Conformance Checking Based on Partially Ordered Event Data

X. Lu, D. Fahland, W.M.P. van der Aalst

Department of Mathematics and Computer Science
Eindhoven University of Technology
P.O. Box 513, 5600 MB Eindhoven, The Netherlands
(x.lu,d.fahland,w.m.p.v.d.aalst)@tue.nl

Abstract. Conformance checking is becoming more important for the analysis of business processes. While the diagnosed results of conformance checking techniques are used in diverse context such as enabling auditing and performance analysis, the quality and reliability of the conformance checking techniques themselves have not been analyzed rigorously. As the existing conformance checking techniques heavily rely on the total ordering of events, their diagnostics are unreliable and often even misleading when the timestamps of events are coarse or incorrect. This paper presents an approach to incorporate flexibility, uncertainty, concurrency and explicit orderings between events in the input as well as in the output of conformance checking using *partially ordered traces* and *partially ordered alignments*, respectively. The paper also illustrates various ways to acquire partially ordered traces from existing logs. In addition, a quantitative-based quality metric is introduced to objectively compare the results of conformance checking. The approach is implemented in ProM plugins and has been evaluated using artificial logs.

1 Introduction

Models are increasingly used to describe business processes, to automate process executions, to communicate with stakeholders, and to evaluate designs. However, process mining research shows that process executions in reality often deviate from documented process models, potentially violating security and compliance policies. As models enable various analysis techniques ranging from verification to simulation, it is essential to provide *diagnostic information about the conformance of process models with respects to event logs recording the real behavior* [1]. This information can be further used to identify and measure deviations, enable auditing and compliance analysis [2, 3]. Moreover, the relationships between models and logs obtained from aligning them can be used to analyze performance and repair process models [4, 5].

Dozens of approaches [1, 4, 6, 7, 2] have been proposed to check conformance between a given model and a sequence of events (a so-called *trace*) from an event log, but all of these approaches assume that the events in a trace are totally ordered (e.g. based on precise timestamps). The state-of-the-art technique in conformance checking is the *alignment* approach proposed in [6, 7], which relates behavior observed in a sequential

trace (i.e. events) to behavior documented in a model (i.e. activities) and identifies deviations between them. Alignments with the least number of deviations are considered to be optimal.

One of the limitations of computing optimal alignments for sequential traces is that current approaches heavily rely on the total ordering of events in a trace. In cases where the timestamps are too coarse (e.g. only the dates are recorded and the order of the events on the same day is unknown) or incorrect (e.g. due to manual recording), using a non-trustworthy total ordering of these events for computing alignment may result in classifying abnormal behavior as conforming and normal behavior as deviating. Moreover, sequential traces are unable to describe concurrent events (e.g. events happening at the same time or events of which we known that there is no causal dependency between then). Furthermore, the dependencies between events in the resulting alignments may be misleading because of ordering problems.

To overcome these limitations, we propose to use partially ordered events rather than totally ordered events. This way we can express causal dependencies, uncertainty, and concurrency in a better way. Moreover, we also argue that computing partially ordered alignments based on partial orders provides more precise diagnostic results for conformance checking.

In this paper, we introduce *partially ordered traces* and use these as input to compute *partially ordered alignments* with respect to a given model. An overview of our approach is shown in Figure 1. We discuss our approach in two parts:

1) Given a partially ordered trace and a process model, we show how to compute a partially ordered alignment.
2) Given a log, we show how to obtain partially ordered traces which are used as input for the first part.

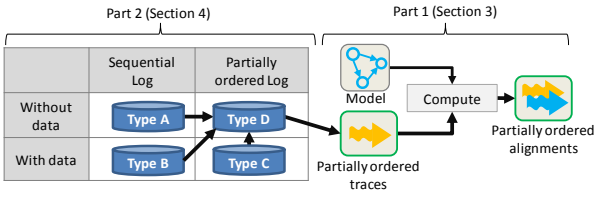

Fig. 1: An overview of our approach.

In the first part, we discuss a generic approach which extends the sequential alignment approach [7] to computing optimal partially ordered alignments. In addition, we also introduce a quantitative-based *alignment quality metric* to measure and to compare the quality of alignments and to evaluate our approach.

In the second part, we discuss ways to derive the input of the first part, i.e. partially ordered traces, from a given log. More specifically, we categorize input logs into four types: sequential logs without data (Type A), sequential logs with data (Type B), partially ordered logs with data (Type C) and partially ordered logs without data (Type D). For each type, we discuss an example of computing partially ordered traces. In addition, we will demonstrate shortcomings of totally ordered alignments that can be overcome by partially ordered alignments.

