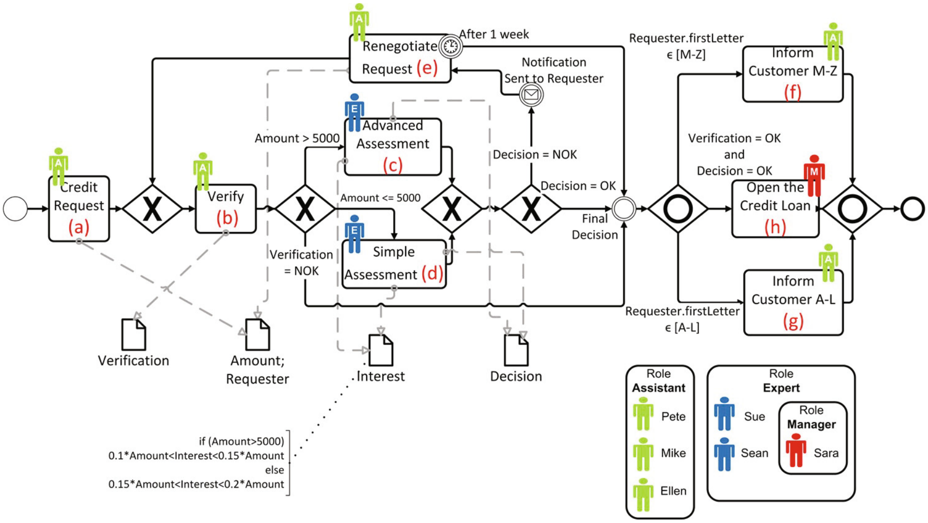


Fig. 1. BPMN diagram of a data and resource-aware process to manage credit requests to buy home appliances. In the remainder, data objects are simply referred with the upper-case initials, e.g., *V*=*Verification*, and activity names by the letter in brackets, e.g. **a**=*Credit Request*.

Along with *process discovery* (learning process models from logs) and *process enhancement* (e.g., extending process models with bottleneck information based on timestamps in event logs), conformance checking belongs to the area of *Process Mining* [5], which is a relatively young research discipline that sits between computational intelligence and data mining on the one hand, and process modeling and analysis on the other hand.

Conformance checking techniques take an event log and a process model and compare the observed behavior with the modeled behavior. 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 a *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* [1,2].

Various conformance checking techniques have been proposed in recent years [1,2,3,6,7,8,9,10,11]. However, all of the techniques described in literature focus on the control flow, i.e. the ordering of activities. They do not take into account other perspectives, such as resources and data. For example, when an activity is executed by the wrong person it is important to detect such a deviation. Conformance checking techniques need to detect that an activity reserved for gold customers is executed for silver customers. Note that information about cases and data is readily available in today's event logs. The routing of a case may depend on data, i.e., a check needs to be performed for claims over 5000. Therefore, it is important to look at the combination of all perspectives.

In a process model each case, i.e. a process instance, is characterized by its case attributes. Paths taken during the execution may be governed by guards and conditions defined over such attributes. Process models also define, for each attribute, its domain, i.e. the values that can be given. Moreover, process models prescribe which attributes every activity can read or write. Last but not least, process models also 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. Moreover, there may be additional rules such as the "four-eyes principle" which does not allow for the situation where the same resource executes two related tasks. If the data and the resource perspective are not considered, process executions can apparently be fully conforming, whereas actually they are not. Let us consider the following example:

Example 1. *A credit institute has activated a process to deal with loans requested by clients. These loans can be used to buy small home appliances (e.g., fridges, TVs, high-quality digital sound systems). A customer can apply for a loan through a shop clerk. The clerk prepares the request by filling out the form and attaching documents that prove the capability to pay off the loan. Upon receiving a new request, the credit institute opens a new case of the process in Figure 1. Dotted lines going from activities to data objects indicate the data objects (i.e., the attributes) that activities are allowed to change the value of. The resource perspective is specified by defining the role that participants need to have in order to execute the corresponding activity.*

Let us also consider the following trace where attribute E stands for *Executor* and denotes the activity executor:[1]

$$\langle (\mathbf{a}, \{A = 3000, R = \text{Michael}, E = \text{Pete}\}), (\mathbf{b}, \{V = OK, E = \text{Sue}, A = 3000, R = \text{Michael}\}),$$
$$(\mathbf{c}, \{I = 530, D = OK, E = \text{Sue}, A = 3000, R = \text{Michael}\}), (\mathbf{f}, \{E = \text{Pete}, A = 3000, R = \text{Michael}\})\rangle.$$

