Conformance Checking in Healthcare Based on Partially Ordered Event Data

Xixi Lu, Ronny S. Mans, Dirk Fahland and Wil M.P. van der Aalst
Department of Mathematics and Computer Science
Eindhoven University of Technology
P.O. Box 513, 5600 MB Eindhoven, The Netherlands

***Abstract*—There is a continuous pressure to make healthcare processes more efficient and effective without sacrificing quality. Conformance checking can be used to improve processes by analyzing event data and directly relating observed behavior and modeled behavior. Conformance checking provides diagnostics that go far beyond measuring traditional key performance indicators. However, current conformance checking techniques focus on a rather simplistic setting where executions of process instances are sequential and homogeneous whereas healthcare processes are known to be dynamic, complex, and ad-hoc. In healthcare process instances of patients often follow a unique path through the process with one-of-a-kind deviations. Moreover, timestamps are often rather coarse (the date is known, but not the time) resulting in an unreliable ordering of events. As current techniques are unable to handle concurrent events, and the obtained sequential alignments are unable to provide structural information about deviations, the diagnostics provided are often insufficient and misleading. This paper presents a novel approach using *partially ordered traces* and *partially ordered alignments* which aims to incorporate unreliability and concurrency in the input while providing diagnostics about deviations that take causalities into account. The approach has been implemented in ProM and was evaluated using event data from a Dutch hospital.**

I. Introduction

There is a tremendous pressure to improve healthcare process quality, efficiency and effectiveness [1], [2]. As part of this undertaking, processes need to be optimized. A way to improve processes is to describe processes in terms of models and investigate their run-time execution using these models. Typically the models only provide an idealized view on how healthcare processes are really executed, i.e, only frequent or idealized scenarios are modeled whereas in reality many more occur. Therefore, various conformance checking techniques were proposed to show diagnostic information about deviations observed in real executions with respect to a given process model [3]. Moreover, insights into relations between a model and observed behavior obtained using conformance checking techniques can be used to enable auditing, compliance analysis, performance analysis and process model enhancements [4]–[7].

Current conformance checking techniques focus more on static environments, such as business processes, where process instances often follow a well-defined workflow. The executions of these process instances are well recorded and have similar conforming and deviating behaviors. In contrast, in healthcare processes are known to be flexible, complex, and ad-hoc [8]. As a result, workflow models are rather considered as guidelines, i.e. it may be well acceptable to deviate from parts of the model. This flexibility leads to a large number of cases deviating from the model. Moreover, a case (e.g. patient) often follows a unique path through the process and may have very specific deviations (e.g. an examination is done twice on a day) which may require the case to be inspected individually.

From the aforementioned healthcare process characteristics it can be concluded that current conformance techniques are unable to deal with cases in flexible and dynamic environments. More specifically, there are two main limitations: (1) using sequential traces as inputs is too rigid and unable to describe concurrent events (e.g. events happened on the same day), and (2) the diagnostic information obtained in sequential alignments gives insufficient and misleading insights into the nature of deviations. Both limitations will be explained in more detail below.

The first limitation is that conformance checking techniques currently use sequential traces as input which heavily rely on the total ordering of events in the traces. In healthcare, the timestamps of events are often too coarse, incorrect, or recorded at a different level of granularity. This issue may lead to an unreliable or incorrect ordering of events in a recorded trace. Figure 1 shows a subset of events of a case found in the case study presented in this paper, which illustrates nicely that timestamps are recorded at different levels of granularity. The timestamp of event *Preoperative Assessment (P)* (i.e. *PANE* in Figure 1) has both a date and a time whereas the timestamps of other events (e.g. *Meeting Surgeon (V_M)* (*Vervolgconsult algemeen - HEE*)) reveal only date information. According to the documented model used in the case study (shown in Figure 7 and discussed in Section IV), first event P and then event V_M should happen whereas the two events are now ordered incorrectly due to the coarse timestamps. Using a non-trustworthy total ordering of the events for computing alignments may result in classifying abnormal behavior as conforming and normal behavior as deviating. In addition, using sequential traces it is not possible to explicitly describe set of events as concurrent. For example, none of the conformance checking techniques use the information that the MRI and the X-ray recorded on the same day can happen in any order.

The other main limitation is that current conformance approaches are often limited to the provision of purely sequential alignments. As a result the moves are totally ordered

	Activity	Resource	Date	Time
10	mri tractus urogenitalis	RDIA	11.07.2011	00:00:00
11	1e consult algemeen	PEHU	13.07.2011	00:00:00
12	opname ziekenhuis	VEC2	13.07.2011	00:00:00
13	Lab Test-LHMA	LHMA	13.07.2011	00:00:00
14	Lab Test-LCHE	LCHE	13.07.2011	00:00:00
15	Lab Test-LHMA	LHMA	14.07.2011	00:00:00
16	Lab Test-LCHE	LCHE	14.07.2011	00:00:00
17	Lab Test-LCHE	LCHE	15.07.2011	00:00:00
18	ontslag ziekenhuis	VEC2	15.07.2011	00:00:00
19	Vervolgconsult algemeen - HEE	PGEO	15.07.2011	00:00:00
20	PANE	RKLyqLV48NP47vAyoWeqvg==	15.07.2011	14:10:00
21	Lab Test-LHMA	LHMA	21.07.2011	00:00:00

Fig. 1: Timestamps of events (labeled with activities names in Dutch) recorded at different levels of granularity.