The remainder of this paper is structured as follows. We first introduce some basic concepts in Section 2. Section 3 defines partially ordered traces and alignments, describes our approach based on computing partially ordered alignments, and intro-

duces a novel alignment quality metric. Section 4 discusses ways to acquire partially ordered traces from classical sequential traces and shows examples where partially ordered alignments perform better than existing conformance checking approaches. Section 5 presents results of experiments we conducted. Section 6 discusses the related work, and Section 7 concludes the paper.

2 Preliminaries

In this section, we first introduce a running example and use the running example to recall some preliminaries related to event logs and alignments.

Running example. Figure 2 shows a simplified process in a hospital. The process starts with a patient having an appointment (A). Next, a doctor can check the patient history (C) while the patient is scheduled for radiology (R) and followed by a lab test (L). The doctor evaluates (E) the result of these tests and determines whether to operate (O) or to send patient home for home treatment (H). Operated patients require nursing (N). Finally, the patient might be re-evaluated (V) to determine whether she has to be operated on again.

Case, Events, Traces, Logs. A *case* is a process instance, i.e. an execution of a process. An *activity* is a well-defined task in a process model (e.g. blue rectangles in Figure 2). Executing an activity for a case results in an *event* recorded in the trace of this case. Each event includes a set of data attributes describing the event. In the classical setting of process mining, a *trace* is thus a totally ordered sequence of events of a case. We use *s-trace* to denote such a *sequential trace*. A *log* is a collection of traces that belong to the same process model. In the running example, a *case* is a patient going through this process. Figure 3(a) shows a s-trace consisting of seven events, each of which has four data attributes: the case id, the event id, the activity name and the timestamp.

Alignments. A *sequential alignment* (abbreviated to *s-alignments*) between a trace and a process model is defined as a sequence of *moves*, each of which relates an event in the trace to an activity in the model. A "good" move is a so-called *synchronous move*, which is an event observed in the trace and allowed according to the documented behavior (i.e. an activity to which the event can be related). Deviations are indicated by so-called *log moves* and *model moves*: a *log move* is an observed event not allowed by the modeled

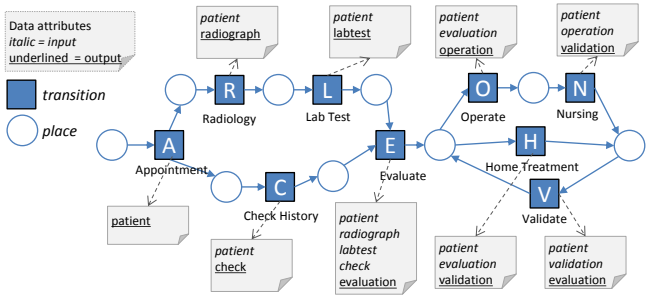

Fig. 2: An example of a simplified process in a hospital.

Case	Evt	Act	Time
120	1	A	14-4-21
120	2	L	14-4-21
120	3	D	14-4-21
120	4	R	14-4-21
120	5	C	14-4-21
120	6	E	14-4-21
120	7	H	14-4-21

(a) s-trace

Act	A	L	D	R	C	≫	E	H
M	A	≫	≫	R	C	L	E	H

(b) s-alignment

Fig. 3: An s-trace and its s-alignment.

behavior; a *model move* is an event that should have been observed according to the modeled behavior but missed in the trace. A *cost function* assigns to each possible move a cost. An s-alignment with lowest cost according to the cost function is an *optimal s-alignment*. For the technical details, we refer to [7]. In the rest of this paper, we use the *standard cost function* which assigns to each log move and model move the same cost of 1 and to each synchronous move a cost of 0. For example, the optimal s-alignment shown in Figure 3(b) between the s-trace in Figure 3(a) and the model in Figure 2 consists of four synchronous moves, two log moves $\left|\frac{L}{\gg}\right|$ and $\left|\frac{D}{\gg}\right|$ and one model move $\left|\frac{\gg}{L}\right|$.

Partial orders. A *partial order* over a set V is a binary relation which is irreflexive, antisymmetric, and transitive. A *directed acyclic graph (DAG)* $G = (V, \rightarrow)$ defines a partial order $\prec$ over V, i.e. for all $v_1, v_2 \in V$, if there is a *path* from v_1 to v_2, then $v_1 \prec v_2$. The transitive reduction of a DAG and its corresponding partial order is unique [8].