Existing conformance checking techniques [2,3,6,7,8,9,10,11] would only consider the control flow and ignore the decision points, values assigned to attributes and resources. Hence, the given trace would be considered as perfectly fitting. The approach proposed in this paper also considers the data and resource perspectives. For example, using our techniques, we can discover violations of rules, such as: *(i)* activity **c** should not be executed since the loan amount is not greater than 5000 (conversely, activity **d** should); *(ii)* for the considered credit loan, the interest should not be more 450 Euros and, hence, proposing an interest of 530 Euros is against the credit-institute's policy for small loans; *(iii)* 'Sue' is not authorized to execute activity **b** since she cannot play role *Assistant*; *(iv)* activity **h** has not been executed and, hence, the decision cannot be positive. The approach we propose is based on the principle of finding an *alignment* of event log and process model. The events in the 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. In [12] an alignment-based conformance checking techniques is described. However, this approach is limited to the control-flow perspective. This paper extends [12] by also taking the data and resource perspectives into account.

We allow costs to be assigned to every potential deviation. Some deviations are more severe than others and the severity can also be influenced by the point in the process when these occur, e.g., skipping a notification activity is more severe for gold customers. Our approach uses the A* algorithm [13] to find, for each trace in the event log,

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

the process execution, among those possible, whose deviations from the log trace has the lowest overall cost. In order to keep the technique as general as possible, we have developed it as independent of both the actual language in which business processes are described and the log format. Together with measuring the degree of conformance, the technique highlights where deviations occur thereby showing the control-flow, data and resource perspectives. In particular, among the different types of deviations that the technique can diagnose, it is capable to compute how much a value assignment to an attribute deviates. Similarly, from the resource viewpoint, the techniques pinpoints which resources and activities more often violate the authorization.

Section 2 illustrates a formalism that abstracts from the actual log and process notation and focuses on the behavior described by the model and recorded in the event logs. Section 3 shows how constructing an optimal alignment of process model and event log can be used to diagnose non-conformance and quantify the fitness. Section 4 elaborates the adaptation of the A* algorithm to solve the problem of conformance checking. Section 5 describes our implementation of this new approach in ProM. Moreover, experimental results are given. Finally, Section 6 concludes the paper, describing future directions of improvement.

2 The General Framework

Typically, any process model, such as the BPMN diagram in Figure 1, relies on constructs such as parallel split nodes, synchronization nodes, decision/choice nodes, conditions, merge nodes, etc. However, the model description can be "expanded" into a (possible infinite) set of (potentially arbitrarily long) traces yielding to a final state, i.e. the set of admissible behaviors. Each trace can be seen as a sequence of *execution steps*, each of which corresponds to the execution of a given process activity. Usually, a process model also defines a set of attributes together with their domain (i.e., the values that can be given). An activity is allowed to read and write attributes in a predefined manner.

Let A, V be, respectively, the finite set of activities and attributes. For all attributes $v \in V$, let us denote with $\mathsf{domAttr}(v)$ the set of values allowed for v (i.e., the attribute domain). Let be $U = \bigcup_{v \in V} \mathsf{domAttr}(v)$. A **execution step** $s = (a_s, \varphi_s)$ consists of an executed activity a_s and a function that denotes an assignment of values to process attributes: $\varphi_s \in V \not\rightarrow U$ s.t. $\forall v \in \mathsf{dom}(\varphi_s).\ \varphi_s(v) \in \mathsf{domAttr}(v)$.[2] Let S be the set of possible execution steps. A **process** $\mathcal{P}$ is the set of all admissible execution traces: $\mathcal{P} \subseteq S^*$. For each execution step $s = (a_s, \varphi_s)$, we use function $\#_{act}(s) = a_s$ to extract the activity associated to the execution step.

Resources are taken into account by "reserving" a special attribute to carry the executor information. Each value assignment to attributes can either be a read or write operation and the semantics depends on the executed activity and event log. For instance, let us consider the trace in Section 1 and the first two execution steps $s' = (\mathbf{a}, \{A = 3000, R = \text{Michael}, E = \text{Pete}\})$ and $s'' = (\mathbf{b}, \{V = OK, E = \text{Sue}, A = 3000, R = \text{Michael}\})$. The assignment $A = 3000$ for s' denotes that the execution of step s' provokes an assignment of value 3000 to attribute A. Conversely, $A = 3000$

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

for s'' indicates that, during the execution of step s'', the value 3000 has been read for attribute A. In the remainder, we focus on the writing operations. It is obvious to see that our approach can be extended to distinguish between read and write operations.

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. Therefore, an **event log** is a multi-set of traces, i.e., $\mathcal{L} \in \mathbb{B}(S^*)$.[3]

3 Aligning Event Log and Process Model

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 $\perp$. For convenience, we introduce the set $S_\perp = S \cup \{\perp\}$.