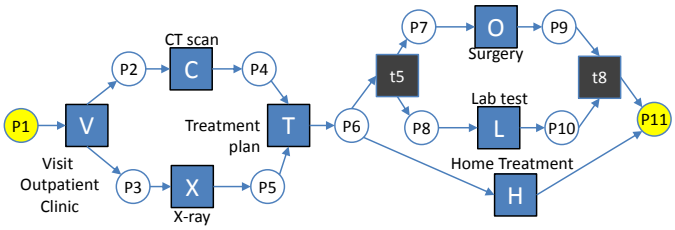

Fig. 2: A simplified healthcare process: each visible transition (i.e. an activity) is denoted by a blue rectangle; an invisible transition is denoted by a gray rectangle; places are denoted by circles; $[P1]$ is the initial marking; $[P11]$ is the final marking.

without explicit dependencies between moves nor their relation to the model. This limitation makes it difficult to inspect an individual alignment while relating the alignment to the process model. Moreover, the diagnostics may point out cases as deviating that are not (false positives) or miss important deviations (false negatives).

To overcome the aforementioned limitations, we propose to use *partially ordered events* rather than totally ordered events. Moreover, we want to compute partially ordered alignments rather than totally ordered alignments. Using a partial order over events, we are able to *express causal dependencies, uncertainty, and concurrency in a better and more explicit way*, which provides the flexibility to deal with coarse timestamps and unreliable ordering of events. By computing partially ordered alignments, we can also provide *diagnostic information about dependencies* defined in the model in comparison to dependencies found in the log. This additional information helps us to pinpoint the deviation found both in the model and in the log much more precisely.

In this paper, we formally define *partially ordered traces* and *partially ordered alignments*. Moreover, we discuss our approach to compute partially ordered traces using timestamps and to compute partially ordered alignments using partially ordered traces. Finally, we discuss the case study which we conducted to evaluate our approach.

The remainder of this paper is structured as follows. Section II recalls basic concepts related to logs and models. In Section III, we formally define the partially ordered traces and alignments and illustrate our partially ordered alignment approach. Section IV presents the result of case study and discusses the insights we obtained. Related work is discussed in Section V. Section VI concludes the paper.

II. Preliminaries

In this section, we recall preliminaries related to logs and models. In addition, we introduce a running example.

A. Running Example

Figure 2 shows a simplified healthcare process in a hospital. The process starts with a patient visiting the outpatient clinic (V). After the visit, both a CT scan (C) and an X-ray scan (X) are performed in parallel (i.e., in any order or at the same time). Based on the result of both scans, a treatment plan for the patient is proposed (T). The patient is then either having a surgery (O) or sent home (H). In parallel to the surgery, a lab test is conducted (L).

B. Behavior

Both event logs and process models serve as the starting point for process conformance checking. A process model describes the life-cycle of cases of a particular type (e.g. patient treatment) as a collection of *possible behaviors*. In contrast, an event log can be seen as a specific *observed behavior* consisting of recorded events. Each *case* going through the process is a *process instance* of the process resulting in a trace of events in the event log.

Definition 1 (Universes). *In the remainder we assume the following universes of names, identifiers, and values: Act is the set of all possible activity names; $Attr$ is the set of all possible attribute names; Val is the set of all possible attribute values; PI is the set of all possible process instance identifiers; EI is the set of all possible event identifiers.*

Process instances and events are represented by unique identifiers. This formalization allows us to refer to a specific event or process instance. This way two events with the same properties and two behaviorally equivalent process instances can be distinguished.

Definition 2 (Partial orders). *A* partial order *over a set E is a binary relation $\prec \subseteq E \times E$ which is (1) irreflexive, i.e. $x \not\prec x$, (2) antisymmetric, i.e. $x \prec y$ implies $y \not\prec x$ and (3) transitive, i.e. if $x \prec y$ and $y \prec z$, then $x \prec z$.*

Definition 3 (Behavior). *$B = (E, act, attr, pi, \prec)$ is a* behavior *if and only if:*

- *$E \subseteq EI$ is a set of events,*
- *$act \in E \to Act$ maps events onto activities,*
- *$attr \in E \to (Attr \not\to Val)$ maps events onto a partial function assigning values to some attributes,*[1]
- *$pi \in E \to PI$ maps events onto process instances, and*
- *$\prec \subseteq E \times E$ is a set of* dependencies *between events and is a partial order on events.*

BH is the set of all behaviors.

[1] $f \in X \not\to Y$ is a partial function with domain $dom(f) \subseteq X$.

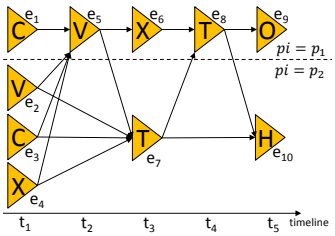

Fig. 3: A Log which is a behavior consisting of events (triangles) related to two process instances p_1 and p_2.

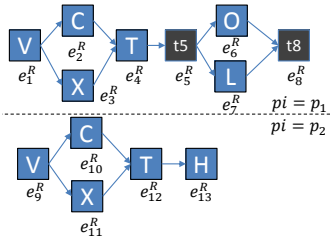

Fig. 4: A run R of a model *Mod* involves two process instances p_1 and p_2 (rectangles denote events in the run).