3 Partially Ordered Alignment

In this section, we define the notion of partially ordered traces and alignments. Moreover, we describe our approach to compute an optimal partially ordered alignment if a partially ordered trace is given. In addition, we also define an alignment quality metric to compare two partially ordered alignments.

Definitions. Given a case with its set of events, a *partially ordered trace* (*p-trace*) is a directed acyclic graph (which defines a partial order) over the set of events. Each *dependency (dep.)* in p-trace from an event e_i to another event e_j indicates that event e_i has led to the execution of e_j. For example, the p-trace shown in Figure 4 has the same events as the case of Figure 3(a). The partial order in Figure 4 shows that events *R*, *L*, *H* and *D* directly depend on event *A*, while *R*, *L*, *H* and *D* are *concurrent* to each other (i.e. in no particular order).

Correspondingly, a *partially ordered alignment* (*p-alignment*) between a p-trace and a process model is a directed acyclic graph (which defines a partial order) over the set of moves between them. A *move* comprises an event in the p-trace and an activity in the model to which the event is related, similar to moves in s-alignments. There are three types of moves: *synchronous moves*, *log moves* and *model moves*. The ordering of moves (i.e. dependencies between the moves) in a p-alignment respects the ordering of their events in the p-trace or the ordering of their activities in the model. For instance, Figure 5 exemplifies a p-alignment between the p-trace shown in Figure 4 and the model shown in Figure 2. The p-alignment shown in Figure 5 has five synchronous moves (denoted by green five-sided polygons, e.g. *A*), two log moves (denoted by yellow triangles, e.g. *D*) and one model move (denoted by blue rectangle, e.g. *E*).

Dependencies between moves originate either from dependencies between log events (yellow), between moves (blue), or from both (green), see Figure 5. A dependency between two moves is a *direct dependency (d-dependency or d-dep.)* if and only if there is no other path between the two moves. The *minimal p-alignment* of a p-alignment is the transitive reduction of the p-alignment and only consists of d-dependencies.

An *optimal p-alignment* is a p-alignment with lowest cost according to the cost function. We also introduce the notion of the *ideal p-alignment* γ_* of a case, which is

the only true p-alignment of the case, i.e. both the diagnosed moves and dependencies of γ_* are correct. The p-alignment shown in Figure 5 is for example assumed to be the ideal p-alignment for the case shown in Figure 3(a).

Approach. Our method extends the approach of [7] to compute an optimal p-alignment between a given p-trace and a model. We first convert the p-trace into a so-called *event net*, which is a Petri net that represents the behavior of the given p-trace. More precisely, each event in the p-trace is represented by a transition in the event net, and each dependency found between two events is converted into a place between their corresponding transitions in the event net. To complete the event net, for each event that has no predecessor or no successor, we add an input place or an output place, respectively.

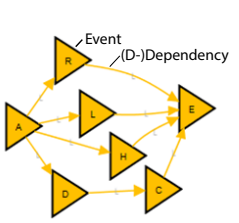

Fig. 4: A p-trace for the case in Figure 3.

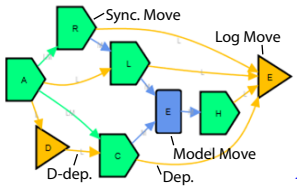

Fig. 5: An optimal p-alignment for the p-trace in Figure 4.

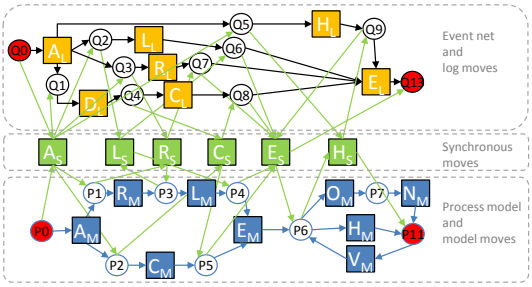

Fig. 6: A product net between the p-trace in Figure 4 and the process model in Figure 2.

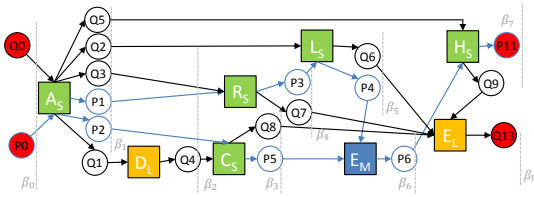

Fig. 7: The partially ordered alignment net of the p-alignment in Figure 5.