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

- (s', s'') is a *move in log* if $s' \in S$ and $s'' = \perp$,
- (s', s'') is a *move in process* if $s' = \perp$ and $s'' \in S$,
- (s', s'') is a *move in both* if $s' \in S$ and $s'' \in S$.

$S_A = (S_\perp \times S_\perp) \setminus \{(\perp, \perp)\}$ is the set of all *legal moves* where the first and the second element of every pair denote possible moves in the log and in the process, respectively.

The **alignment** of two execution traces $\sigma', \sigma'' \in S^*$ is a sequence $\gamma \in {S_A}^*$ such that, ignoring all occurrences of $\perp$, the projection on the first element yields to σ' and the project on the second yields to σ''. In particular, γ is a **complete alignment** if $\sigma' \in \mathcal{L}$ and $\sigma'' \in \mathcal{P}$.

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 costs 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 alignments γ as the sum of the cost of each individual move: $\mathcal{K}(\gamma) = \sum_{(s',s'') \in \gamma} \kappa(s', s'')$.

Example 1 (cont.). *When checking for conformance, the business analysts repute more severe the misconformances on activities that are concerned with interactions with customers, since they can undermine the reputation of the credit institute. Therefore, every alignment step between* $\perp$ *and an execution step for activities* ***c*** *or* ***d*** *is given a cost 1, whereas a cost of 10 is given to alignment steps between* $\perp$ *and execution steps for any activity different from* ***c*** *and* ***d****:*

$$\forall s \in S.\ \overline{\kappa}(s, \perp) = \overline{\kappa}(\perp, s) = \begin{cases} 1 & \textit{if} \quad \#_{act}(s) \in \{\boldsymbol{c}, \boldsymbol{d}\} \\ 10 & \textit{if} \quad \#_{act}(s) \notin \{\boldsymbol{c}, \boldsymbol{d}\} \end{cases}$$

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

γ_1	
a,{A=3000,R=Michael,E=Pete}	**a**,{A=3000,R=Michael,E=Pete}
b,{V=OK,E=Pete}	**b**,{V=OK,E=Sue}
c,{I=530,D=OK,E=Sue}	⊥
⊥	**d**,{I=599,D=NOK,E=Sue}
f,{E=Pete}	**f**,{E=Pete}

γ_2	
a,{A=3000,R=Michael,E=Pete}	**a**,{A=3000,R=Michael,E=Pete}
b,{V=OK,E=Pete}	⊥
⊥	**b**,{V=OK,E=Sue}
c,{I=530,D=OK,E=Sue}	⊥
	d,{I=599,D=NOK,E=Sue}
f,{E=Pete}	**f**,{E=Ellen}

γ_3	
a,{A=3000,R=Michael,E=Pete}	**a**,{A=3000,R=Michael,E=Pete}
b,{V=OK,E=Pete}	**b**,{V=OK,E=Sean}
c,{I=530,D=OK,E=Sue}	⊥
⊥	**d**,{I=500,D=NOK,E=Sue}
f,{E=Pete}	**f**,{E=Pete}

γ_4	
a,{A=3000,R=Michael,E=Pete}	**a**,{A=5001,R=Michael,E=Pete}
b,{V=OK,E=Pete}	**b**,{V=OK,E=Sean}
c,{I=530,D=OK,E=Sue}	**c**,{I=530,D=NOK,E=Sue}
f,{E=Pete}	**f**,{E=Pete}

Fig. 2. Four possible alignments of the log trace described in Section 1 and the process model in Figure 1

Let $\mathsf{Diff}(s', s'')$ *be the set of attributes to which both steps* s' *and* s'' *assign a value, but a different one. Every move in both is assigned a cost as follows:*

$$\forall s', s'' \in S.\ \overline{\kappa}(s', s'') = \begin{cases} 0.2 \cdot \|\mathit{Diff}(s', s'')\| & \mathit{if} \quad \#_{act}(s') = \#_{act}(s'') \wedge \#_{act}(s') \in \{\boldsymbol{c}, \boldsymbol{d}\} \\ 3 \cdot \|\mathit{Diff}(s', s'')\| & \mathit{if} \quad \#_{act}(s') = \#_{act}(s'') \wedge \#_{act}(s') \notin \{\boldsymbol{c}, \boldsymbol{d}\} \\ \infty & \mathit{otherwise} \end{cases}$$