Given an behavior $B = (E, act, attr, pi, \prec)$ and event $e \in E$: $act(e)$ is the activity corresponding to e, $pi(e)$ is the process instance corresponding to e, $attr(e)$ defines a mapping from selected attributes onto attribute values. $dom(attr(e))$ is the set of attributes having a value in e. Suppose $time \in dom(attr(e))$, then $attr(e)(time)$ is the value of the $time$ attribute for event e. Figure 3[2] exemplifies a behavior $B = (E, act, attr, pi, \prec)$ comprised of 10 events which are involved in two process instances p_1 and p_2:

- the set E of events is $\{e_1, \cdots, e_{10}\}$;
- $act(e_1) = C$, $act(e_{10}) = H$, etc.;
- for $1 \leq i \leq 4$, $attr(e_i)(time) = t_1$, $attr(e_5)(time) = t_2$, e.g., if t_2 = "2014-05-30 11:00", $attr(e_5)(date)$ = "2014-05-30", etc.;
- for $i \in \{2, 3, 4, 7, 10\}$, $pi(e_i) = p_2$, etc.;
- $e_1 \prec e_5$, $e_2 \prec e_7$, $e_2 \prec e_5$, $e_2 \prec e_6$, etc.;
- $\prec$ is a partial order.

For defining behavior, we assume a partial order on events. $x \prec y$ indicates that y depends on the execution of x. Note that a total order is by definition also a partial order.

Definition 4 (Event Logs, Runs and Models). *L is an* event log *if $L \in BH$, i.e., L is a behavior. Mod is a* model *if $Mod \subseteq BH$ and $Mod \neq \emptyset$, i.e., a model is a non-empty set of behaviors. R is a run of model Mod if $R \in Mod$, i.e. a run is a behavior allowed by some model.*

Both *events logs* and *runs* of a process model can be described as *behaviors* (e.g., Figure 3 also shows a log). A process model can have an infinite set of runs (i.e. potential behaviors). If a run of a model is correctly recorded, the run results in a behaviorally equivalent event log. A labeled marked Petri net [9] can be defined as a model *Mod* and have a set of runs. Figure 4 illustrates one run $R = (E, act, attr, pi, \prec)$ of the Petri net model shown in Figure 2. The run R involves two process instances p_1 and p_2. Figure 4 shows that $E = \{e^R_1, \cdots, e^R_{13}\}$, $act(e^R_1) = V$, $e^R_1 \prec e^R_2$, $pi(e_1) = p_1$, etc. Note that by the definition of a behavior, Figure 4 can also be perceived as two runs p_1 and p_2.

[2]For clarity Figure 3 does not show the transitive closure of all dependencies. In general, any directed acyclic graph defines a partial order. For visualization we will often just show the transitive reduction, i.e., the minimal set of arcs whose transitive closure defines a partial.

III. Partially Ordered Traces and Alignments

Next, we formally define partially ordered traces and alignments, followed by our approach to compute them.

A. Definitions

Classical process mining often assumes a total order of events. We generalize the key notions to partial orders.

Definition 5 (P-Trace). *$C = (E^C, act^C, attr^C, pi^C, \prec^C)$ is a partially ordered trace (p-trace) if and only if (1) C is a behavior (i.e., $C \in BH$) and (2) $\forall_{e_1, e_2 \in E^C}\ pi^C(e_1) = pi^C(e_2)$, i.e. all events in C belong to the same process instance.*

Figure 3 exemplifies two p-traces p_1 and p_2 if we neglect the dependencies between the events of different process instances.

Definition 6 (Concurrency). *Let $C = (E^C, act^C, attr^C, pi^C, \prec^C)$ be a p-trace, two events $e_1, e_2 \in E^C$ are* concurrent to *each other, i.e. $e_1 \mid\mid e_2$, if and only if $e_1 \not\prec^C e_2$ and $e_2 \not\prec^C e_1$.*

An alignment relates observed behavior (e.g. a trace) to modeled behavior (e.g. a run) in terms of *moves* and identifies behavior that can not be mimicked between them as deviations. We use the symbol $\gg$ to denote a "no move": $\gg \notin EI$. For any set $E \subseteq EI$: $E^{\gg} = E \cup \{\gg\}$. Let E^C be a set of log events and E^R be a set of run events. $(e^C, e^R) \in ((E^C)^{\gg} \times (E^R)^{\gg})$ is a move. There are four types of moves:

- (e^C, e^R) is a *synchronous move* if $e^C \neq \gg$ and $e^R \neq \gg$
- (e^C, e^R) is a *log move* if $e^C \neq \gg$ and $e^R = \gg$,
- (e^C, e^R) is a *model move* if $e^C = \gg$ and $e^R \neq \gg$, and
- (e^C, e^R) is an *invalid move* if $e^C = \gg$ and $e^R = \gg$.

We use a notion of alignments different from literature. The moves are partially ordered rather than totally ordered.

Definition 7 (P-Alignment). *Let Mod be a model and $C = (E^C, act^C, attr^C, pi^C, \prec^C) \in BH$ be a p-trace, $A = (C, R, M, \prec^M)$ is a partially ordered alignment* (p-alignment) *between the p-trace C and the model Mod if and only if (a) $R = (E^R, act^R, attr^R, pi^R, \prec^R) \in Mod$ is a run of the model, (b) $M \subseteq ((E^C)^{\gg} \times (E^R)^{\gg}) \setminus \{(\gg, \gg)\}$ is a set of* moves*, and (c) $\prec^M \subseteq M \times M$ is a partial order such that:*