After computing the event net, we join the event net with the process model to obtain a *product net* which consist three types of transition representing the three types of move (i.e. log moves, model moves, and synchronous moves). For further detail, we refer to [7]. Figure 6 exemplifies the product net between the p-trace shown in Figure 4 and the process shown in Figure 2.

Next, we compute a firing sequence with a lowest cost (according to the standard cost function) from the initial marking to the final marking using the A^*-approach proposed in [7]. Then, we replay the firing sequence on the product net. We only retain the places visited, the transitions fired and the arcs between them. We call the retained net an *optimal alignment net*. For example, Figure 7 shows an optimal alignment net of the product net in Figure 6. Finally, we convert an optimal alignment net into an optimal p-alignment by replacing the places between transitions with dependencies. Figure 5 shows the optimal p-alignment converted from the optimal alignment net shown in Figure 7.

Table 1: Compare s-alignments and p-alignments using quality metrics.

	Sync. Move			Log Move			Model Move			d-dependencies		
	TP	FP	FN	TP	FP	FN	TP	FP	FN	TP	FP	FN
Ideal	5	0	0	2	0	0	1	0	0	8	0	0
Seq.	4	1	1	1	1	1	0	1	1	2	5	6
Type A	5	1	0	1	0	1	0	0	1	5	1	3
Type B	5	1	0	1	0	1	0	0	1	7	0	1

Alignment Quality Metrics. To compare two p-alignments, we define the true positives, false positives, and false negatives for synchronous moves, log moves, model moves, and d-dependencies. Assuming the ideal p-alignment γ_* is known for a given case, we can compute a p-trace and compare an (optimal) p-alignment γ' of the p-trace to the ideal p-alignment as follows.

- For each synchronous move $m_s \in \gamma'$, if m_s is also found in the ideal p-alignment γ_*, the synchronous move m_s is *True Positive (TP)*; if m_s not found in γ_*, m_s is *False Positive (FP)*. The same for each log move, model move and d-dependency found in γ'.
- For each synchronous move m_{s*} found in the ideal p-alignment γ_* but not in γ', m_{s*} is considered to be a *False Negative (FN)*. The same for each log move, model move and d-dependency found in γ_*.

As additional quality metric, we can compute the F1-score $F_1 = (2 \times TP)/(2 \times TP + FP + FN)$ for the moves and dependencies identified, which is the harmonic mean of recall and precision [9].

Note that by definition, an s-alignment is also a p-alignment and thus can be compared to an ideal p-alignment. For example, we can convert the s-alignment in Figure 3(b) into the p-alignment shown in Figure 8. Assuming Figure 5 shows the ideal p-alignment of the same case in Figure 3(a), the first row and the second row in Table 1 respectively show the quality metrices for the ideal p-alignment and the s-alignment (compared to the ideal p-alignment). For instance, in the s-alignment we found one FP synchronous move *E* (i.e. the five-sided polygon *E*), which is a log move in the ideal p-alignment (i.e. a FN log move) in Figure 5. This example shows that the s-alignment approach may classify abnormal behavior as conforming. Furthermore, the FP log move *L* found in the s-alignment is classified as a synchronous move in the ideal alignment: the s-alignment approach may claim conforming behavior as deviating.

4 Conversion and Comparison

In Section 3, we explained our approach for computing an optimal p-alignment when given a p-trace. In this section, we discuss ways to compute p-traces. Recall the four

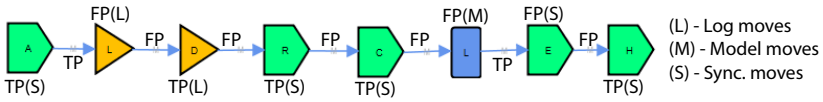

Fig. 8: A p-alignment which visualizes the s-alignment in Figure 3(b).

types of log defined in the introduction, if a log (with or without data) is already partially ordered, i.e. Type C and D, we can simply consider its p-traces and neglect the data attributes. For sequential logs with or without data attributes (i.e., type B and A, respectively), we illustrate for each type an example to compute partially ordered traces. In addition, we motivate p-alignments by using these examples and compare the results based on our alignment quality metrics.

4.1 Type A - Sequential Logs without Data

Type A denotes sequential event logs without data. Each log of this type is a collection of s-traces in which each event has only the basic attributes: the event identifier, the activity name and the timestamp. For this type of log, there are various situations in which we can compute p-traces and use the p-traces to obtain p-alignments. One of the possible situations is when the timestamps of events are coarse, and the ordering of events are unreliable. For instance, for each event only the date is recorded which may lead to multiple events having the same timestamp, exemplified by the trace in Figure 3 in which all seven events occurred on the same day.