The idea is that moves in both with different value assignment to attributes are given a higher cost for activities **c** *and* **d** *rather than for any other activity. Let us consider again the log trace given in Section 1. Figure 2 shows four possible alignments. It is easy to check that* $\mathcal{K}(\gamma_1) = \mathcal{K}(\gamma_3) = 0{+}3{+}1{+}1{+}0 = 5$, $\mathcal{K}(\gamma_2) = 0{+}10{+}10{+}1{+}1{+}2 = 24$ *and* $\mathcal{K}(\gamma_4) = 3 + 2 + 0.6 + 0 = 5.6$ *and, hence, alignments* γ_1 *and* γ_3 *are certainly better than* γ_2 *and* γ_4*.*

So far we have considered single complete alignments. However, 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 with respect to all $\sigma'_P \in \mathcal{P}$. We refer to it as an optimal alignment. Let $\Gamma_{\sigma_L,\mathcal{P}}$ be the 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 of the same minimal cost.

Example 1 (cont.). *For this example, using the cost function* $\overline{\kappa}$ *defined above,* γ_1 *and* γ_3 *are both optimal alignments. Of course, the set of optimal alignments depends on the cost function* κ*. For instance, let us consider a cost function* $\widehat{\kappa}$ *s.t.* $\forall s \in S.\ \widehat{\kappa}(\perp, s) = \widehat{\kappa}(s, \perp) = 10$ *and* $\forall s', s'' \in S. \widehat{\kappa}(s', s'') = \overline{\kappa}(s', s'')$*. Using* $\widehat{\kappa}$ *as cost function, the alignment* γ_4 *would be optimal with* $\mathcal{K}(\gamma_4) = 5.6$*, whereas alignments* γ_1 *and* γ_3 *would no more be optimal since* $\mathcal{K}(\gamma_1) = \mathcal{K}(\gamma_3) = 22$*.*

In the next section we propose an approach to create an optimal alignment with respect to a custom cost function κ. The approach is based on the A* algorithm, i.e. an algorithm intended to find the path with the lowest overall cost between two nodes in a direct graph with costs associated to nodes. We have adapted it to derive one of the optimal alignments.

4 The A* Algorithm for Conformance Checking

The A* algorithm, initially proposed in [13], aims at finding a path in a graph V from a given *source* node v_0 to any node $v \in V$ in a target set. With every node v of graph V there is an associated cost, which is determined by an *evaluation* function $f(v) = g(v) + h(v)$, where

- $g : V \to \mathbb{R}_0^+$ is a function that returns the smallest path cost from v_0 to v;
- $h : V \to \mathbb{R}_0^+$ is an heuristic function that estimates the path cost from v to its preferred target node.

Function h is said to be *admissible* if it returns a value that underestimates the distance of a path from a node v' to its preferred target node v'', i.e. $h(v') \leq g(v'')$. If h is admissible, A* finds a path that is guaranteed to have the overall lowest cost.

The A* algorithm keeps a priority queue of nodes to be visited: higher priority is given to nodes with lower costs so as to traverse those with the lowest costs at first. The algorithms works iteratively: at each step, the node v with lowest cost is taken from the priority queue. If v belongs to the target set, the algorithm ends returning node v. Otherwise, v is expanded: every successors v' is added to priority queue with a cost $f(v')$.

We employ A* to find any of the optimal alignments between a log trace $\sigma_L \in S^*$ and a Process Model $\mathcal{P}$. In order to be able to apply A*, an opportune search space needs to be defined. Every node γ of the search space V is associated to a different alignment that is a prefix of some complete alignment of σ_L and $\mathcal{P}$. Since a different alignment is also associated to every node and vice versa, later on we use the alignment to refer to the associated state. The source node is empty alignment $\gamma_0 = \langle\rangle$ and the set of target nodes includes every complete alignment of σ_L and $\mathcal{P}$.

Let us denote the length of a sequence σ with $\|\sigma\|$. Given a node/alignment $\gamma \in V$, the search-space successors of γ include all alignments $\gamma' \in V$ obtained from γ by concatenating exactly one move step. Let us consider a custom cost function κ and denote with $\kappa^{\min}$ the smallest value returned by κ that is greater than 0. Given an alignment $\gamma \in V$ of σ'_L and σ'_P, the cost of path from the initial node to node $\gamma \in V$ is:

$$g(\gamma) = \kappa^{\min} \cdot \|\sigma'_L\| + \mathcal{K}(\gamma).$$

It is easy to check that, given a log trace σ_L and two complete alignments γ'_C and γ''_C of σ_L and $\mathcal{P}$, $\mathcal{K}(\gamma'_C) < \mathcal{K}(\gamma''_C)$ iff $g(\gamma'_C) < g(\gamma''_C)$ and $\mathcal{K}(\gamma'_C) = \mathcal{K}(\gamma''_C)$ iff $g(\gamma'_C) = g(\gamma''_C)$. Therefore, an optimal solution returned by the A* algorithm coincides with an optimal alignment. Term $\kappa^{\min} \cdot \|\sigma'_L\|$, which does not affect the optimality, has been added because it allows us to define a more efficient admissible heuristics. Given an alignment $\gamma \in V$ of σ'_L and σ'_P, we employ the following heuristics:

$$h(\gamma) = \kappa^{\min} \cdot (\|\sigma_L\| - \|\sigma'_L\|)$$

For alignment γ, the number of steps to add in order to reach a complete alignment is lower bounded by the number of execution steps of trace σ_L that have not been included yet in the alignment, i.e. $\|\sigma_L\| - \|\sigma'_L\|$. Since the additional cost to traverse a single node is at least $\kappa^{\min}$, the cost to reach a target node is at least $h(\gamma)$, corresponding to the case when the part of the log trace that still needs to be included in the alignment fits in full.

5 Implementation and Experiments

The *Data-Aware Conformance Checker* is implemented as a software plug-in of ProM, a generic open-source framework for implementing process mining tools in a standard environment [14]. The plug-in takes as input a process model and a log and, by employing the techniques described in Section 4, answers to the conformance-checking questions expressed in the Section 1.

Extended Casual Nets. Our data-aware conformance-checking engine is completely independent of the process modeling language. As a proof of concept, we have used *Causal Nets* as concrete language to represent process models and extended it in order to describe the aspects related to the data and resource perspective. While Casual Nets without the data and resource perspective are thoroughly described in [1], space limitations prevent us from giving here a full formalization for their extension with these perspectives. A Casual Net extended with data is a graph where nodes represent activities and arcs represent causal dependencies. Each activity has a set of possible *input bindings* and *output bindings*. The occurrence of an activity is represented by an *activity binding* (a, ab^I, ab^O, ϕ), which denotes the occurrence of activity a with input binding ab^I and output binding ab^O and data binding function ϕ, where data attributes have global scopes. The input and output bindings include the activities that precede and succeed the occurrence of activity a. If there exists an attribute $v \in \mathsf{dom}(\phi))$ and a value u such that $\phi(v) = u$, the occurrence of a provokes to overwrite the value of attribute u with u. The definition of a process $\mathcal{P}$ in Section 2 is also applicable to Extended Casual Nets: there exists a distinct process trace for each legal sequence of activity bindings that ends with the final activity.[4] Given a valid sequence of activity bindings, the corresponding process trace contains a different execution step (a, ϕ) for each activity binding (a, ab^I, ab^O, ϕ). And the order of the execution steps in a process trace complies the order of activity bindings in the corresponding activity-bindings sequences.

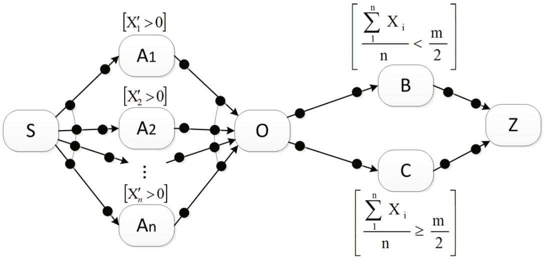
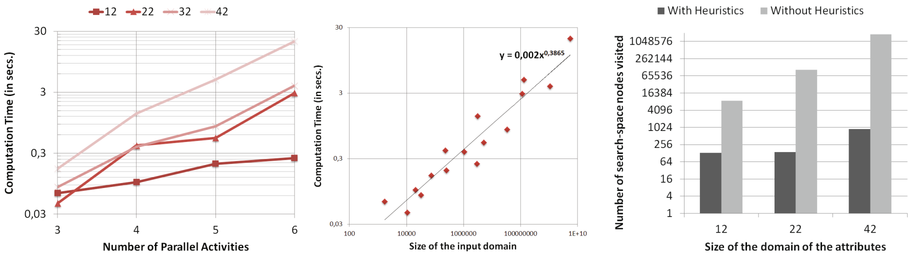

$[X'_1 > 0]$
A1
$[X'_2 > 0]$
A2
⋮
$[X'_n > 0]$
An
S
O
$\left[\frac{\sum_1^n X_i}{n} < \frac{m}{2}\right]$
B
C
$\left[\frac{\sum_1^n X_i}{n} \geq \frac{m}{2}\right]$
Z

Fig. 3. An Extended Causal Net for Example 2

Example 2. *Figure 3 shows an example of a Causal Net extended with data. There is a set of n different process attributes $X_1, \ldots, X_n$, each of which is a natural number between 1 and m. Node S is the starting activity: it has no input binding and one output binding, which is the set $\{A_1, \ldots, A_n\}$ of activities. This means activity S is followed by activities $A_1, \ldots, A_n$ executed in any order (i.e., AND-split). Activity A_i is associated a guard $X'_i \geq 0$; when an attribute, e.g. X_i, is annotated with the prime symbol in a*

[4] The lack of space does not allow us to elaborate more the concept of "legal sequence". In a few words, a sequence of activity bindings is valid if every predecessor activity and successor activity always agree on their bindings as well as the guards always hold in relation with the value assignments given. Interested readers can also refer to [1].