1) $\forall_{e \in E^C} \mid \{(e^C, e^R) \in M \mid e = e^C\} \mid = 1$, *i.e., for each event in the log there is precisely one corresponding move;*
2) $\forall_{e \in E^R} \mid \{(e^C, e^R) \in M \mid e = e^R\} \mid = 1$, *i.e., for each event in the run there is precisely one corresponding move;*
3) $\forall_{e^C \in E^C, e^R \in E^R}\ pi^C(e^C) = pi^R(e^R)$, *i.e., events involved in the alignment belong to the same process instance*[3]*;*
4) $\forall_{(e^C, e^R) \in M \cap (E^C \times E^R)}\ act^C(e^C) = act^R(e^R)$, *i.e., events involved in a synchronous move refer to the same activity;*
5) $\forall_{(e^C_1, e^R_1), (e^C_2, e^R_2) \in M}\ ((e^C_1, e^R_1) \prec^M (e^C_2, e^R_2)) \Leftrightarrow ((e^R_1, e^R_2 \in E^R \wedge e^R_1 \prec^R e^R_2) \vee (e^C_1, e^C_2 \in E^C \wedge e^C_1 \prec^C e^C_2))$, *i.e., the ordering of the moves in the alignment respects the ordering in the trace or the ordering in the run.*

[3]Note that the definitions of p-traces and p-alignments restrict the events of the run of a p-alignment to be related to a single process instance.

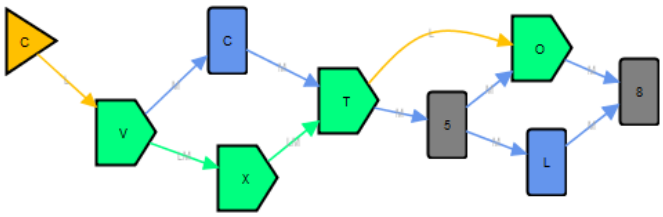

Fig. 5: An optimal p-alignment between the p-trace p_1 and the run p_1: five-sided green polygons denote synchronous moves; blue rectangles denote visible model moves; gray rectangles denote invisible model moves; yellow triangles denote log moves; blue and green arcs denote dependencies in the run; yellow and green arcs denote dependencies in the p-trace.

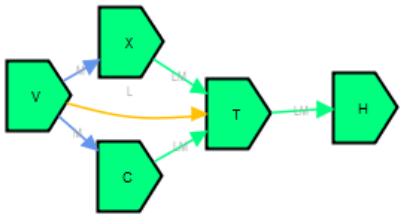

Fig. 6: An optimal p-alignment between the p-trace p_2 and the run p_1.

Given a p-trace and a model, infinitely many alignments are possible, many of which poorly map the observed behavior onto the modeled behavior. Therefore, we typically define a notion of optimal alignments using a *cost function* for moves where model and log deviate (i.e. log moves or model moves). The *standard cost function* assigns each possible log move and model move a cost of 1. The alignment with the least cost is considered to be optimal.

Figures 5 and 6 respectively show two optimal p-alignments between the model in Figure 2 and the two process instances p_1 and p_2 (of which the p-traces are shown in Figure 3, and the runs are shown in Figure 4). Figure 5 shows that the p-alignment of p_1 has the following moves:

- one log move denoted by the yellow triangle labeled with C (i.e. event e_1 in the p-trace p_1 shown in Figure 3);
- two visible model moves denoted by the two blue rectangles labeled with C and L, respectively;
- two invisible model moves denoted by the two gray rectangles labeled with 5 and 8, respectively;
- four synchronous moves denoted by the four five-sided green polygons labeled V, X, T and O.

Moreover, the same color code is used for dependencies between moves: yellow and green dependencies between moves are found in the p-trace (i.e. in $\prec^C$); blue and green dependencies are found in the run. In addition, projecting a p-alignment on the yellow and green parts, we obtain the original p-trace; and projecting a p-alignment on the blue, gray and green parts, we obtain the run. For example, the yellow and green parts of the p-alignment in Figure 5 show the original totally ordered case p_1, with a log move on C, whereas the blue, gray and green parts of the p-alignment shows the run p_1 in Figure 4 with a model move on C and two invisible model moves on t_5 and t_8.

B. Computing P-Traces

One of the situations observed in healthcare domain is that the timestamps of events are often coarse or recorded using different level of granularity (e.g. the timestamps of some events only have the date whereas other events have both date and times), as shown in Figure 1. This quality issue of the timestamps leads to an unreliable ordering of events which the sequential alignment approach can not handle. In contrast, our partial order approach can treat these events as concurrent (i.e., they can occur in any order or at the same time).

A simple approach to compute p-traces in this situation is to assume that the events that happened on the same day are concurrent and neglect the time at which the events are executed. Formally, given a set E^C of events related to a single process instance and functions act^C, $attr^C$ and pi^C, we compute a p-trace $C = (E^C, act^C, attr^C, pi^C, \prec^C)$ with the set of dependencies $\prec^C = \{e_1 \prec^C e_2 \mid e_1, e_2 \in E^C \wedge attr(e_1)(date) < attr(e_2)(date)\}$.

C. Computing P-Alignments

Our method to compute optimal p-alignments extends the A^* approach proposed in [10] and uses p-traces as input.

Using the approach of [11], we first convert a given p-trace into a labeled marked Petri net, also called an *event net* in [10], which represents the observed behavior of the p-trace. After obtaining the event net, we compute the synchronous product between the event net and the process model [9]. Each transition in the synchronous product is a possible move. The synchronous transitions are composed of two transitions referring to the same activity, one in the event net and one in the process model. The two transitions are fused into one. Assigning each transition a cost, we can reuse the A^* algorithm in [10] to obtain an optimal firing sequence with lowest cost from the initial marking to the final marking. Based on the obtained optimal firing sequence in the synchronous product, we can unfold the synchronous product to compute an optimal concurrent run by [12]. This optimal concurrent run can be converted into a partial order of moves, which is our optimal partially ordered alignment [9]. The corresponding run of the alignment is obtained by projecting the moves on the process model transitions.