A simple approach to compute p-traces in this situation is to consider the events having the same timestamp to be concurrent. This approach adds flexibility when computing alignments and removes false positive log moves and model moves. Figure 9 shows the p-trace computed for the s-trace shown in Figure 3 using this approach. Since all seven events have the same timestamp, they are considered to be concurrent, i.e. no dependency between them as shown in Figure 9. Therefore, the events could have happened at any order, resulting in the p-alignment shown by Figure 10.

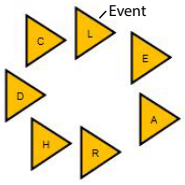

Fig. 9: A p-trace for Figure 3(a) derived based on timestamps.

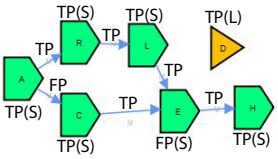

Fig. 10: An optimal p-alignment for the p-trace in Figure 9.

Computing the quality metric of the p-alignment shown in Figure 10 with respect to the ideal alignment shown in Figure 5, we obtain the result shown by the third entry in Table 1. Compared to the s-alignment shown in Figure 8 of the same case, there are no FP log moves or FP model moves in the p-alignment, and only one FP d-dependency. Moreover, we find 2.5 times more TP d-dependencies.

4.2 Type B - Sequential Logs Annotated with Data

In this section, we first define data annotated logs and then discuss how to use this type of log to compute p-traces.

Definition. We use the term *Data Annotated Log* (DAL) to denote a specific type of event log, in which each event has a set of clearly annotated input attributes and of output attributes, i.e. in addition to the name and the value of an attribute, we also have a meta data for each attribute which indicates whether the attribute is an input or an

output of the event. *Input attributes* of an event are attributes that already existed and are *read* when executing the activity that results in the event. Similarly, *output attributes* of an event are attributes that are *written* (created or updated) by the event. In addition, we assume that if the value of a data attribute d_1 depends on the value of another data attribute d_2, there exists an event that reads d_1 and writes d_2.

Figure 11 shows an s-trace of the same case as in Figure 3 but with data. Each event has additional data attributes that are annotated as inputs (written in italics) or outputs (underlined). The column names denote the abbreviated identifier of attributes defined in the process model shown in Figure 2.

Obtaining DAL. One may argue that this type of log is difficult to obtain. However, there are simple heuristics to convert a log enriched with data attributes but without annotations to a DAL. Given a log in which each event has a set of data attributes, if a specification of the input and output attributes of each activity is available (e.g. given by a domain expert or documented as shown in Figure 2), we can use this specification to annotate the data attributes of events in a log. Otherwise, we can determine for each event in a trace and each of its data attributes whether it is an input or an output using the following heuristics:

1. When a data attribute appears the first time in an event in the trace, the event has output the data (e.g. attribute p of event 1 and attribute e of event 6 in Figure 11);
2. Every time the data attribute with the same data attribute name appears in a succeeding event, if the value of this attribute has changed compared to the previous appearance, then the event has output the data (e.g. attribute e of event 7 in Figure 11 has a different value compared to the previous event E that has attribute e, therefore, attribute e of event 7 is annotated as output (i.e. underlined));
3. Otherwise, the data is an input of the event (e.g. attribute p in events $2-6$ is considered as input because the value of p is not changed).

Thus, Figure 11 also exemplifies a trace annotated using this simple heuristic. We have illustrated a simple heuristic approach to show that it is possible to obtain DALs without any specification. Finding better heuristics is a relevant topics, but out of scope of this paper. In the following, we assume that DALs are available for computing p-traces.

Computing Partially Ordered Traces. After obtaining a data annotated sequential log, the data dependencies between the input and output attributes of events can be used to derive dependencies and concurrency between events. When two events e_i and e_j in a trace with $i < j$ accessed a common data attribute, we assume that there is a dependency between the two events. Based on this assumption, we derive two rules: (1) when two

				Annotated Data					
Case	Evt	Act	Time	p	r	l	d	c	e
120	1	A	14-4-21	Lu					
120	2	L	14-4-21	*Lu*		1			
120	3	D	14-4-21	*Lu*			Z		
120	4	R	14-4-21	*Lu*	5				
120	5	C	14-4-21	*Lu*			*Z*	5	
120	6	E	14-4-21	*Lu*	*5*	*1*		*5*	true
120	7	H	14-4-21	*Lu*					false

Fig. 11: An s-trace with annotated attributes.