(a) The computation time for different numbers n of parallel activities. Each series refers to a different size m of the domain of each attribute X_i (with $m \in \{12, 22, 32, 42\}$).

(b) The influence of the size of the input domain on the computation time. In particular, the trend line shows that the computation time grows sub-linearly with growing sizes of the input domain.

(c) The number of visited search-space nodes in the case of 3 parallel activities with and without using the heuristics. The x axis refers to the size m of the domain of each attribute X_i.

Fig. 4. The results of the experiments conducted on Example 2

guard, the activity, e.g. A_i, is prescribed to update the value of the attribute. And the written value must not violate the guards, e.g. X_i has to be assigned a non-negative value. Activity O is characterized by an input binding $\{A_1, \ldots, A_n\}$, which means that O can only be executed after all activities $A_1, \ldots, A_n$ have been (i.e., AND-join). Two possible output bindings are modeled for O: B and C. Therefore, O is followed by either B or C (i.e., XOR-split). B and C are associated with two guards indicating that activities B or C can follow O if the average of values for $X_1, \ldots, X_n$ is less than $m/2$ or, vice versa, greater or equal to $m/2$.

Experiments. As input for experiments, we generated event logs by modeling the Examples 2 in CPN Tools (`http://cpntools.org`) and simulating the model. In particular, we considered all combinations in which the number n of parallel activities ranges from 3 to 6 and each attribute can be given a value between 1 and m, with $m \in \{12, 22, 32, 42\}$.

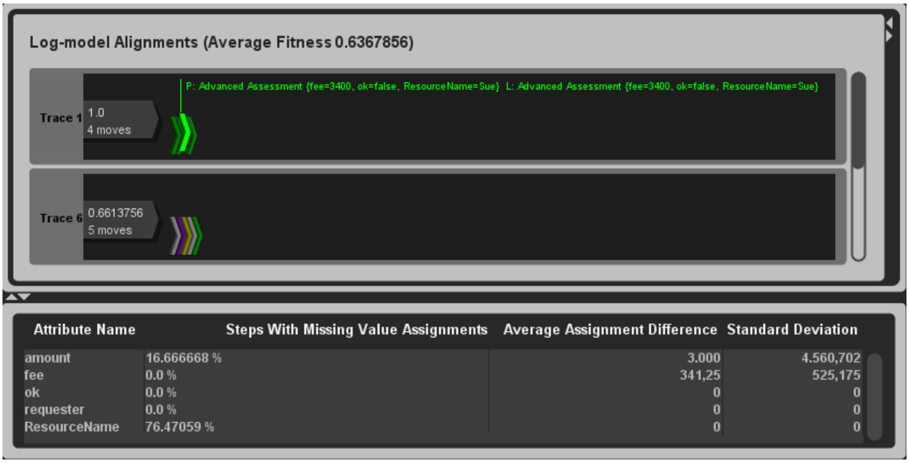
For each combinations, the log is composed by 6 different traces, generated in a random fashion but perfectly fitting. In order to make the conformance-checking analysis more challenging, from the generated logs, we have removed every occurrence of activity A_1 as well as we have swapped every occurrence of activity B and C. Moreover, we have set the cost of moving only in the process or in the log three times bigger than moving in both with different value assignments to attribute. In this way, complete alignments that contain move only in the process and in the log are always given a cost higher than move in both with different value assignments. Therefore, in order to find an optimal alignment, the conformance checker needs to find different value assignments to attributes $X_1, \ldots, X_n$ from what encountered in the log. In this way, moves only in the log or in the process can be avoided for B or C.

Figure 4 illustrates the results of the experiments. The graph in Figure 4a shows the computation time to find optimal alignments for different numbers n of parallel activities and for different sizes m of the domain of each attribute X_i. The x axis refers to number n, where the y axis is the computation time. Four series are plotted for different attribute domain sizes $m \in \{12, 22, 32, 42\}$.

For each series, the computation time grows exponentially with the number n of parallel activities. On the other hand, the size of the input domain is roughly m^n and, hence, grows exponentially with the number n of parallel activities. Indeed, each of n attributes $X_1, \ldots, X_n$ should be assigned one out of m values.