IV. Experimental Evaluation

We implemented our p-alignment approach in the *Enriched-LogReplayer* package of the process mining toolkit ProM[4]. The package provides the plug-ins named *Advanced Partial Replayer*, *Visualize PTraces as Graphs* and *Visualize PAlignments as Graphs*. These plug-ins are used to conduct a case study of a real-life log of a healthcare process.

[4]www.processmining.org

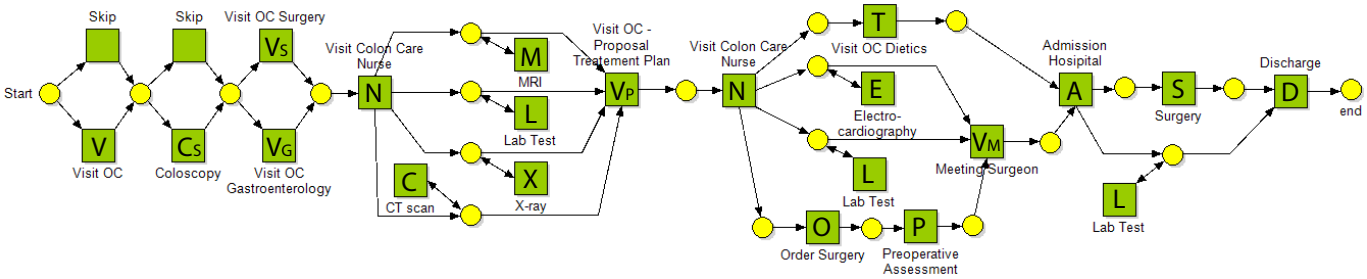

Fig. 7: The process model used in the case study.

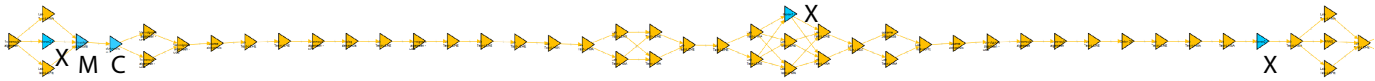

Fig. 8: A p-trace with an undesired, highly sequentialized structure which indicates the patient has to visit the hospital multiple days in which only a single activity is performed.

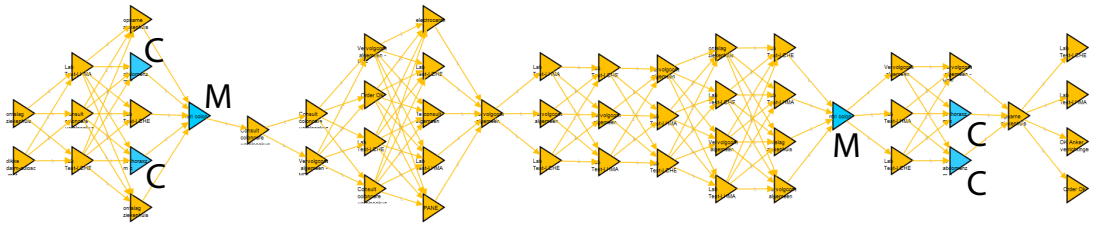

Fig. 9: A p-trace with a more concurrent structure (i.e. more activities are scheduled on the same day); the two MRI events (M) are scheduled on a single day indicating a potential bottleneck.

A. Case Study Settings and Preprocessing

The process model and the (anonymized) event log of this case study were provided by Maastricht University Medical Center (MUMC+), a large academic hospital in the Netherlands[5]. Figure 7 shows the process model for diagnosing and performing surgery on patients with rectal cancer. The process starts with a patient visiting the outpatient clinic (transition V). This visit may also be skipped (*Skip*) (for example, when the patient is sent by the first aid department or a general practitioner). After the visit to the outpatient clinic a coloscopy is performed (Cs) followed by another visit to the outpatient clinic of gastro-enterology (V_G) or surgery (V_S). Next, the patient visits the colon care nurse (N) during which the plan for work-ups are made. Various lab tests (L), CT-scans (C), MRI's (M), and X-ray scans (X) are made before the patient visits the outpatient clinic again (V_P). During this meeting the treatment plan is proposed. Next, the patient visits the colon care nurse (N) again to commence the surgery process. In sequence, a surgery is ordered (O) and preoperative assessment is conducted (P), whereas an electrocardiography (E) and lab tests (L) are conducted concurrently. After having a meeting with the surgeon (V_M), the patient is admitted to the hospital and the surgery is executed (S). Finally, the patient is discharged (D).

We obtained a log containing all relevant information for 34 patients following this process. The events in the log are labeled with 25 distinct activities each of which map to one or multiple transitions in the model. The resulting log consists of 1937 events. The traces have on average 57 events (between 16 and 137 per trace).

[5]http://www.mumc.nl

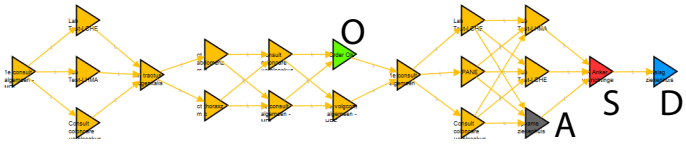

Fig. 10: A p-trace in which the *Order Surgery* (O), *Admission* (A), *Surgery* (S) and *Discharged* (D) are executed in close proximity (few intermediate events).

B. Observations Obtained Using P-Traces

We used the method of Section III-B to obtain p-traces for the case study. Figures 8 to 11 demonstrate a subset of the p-traces we obtained. We highlight the relevant nodes by annotating them with a letter that indicates the activity name. We recall that events concurrent to each other (i.e. in this particular case study, events occurring on the same day) appear on the same vertical line in the figures. Based on the structural perception of these partial orders on events, we discuss three main observations.