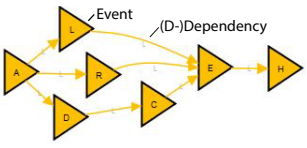

Fig. 12: A p-trace for Figure 11 derived based on data dependencies.

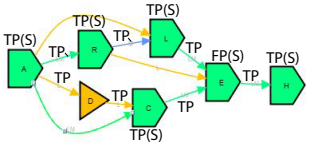

Fig. 13: *An optimal p-alignment for the p-trace in Figure 12.*

events both read (or write) the same data attribute with the same value, then they are concurrent; (2) otherwise, there is a dependency between them.

For example, shown in Figures 11 and 12, events *D* (event 3) and *R* (event 4) (only) have data element *p* in common but both have read the same value for *p* which indicates there is no dependency between *D* and *R*, whereas events *D* (event 3) and *C* (event 5) have data element *p* and *d* in common and since *D* writes *d* and *C* read *d*, we add a dependency from *D* to *C*.

Using the p-trace shown in Figure 12 as input, we compute an optimal p-alignment shown in Figure 13. The fourth entry in Table 1 shows the measurement of this p-alignment. Compared to the s-alignment, the p-alignment in Figure 13 shows the same improvements as the p-alignment in Figure 10. Moreover, the p-alignment computed using data dependencies is able to locate the log move D more precisely than the other two alignments, increasing the true positive d-dependencies to 7.

5 Experimental Results

We implemented our p-alignment approach described in Sections 3 and 4 in the *PartialOrderReplayer* package of the process mining toolkit ProM. The package provides the plug-in named *Partial Aware Replayer*. To evaluate our approach, we designed the hospital model shown in Figure 2 in a tool called CPN Tools and randomly simulated an event log of 1000 traces with in total 6590 events[1]; each trace with 6 to 12 events. All events have the *same* timestamp, and each has 1 to 5 data attributes as specified in Figure 2. We performed four small experiments[2]. For each experiment, we computed three types of optimal alignments using three approaches: (1) s-alignments using the approach in [7]; (2) p-alignments using the p-traces converted based on the approach for Type A logs described in Section 4.1; (3) p-alignments of the p-traces obtained using the approach for Type B logs (with data attributes already annotated) described in Section 4.2. The quality of each optimal alignment is measured with respect to the ideal alignment, which is known since the log is generated artificially.

- **Experiment 1.** In this experiment, the input is the perfectly generated sequential log in which all events are correctly ordered.
- **Experiment 2 with shuffled events.** The perfectly generated event log is used but the events in a trace are randomly shuffled. Thus the ordering of events is unreliable.
- **Experiment 3.** The input is the generated sequential log with deviations added as follows. For each trace, two events are added, and two are removed from the trace. For each event added, a predecessor and a successor are randomly chosen which ensure the true direct causal dependencies (for obtaining ideal alignments only). Each added event is then inserted between the range of its predecessor and successor and has the same timestamps as other events. Each added event reads a data attribute produced by its predecessor and writes an output data attribute being an input to it successor.
- **Experiment 4 with shuffled events.** For this experiment, we randomly shuffled the events of each trace in the log obtained in experiment 3.

[1] The files can be downloaded at https://svn.win.tue.nl/repos/prom/Documentation/PartialOrderReplayer/SBP2014.zip

[2] The implementation of the experiments can be found in the same package of ProM (i.e. the class *ExperimentSBP)*

Table 2: The average results of 10 runs of the four experiments.