To sum up, the experiments proves that the computation time is upper bounded by a polynomial expression in the size of the input. To have a more precise estimation, we have plotted a different graph in which the x axis is the domain size and y is the computation time. The graph is shown in Figure 4b where the dotted line delineates the regression curve that better represents the trend. For this example, the actual computation time to find a solution grows as a square root of the input domain size. This sub-linear trend demonstrates that, in practical cases, the time to find an optimal alignment is only relatively affected by the number of values a certain process attribute can be given. This remarkable result is certainly related to the goodness of the employed heuristic function. In the worst case, the theoretical complexity remains exponential in the size of the domain. But, in practice, the heuristic allows the algorithm to significantly cut the number of search-space nodes to visit and, hence, the computation time to find a solution. As a matter of fact, Figure 4c shows the number of visited nodes in case of 3 parallel activities and for different values of m. In particular, we compare such a number in the case both the heuristic is used and is unused: the heuristics roughly instructs the algorithm to only visit a logarithmic number of nodes with respect to the case when the heuristic is not used.

Visualization of the Results in the Operationalization as ProM plug-in. We conclude this section by showing the actual operationalization as ProM plug-in in a scenarios in which we want to check the conformance of a given log against the process of Example 1. The log contains one perfectly-fitting trace and other trace with different problems. Figure 5 illustrates how the conformance-checking results are visualized: the optimal alignment of each log trace is shown as a sequence of triangles, each representing a move in the process and/or in the log. The triangle colors represent the alignment type. The green and white color identify moves in both with the same attribute assignment or with a different one; yellow and purple report moves only in the log or in the process, respectively. When the user passes over a triangle with the mouse, the plug-in shows the execution step(s) associated to the move. The value near to every trace is the *fitness value* of the trace, i.e. a value between 0 and 1 which quantifies the quality of the alignment. Fitness value 1 identifies the perfect alignment. Conversely, a fitness value 0 pinpoints the alignment with the largest possible cost, which typically only consists by moves in log and moves in process. Interested readers can refer to [2] where fitness values are computed in the same way. At the bottom, a table shows some statistics on the attribute assignments in the moves present in the optimal alignments shown in the upper part of the screen. The second column highlights the percentage of log steps that do not provide assignment. The last two columns report the average and the standard deviation of the difference of the values assigned to attributes in the moves. We use the hamming distance to compute the string differences and, in case of boolean attributes, we consider true as value 1 and false as value 0.



Log-model Alignments (Average Fitness 0.6367856)

Trace 1 1.0 4 moves

P: Advanced Assessment {fee=3400, ok=false, ResourceName=Sue} L: Advanced Assessment {fee=3400, ok=false, ResourceName=Sue}

Trace 6 0.6613756 5 moves

Attribute Name	Steps With Missing Value Assignments	Average Assignment Difference	Standard Deviation
amount	16.666668 %	3.000	4.560,702
fee	0.0 %	341,25	525,175
ok	0.0 %	0	0
requester	0.0 %	0	0
ResourceName	76.47059 %	0	0

Fig. 5. A screenshot of the ProM plug-in: how optimal alignments are visualized and what statistics are available on the process attributes

6 Conclusion

Process mining can be seen as the "missing link" between data mining and business process management. Although process discovery attracted the lion's share of attention, conformance checking is at least as important. It is vital to relate process models (handmade or discovered) to event logs. First of all, it may be used to audit processes to see whether reality conforms to some normative of descriptive model [15]. Deviations may point to fraud, inefficiencies, and poorly designed or outdated procedures. Second, conformance checking can be used to evaluate the performance of a process discovery technique. Finally, the alignment between model and log may be used for performance analysis, e.g., detecting bottlenecks [1].

Existing conformance checking techniques focus on the control flow thereby ignoring the other perspectives (data and resources). This paper presents a technique that takes data and resources into account when checking for process conformance. The proposed heuristics-based approach seems extremely promising since it allows for cutting out a significant part of the search space during the analysis. As a matter of fact, the computation time seems to be sub-linear, at least for the example used during the experiments.

Of course, a larger set of experiments with different processes is needed to verify our findings. Moreover, the absolute value of the computation time is still relatively high and that seems to be mostly related to the parsing of the guard expressions to determine the node successors in the search space. The parsing operations approximately take 70% of the overall computation time: we are currently investigating how to reduce the number of guards to be evaluated, along with integrating a more efficient parser.

Acknowledgements. The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 under grant agreement n^{o} 257593 (ACSI).