First, the structure of the obtained p-traces show that the cases are executed rather sequentially. The p-traces shown in Figures 8 and 11 exemplify this claim. In these two p-traces,

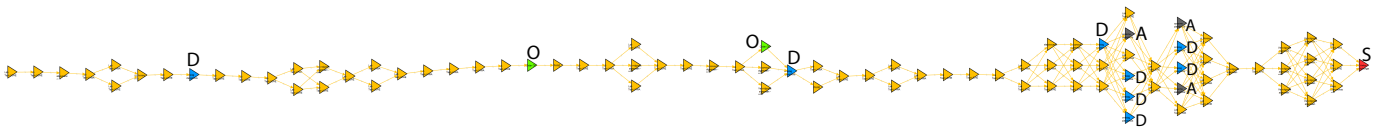

Fig. 11: A p-trace with a highly sequential initial part followed by a fragment having a more parallel structure; The *Order Surgery* (O) and the *Surgery* events are executed far apart from each other.

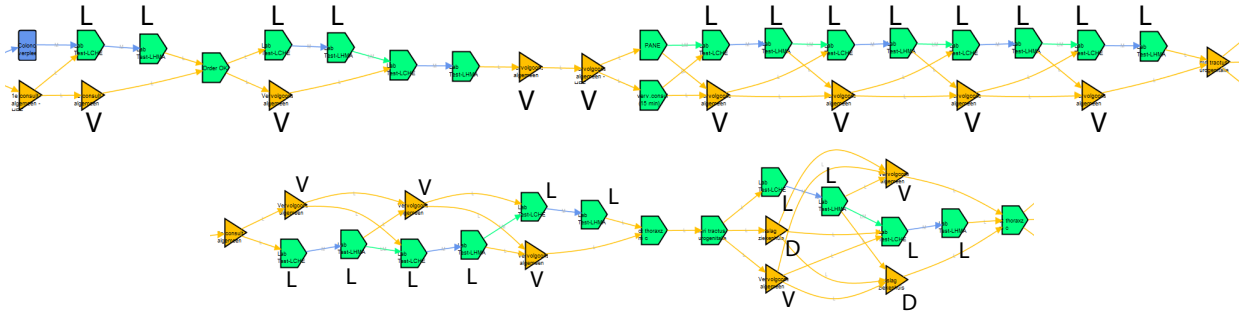

Fig. 12: Two p-alignments showing the patterns 1(a) "*Visit OC* log moves (V) in concurrent with *Lab Test* events (L)".

most events are executed on distinct days. In addition, events labeled with the four activities *MRI* (M), *Lab Test* (L), *X-ray* (X) and *CT scan* (C) should occur in parallel but are often executed on consecutive, distinct days. Obviously, this is not desired. We also notice that *MRI* events often occur in isolation (no other events on the same day). Moreover, MRI events are also rarely executed concurrently with other events. This observation indicates a potential bottleneck. The observed highly sequential structure of the p-traces indicates that patients on average have to visit the hospital frequently for a small set of events.

We also noticed that for 17 out of the 34 cases, two *CT scans* (C) are performed concurrently but no X-ray scan is performed. Figure 9 exemplifies one of these traces. Using domain knowledge, we infer that one of the CT scans (i.e. *CT thorax*) is performed instead of the *X-ray scan (X)* (i.e. *X-thorax*). The reason for this replacement is to decrease the number of appointments patients have to make.

Another interesting observation is that in general when event *Order Surgery* (*Order OK* (O)) is executed close to *Surgery* (*OK Anker* (S)), the overall structure of the entire p-traces of these cases are more simple than the ones of the p-traces in which these two events are executed far from each other. For 14 out of the 34 cases, a *Surgery* event (S) is executed within 7 dependencies after an *Order Surgery* event (O), and for these p-traces, the average length of a shortest path from a start event to an end event (i.e. 20.2 dependencies) is two times less than the average length (i.e. 43.6 dependencies) of a path in the p-traces in which the two events are executed greater than 7 dependencies apart. Figures 10 show one case in which these events are executed closely after each other resulting into a much simpler p-trace. Figure 11 exemplifies a case in which these events are executed far apart from each other.

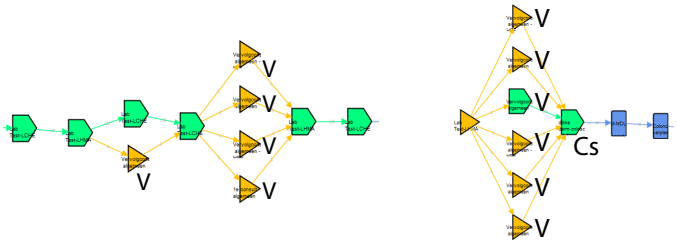

Fig. 13: Two p-alignments showing the pattern 1(b) "*Visit OC* (V) highly in concurrent with itself".

C. Observations Obtained Using P-Alignments

The p-traces obtained are used to compute p-alignments as described in Section III-C. Figures 12 to 17 show a subset of the p-alignments we obtained. We discuss three of our main observations related to the p-alignments:

(1) In the obtained p-alignments, we observe many log moves related to activity *Visit OC* (V) (i.e. events labeled with *Vervolg consult algemeen*) which are found in three different patterns.

a) Figure 12 illustrates the first pattern in two alignments: a sequence of multiple *Visit OC* log moves (V) is often found concurrent to *Lab Test* events (L). Using domain knowledge, we know that a lab test is typically performed before a chemotherapy treatment, which indicates that *Visit OC* log moves may be recorded instead of this treatment.

b) Figures 13 and 14 exemplify the second observed pattern: multiple log moves of *Visit OC* (V) in parallel with itself, i.e. recorded on same days.

c) The third pattern shows that *Visit OC* log moves (V) are executed repetitively in consecutive days, illustrated in Figure 16.