		Moves										d-dependencies			
		Sync. Moves			Log Moves			Model Moves							
		TP	FP	FN	TP	FP	FN	TP	FP	FN	F1	TP	FP	FN	F1
Exp 1.	Seq.	6.59	0	0	0	0	0	0	0	0	1*	4.28	1.31	2.31	0.70
	A	6.59	0	0	0	0	0	0	0	0	1*	6.54	0.05	0.05	0.99
	B	6.59	0	0	0	0	0	0	0	0	1*	6.59	0	0	1.00
Exp 2.	Seq.	3.61	0	2.98	0	2.98	0	0	2.64	0	0.71*	3.07	5.16	3.52	0.41
shuffled	A	6.59	0	0	0	0	0	0	0	0	1*	6.48	0.11	0.11	0.98
	B	4.09	0	2.50	0	2.50	0	0	2.29	0	0.77*	5.72	3.44	0.87	0.73
Exp 3.	Seq.	**4.21**	0.60	0.38	1.40	0.38	0.60	1.64	0.08	0.36	0.84	3.92	3.39	5.68	0.46
	A	**3.94**	1.23	0.65	0.77	0.65	1.23	1.39	0.02	0.61	0.69	5.40	1.18	4.20	0.67
	B	**4.03**	0.92	0.56	1.08	0.56	0.92	1.53	0.06	0.47	0.76	6.60	1.96	3.00	0.73
Exp 4.	Seq.	**2.43**	0.84	2.16	1.16	2.16	0.84	1.48	1.51	0.52	0.55	2.93	5.65	6.67	0.32
shuffled	A	**3.93**	1.24	0.66	0.77	0.66	1.24	1.39	0.02	0.61	0.69	5.28	1.29	4.32	0.65
	B	**2.67**	1.04	1.92	0.96	1.92	1.04	1.46	1.19	0.54	0.55	5.23	4.42	4.37	0.54

* denotes F1 scores of synchronous moves. The discussed results are underlined.

Discussion. Table 2 shows the average quality measurements per optimal p-alignment (rounded in two decimals) of ten random executions of each experiment. In addition, we added the average F1-scores of moves (column 12) and of d-dependencies (column 16). The difference between the scores of the three types of p-alignments is significant for experiments 2 to 4[1]. Figure 14 illustrates the confidence intervals of the TP synchronous move rates of experiment 3 and 4 (i.e. the last 6 entries in column 3 of Table 2).

As can be seen in the last four columns of Table 2, in all four experiments, the TP, FP and FN scores of d-dependencies of the p-alignments are improved compared to the s-alignments, identifying 1.5 to 3.4 TP d-dependencies more and 1.2 to 5 FP d-dependencies less per alignment on average compared to the s-alignments. An increase of 40% to 130% percent of the F1-scores of the d-dependencies in the p-alignments confirms this observation.

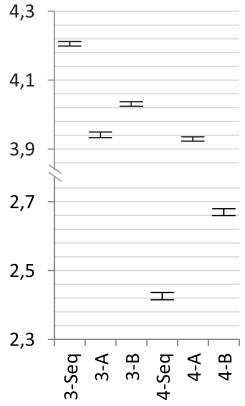

Fig. 14: The CIs of TP Sync. moves of Exp.3 and 4.

The results also show that when the ordering of events is unreliable, i.e. in Exp. 2 and Exp. 4, the two p-alignment approaches identify more TP synchronous moves and less FP log moves and FP model moves than the s-alignment approach, which suggest that the p-alignment approaches are more flexible. However, this flexibility also leads to identifying more FP synchronous moves (in Exp. 3 and 4). The average F1-scores of moves also show that the p-alignment approaches perform at least as good as the s-alignment approach except in situations when the ordering of events is reliable and traces contain noise as in Exp. 3.

Based on the results of the experiments, we have shown that we can obtain better results using the p-alignment approach, especially in unreliable and flexible settings. In addition, the difference between the quality metrics of the two p-alignment approaches indicates that the quality of p-alignments also depends on the quality of derived p-traces.

6 Related Work

Various techniques have been proposed to check conformance between the modeled and observed behavior. The token-based replay approach proposed by Rozinat and Van der Aalst [1] measures the number of remaining tokens and of missing tokens in the process model when replaying the log to provide diagnotics about the quality of the model and deviations in the log. The state-of-art technique in conformance checking is the alignment approach proposed by Adriansyah et al. [6, 2, 7] which can handle complex constructs such as invisible transitions and duplicated transitions while providing detail information on deviations. The resulting diagnostics of these techniques have been applied in various context. For example, it is used to assess the quality of a model with respect to the reality [10], to repair or simplify models based on diagnosed deviations [11, 5], to perform auditing and compliance analysis [12, 3, 2], to find decision points in processes [13], to conduct root cause analysis [14] and performance analysis [4].

While using the result of conformance checking in various applications, much less literature are found in investigating the quality of the input of conformance checking as well as the quality of its results. Bose et al [15] discussed various quality issues found in event logs. The alignment approach assumes that with assigning the right cost to moves the "ideal alignment" can be found in optimal alignments [6, 4] without considering that the log may have quality problems.