References

1. van der Aalst, W.M.P.: Process Mining - Discovery, Conformance and Enhancement of Business Processes. Springer (2011)
2. van der Aalst, W., Adriansyah, A., van Dongen, B.: Replaying history on process models for conformance checking and performance analysis. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 2(2), 182–192 (2012)
3. Rozinat, A., van der Aalst, W.: Conformance Checking of Processes Based on Monitoring Real Behavior. Information Systems 33, 64–95 (2008)
4. Manyika, J., Chui, M., Brown, B., Bughin, J., Dobbs, R., Roxburgh, C., Byers, A.H.: Big data: The next frontier for innovation, competition, and productivity. Technical report, McKinsey Global Institute (MGI) (May 2011)
5. van der Aalst, W., Adriansyah, A., de Medeiros, A.K.A., Arcieri, F., Baier, T., Blickle, T., Bose, J.C., van den Brand, P., Brandtjen, R., Buijs, J., Burattin, A., Carmona, J., Castellanos, M., Claes, J., Cook, J., Costantini, N., Curbera, F., Damiani, E., de Leoni, M., Delias, P., van Dongen, B.F., Dumas, M., Dustdar, S., Fahland, D., Ferreira, D.R., Gaaloul, W., van Geffen, F., Goel, S., Günther, C., Guzzo, A., Harmon, P., ter Hofstede, A., Hoogland, J., Ingvaldsen, J.E., Kato, K., Kuhn, R., Kumar, A., La Rosa, M., Maggi, F., Malerba, D., Mans, R.S., Manuel, A., McCreesh, M., Mello, P., Mendling, J., Montali, M., Motahari-Nezhad, H.R., zur Muehlen, M., Munoz-Gama, J., Pontieri, L., Ribeiro, J., Rozinat, A., Seguel Pérez, H., Seguel Pérez, R., Sepúlveda, M., Sinur, J., Soffer, P., Song, M., Sperduti, A., Stilo, G., Stoel, C., Swenson, K., Talamo, M., Tan, W., Turner, C., Vanthienen, J., Varvaressos, G., Verbeek, E., Verdonk, M., Vigo, R., Wang, J., Weber, B., Weidlich, M., Weijters, T., Wen, L., Westergaard, M., Wynn, M.: Process Mining Manifesto. In: Daniel, F., Barkaoui, K., Dustdar, S. (eds.) BPM Workshops 2011, Part I. LNBIP, vol. 99, pp. 169–194. Springer, Heidelberg (2012)
6. Weijters, A., van der Aalst, W., de Medeiros, A.A.: Process Mining with the Heuristics Miner-algorithm. Technical report, Eindhoven University of Technology, Eindhoven, BETA Working Paper Series, WP 166 (2006)
7. de Medeiros, A.A., Weijters, A., van der Aalst, W.: Genetic Process Mining: an Experimental Evaluation. Data Mining and Knowledge Discovery 14, 245–304 (2007)
8. Adriansyah, A., van Dongen, B.F., van der Aalst, W.M.P.: Towards Robust Conformance Checking. In: Muehlen, M.z., Su, J. (eds.) BPM 2010 Workshops. LNBIP, vol. 66, pp. 122–133. Springer, Heidelberg (2011)
9. Weidlich, M., Polyvyanyy, A., Desai, N., Mendling, J.: Process Compliance Measurement Based on Behavioural Profiles. In: Pernici, B. (ed.) CAiSE 2010. LNCS, vol. 6051, pp. 499–514. Springer, Heidelberg (2010)
10. Rozinat, A., Veloso, M., van der Aalst, W.: Using hidden markov models to evaluate the quality of discovered process models. Technical report, BPM Center Report BPM-08-10 (2008)
11. 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, 147–176 (1999)
12. Adriansyah, A., van Dongen, B., van der Aalst, W.: Conformance Checking Using Cost-Based Fitness Analysis. In: IEEE International Enterprise Distributed Object Computing Conference, pp. 55–64. IEEE Computer Society (2011)
13. Dechter, R., Pearl, J.: Generalized best-first search strategies and the optimality of A*. Journal of the ACM (JACM) 32, 505–536 (1985)
14. Verbeek, H.M.W., Buijs, J.C.A.M., van Dongen, B.F., van der Aalst, W.M.P.: XES, XESame, and ProM 6. In: Soffer, P., Proper, E. (eds.) CAiSE Forum 2010. LNBIP, vol. 72, pp. 60–75. Springer, Heidelberg (2011)
15. van der Aalst, W.M.P., van Hee, K., van der Werf, J.M.E.M., Verdonk, M.: Auditing 2.0: Using Process Mining to Support Tomorrow's Auditor. IEEE Computer 43(3), 90–93 (2010)