Due to the large amount of the duplicated *Visit OC* log moves (V) found in the log and their general meaning (i.e. can

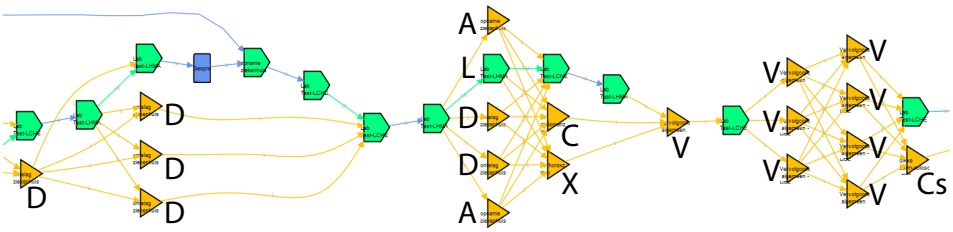

Fig. 14: A p-alignment showing the pattern 1(b) and the pattern (2) "*Discharged* log moves (D) in combination with *Admission* (A)".

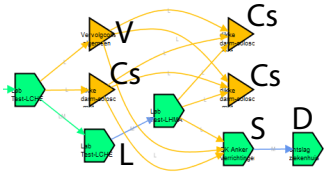

Fig. 15: A p-alignment showing the pattern (3) "duplicated *coloscopy* log moves (C_S)".

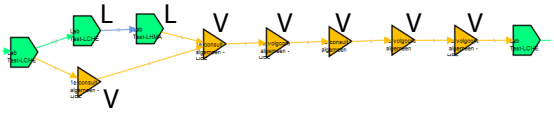

Fig. 16: A p-alignment showing the pattern 1(c) "*Visit OC* log moves (V) in sequence".

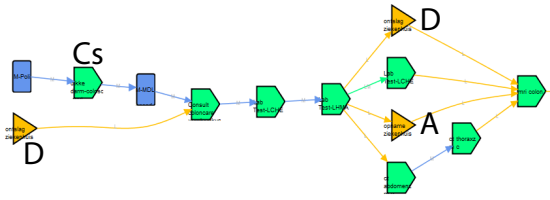

Fig. 17: A p-alignment showing the pattern (2) "*Discharge* log moves (D) in concurrent to *Coloscopy* (C_S) and *Admission* (*A*)".

be mapped to many activities in the model), it may be an interesting direction for business users to further investigate the causes of these log moves. For example, these duplicated log moves may refer to a single visit of the patient in which multiple medical specialists participate or are they related to erroneous registrations.

(2) We also found log moves labeled with *Discharged (D)* (i.e. events labeled with *ontslag ziekenhuis*) in the early stage of some process instances and in parallel with other moves such as *Coloscopy* (C_S) (labeled with *dikke darm-coloscopy*, 21 times), *Admission Hospital (A)* (labeled with *opname ziekenhuis*, 12 times), or *Visit OC (V)* (14 times). This is illustrated in Figures 14 and 17. This indicates that the meaning of the *Discharged* log moves (A) is rather confusing and the logging is inconsistent. Using domain knowledge, we infer that a *Discharged (D)* in combination with *Admission Hospital (A)* indicates that the patient is admitted to the hopital (e.g. because of surgery). Otherwise, the patient simply visits the daycare center (e.g. because of a colonoscopy) and left the hospital after the visit.

(3) Next we consider the duplicated log moves involving activity *Coloscopy* (C_S) which are found in parallel or on consecutive days in the alignments (see Figure 15). This observation may indicate multiple specialists involved in the examination or duplicated logging in systems which can potentially obstruct the accounting processes (e.g. an expensive test for a patient is logged multiple times in different departments).

Despite the many log moves we discussed, the obtained p-alignments reveal a certain "conforming" path of the execution: a path of synchronous moves through the process. This suggests that the processes are following the general procedure documented in the process. The log moves indicate additional events required for the particular patient and situation.

D. Discussion

Recall the two aims described in Section I, we would like (1) to use the more flexible partial order structure while being able to handle concurrent events for conformance checking, and (2) to obtain individual alignments in which the dependencies between moves and its relation to the model are clearly and explicitly defined. The result of the case study shows that partially ordered alignments have indeed achieved these two aims. The p-traces obtained show useful insights into the structure and parallelism of executions. In addition, relaxing the total ordering of events to a partial order, our partially ordered alignment approach identifies 28 (out of 1479) synchronous moves more than the sequential alignment approach.

More importantly, the obtained p-alignments show deviating patterns in structure and concurrency that can be easily recognized. Each deviation is nicely positioned concurrently or between synchronous moves and can be discussed in this context and in a structured way. Both moves and dependencies found in the p-alignments are explicit and their relations to the model are preserved. These properties allow users to inspect radically deviating cases in a structured way, which might be more appropriate than discussing all cases together in a single analysis.