In comparison to existing conformance checking techniques, the approach presented in this paper used partially ordered traces and alignments to provide a way to incorporate flexibility, uncertainty, concurrency and explicit dependencies in inputs as well as in outputs of conformance checking to improve the quality of results. Partially ordered traces and runs have been defined and discussed in diverse other settings. Lorenz et al. define partially ordered runs of Petri nets in order to analyze properties of Petri nets [16]. Lassen et al. presented an approach to convert basic message sequence charts into p-traces and used these explicit casual dependencies to improve the process discovery result [17]. Fahland and Van der Aalst used partially ordered runs to simplify process models [11].

7 Conclusion

In this paper, we presented a generic approach for computing partially ordered alignments using partially ordered traces. In addition, we illustrated two ways to obtain partially ordered traces as input for computing p-alignments from given sequential event logs. Furthermore, we introduced a quantitative quality metric to compare alignments with respect to the ideal alignments. The evaluation results show that the quality of p-alignments is improved compared to s-alignments especially in unreliable settings. Our approach provided a first step towards improving the quality of conformance checking in more realistic circumstances.

Future research aims at incorporating probabilistic data to find better p-alignments. In addition, we are also interested in approaches to compute the ideal partially ordered trace. Moreover, partially ordered alignments can be used to analyze data flows or to compute alignments in a distributed manner.

Acknowledgments. This research is supported by the Dutch Cyber Security program in the context of the PriCE project. We thank Boudewijn van Dongen for his support in this work.

References

1. Rozinat, A., van der Aalst, W.M.P.: Conformance checking of processes based on monitoring real behavior. Information Systems **33**(1) (2008) 64–95
2. Adriansyah, A., van Dongen, B.F., Zannone, N.: Controlling break-the-glass through alignment. In: Social Computing, 2013 International Conference on, IEEE (2013) 606–611
3. Ramezani, E., Fahland, D., van der Aalst, W.M.P.: Where Did I Misbehave ? Diagnostic Information in Compliance Checking. (2012) 262–278
4. van der Aalst, W.M.P., 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) (2012) 182–192
5. Fahland, D., van der Aalst, W.M.P.: Model repair aligning process models to reality. Information Systems (2013)
6. Adriansyah, A., van Dongen, B.F., van der Aalst, W.M.P.: Conformance checking using cost-based fitness analysis. In: Enterprise Distributed Object Computing Conference (EDOC), 2011 15th IEEE International, IEEE (2011) 55–64
7. Adriansyah, A., van Dongen, B.F., van der Aalst, W.M.P.: Memory-efficient alignment of observed and modeled behavior. BPMcenter. org, Tech. Rep (2013)
8. Aho, A.V., Garey, M.R., Ullman, J.D.: The transitive reduction of a directed graph. SIAM Journal on Computing **1**(2) (1972) 131–137
9. Manning, C.D., Raghavan, P., Schütze, H.: Introduction to information retrieval. Volume 1. Cambridge university press Cambridge (2008)
10. Buijs, J.C., van Dongen, B.F., van der Aalst, W.M.P.: On the role of fitness, precision, generalization and simplicity in process discovery. In: On the Move to Meaningful Internet Systems: OTM 2012. Springer (2012) 305–322
11. Fahland, D., van der Aalst, W.M.P.: Simplifying discovered process models in a controlled manner. Information Systems **38**(4) (2013) 585–605
12. Cederquist, J.G., Corin, R., Dekker, M.A.C., Etalle, S., den Hartog, J.I., Lenzini, G.: Audit-based compliance control. International Journal of Information Security **6**(2-3) (February 2007) 133–151
13. Leoni, M.d., Dumas, M., García-Bañuelos, L.: Discovering branching conditions from business process execution logs. In: Fundamental Approaches to Software Engineering. Springer (2013) 114–129
14. Suriadi, S., Ouyang, C., van der Aalst, W.M.P., ter Hofstede, A.H.M.: Root cause analysis with enriched process logs. In: Business Process Management Workshops, Springer (2013) 174–186
15. Bose, R.P.J.C., Mans, R.S., van der Aalst, W.M.P.: Wanna improve process mining results? In: Computational Intelligence and Data Mining (CIDM), 2013 IEEE Symposium on, IEEE (2013) 127–134
16. Lorenz, R., Desel, J., Juhás, G.: Models from scenarios. In: Transactions on Petri Nets and Other Models of Concurrency VII. Springer (2013) 314–371
17. Lassen, K.B., van Dongen, B.F.: Translating message sequence charts to other process languages using process mining. In: Transactions on Petri Nets and Other Models of Concurrency I. Springer (2008) 71–85