Also from a practical point of view our results are interesting for experts of the hospital that provided the log. For example, from a service level point of view we noticed that at many days only one appointment is taking place. In general patients prefer to have multiple appointments on one day so that they have to travel less to the hospital. As a suggestion, more appointments may be scheduled on one day. Furthermore, from our observations it could also be seen that registrations may be confusing. So, from an accounting perspective it could be potentially problematic for the hospital to demonstrate which services have been delivered to patients. From a costing point it may be the case that services have been delivered and that the hospital is not paid for them or the other way around. Finally, from a planning point of view, erroneous registrations may lead to wrong calculations with regard to e.g.

resource utilization or waiting times of activities. For example, do events labeled with *Visit OC (V)* refer to a visit of a patient to the outpatient clinic or do they refer to a chemotherapy that is given. To summarize, the way services are registered may be improved to further optimize the process.

V. Related Work

Many approaches have been proposed to check conformance between observed behavior and modeled behavior. In [4] a token-based replay approach is discussed which measures the number of remaining and missing tokens when replaying the observed behavior in the model and consider these as deviations. Andriansyah et al. present an alignment approach using the A^* algorithm [10]. The alignment approach uses sequential traces as input and computes a sequential run allowed by the model that has the least number of deviations (e.g. missing events or observed prohibited events) compared to the traces. Various applications are built on the resulting diagnostics provided by conformance checking techniques. For example, model repair techniques use the diagnostics to improve the model structure [13]. Moreover, the diagnostics are enriched with data attributes and timestamps to conduct data-aware conformance checking [14], root cause analysis [15], and performance analysis [3]. In combination with classification techniques, the results of alignments are also used to find decision points in process model.

In the context of Petri nets, many publications related to partial orders of events can be found. Nielsen et al. defined elementary event structure and discussed the construction of partially ordered runs [11]. Desel et al. [12] and Lorenz et al. [16] discussed in their work the construction of partially ordered runs by unfolding the net using traces and analyzed the properties of Petri nets using partially ordered runs. On the contrary, little work in the literature considers incorporating partial orders of events in process mining techniques. Both process discovery and process conformance checking techniques often assume a total ordering of events. Lassen et al. discussed an approach to convert basic message sequence charts (MSCs) that already have explicit partial order structure into p-traces to synthesize a high-quality process model from MSCs [17]. Fahland and Van der Aalst used partially ordered runs of fitting traces to simplify process models [13]. Our approach distinguishes itself from other conformance checking techniques by using partially ordered events as input. Moreover, we compute partially ordered alignments between partially ordered events and runs thereby providing much better diagnostics.

VI. Conclusion

In this paper, we formally defined partially ordered traces and partially ordered alignments using a unified notion of behavior. The approach presented in this paper for computing partially ordered alignments by using partially ordered traces is generic and can also be used for sequential traces. A case study shows that both partially ordered traces and partially ordered alignments are able to handle concurrency and reveal useful insights into the structural information of process executions in addition to deviating events. Future work aims at applying partially ordered alignments in various contexts such as model repair and data flow analysis and using the partial orders to compute alignments in a distributed manner.

Acknowledgment

This research is supported by the Dutch Cyber Security program in the context of the PriCE project and by the Dutch Technology Foundation STW, applied science division of NWO and the Technology Program of the Ministry of Economic Affairs.

References

[1] B. Cardoen, E. Demeulemeester, and J. Beliën, "Operating Room Planning and Scheduling: A Literature Review," *European Journal of Operational Research*, vol. 201, no. 3, pp. 921–932, 2010.

[2] M. T. Taner, B. Sezen, and J. Antony, "An Overview of Six Sigma Applications in Healthcare Industry," *International Journal of Health Care Quality Assurance*, vol. 20, no. 4, pp. 329–340, 2007.

[3] W. M. P. van der Aalst, A. Adriansyah, and B. van Dongen, "Replaying History on Process Models for Conformance Checking and Performance Analysis," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 2, no. 2, pp. 182–192, 2012.

[4] A. Rozinat and W. M. P. van der Aalst, "Conformance Checking of Processes Based on Monitoring Real Behavior," *Information Systems*, vol. 33, no. 1, pp. 64–95, 2008.

[5] E. Ramezani, D. Fahland, and W. M. P. van der Aalst, "Where Did I Misbehave ? Diagnostic Information in Compliance Checking," 2012, pp. 262–278.

[6] M. de Leoni, M. Dumas, and L. García-Bañuelos, "Discovering Branching Conditions From Business Process Execution Logs," in *Fundamental Approaches to Software Engineering*. Springer, 2013, pp. 114–129.

[7] D. Fahland and W. M. P. van der Aalst, "Model Repair Aligning Process Models to Reality," *Information Systems*, 2013.

[8] A. Rebuge and D. Ferreira, "Business Process Analysis in Healthcare Environments: A Methodology Based on Process Mining," *Information Systems*, vol. 37, no. 2, 2012.

[9] G. Winskel, "Petri Nets, Algebras, Morphisms, and Compositionality," *Information and Computation*, vol. 72, no. 3, pp. 197–238, 1987.

[10] A. Adriansyah, B. F. van Dongen, and W. M. P. van der Aalst, "Memory-Efficient Alignment of Observed and Modeled Behavior," *BPMcenter. org, Tech. Rep*, 2013.

[11] M. Nielsen, G. Plotkin, and G. Winskel, "Petri Nets, Event Structures and Domains, Part I," *Theoretical Computer Science*, vol. 13, no. 1, pp. 85–108, 1981.

[12] J. Desel and W. Reisig, "Place/Transition Petri Nets," in *Lectures on Petri Nets I: Basic Models*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 1998, vol. 1491, pp. 122–173.

[13] D. Fahland and W. M. P. van der Aalst, "Simplifying Discovered Process Models in A Controlled Manner," *Information Systems*, vol. 38, no. 4, pp. 585–605, 2013.

[14] M. de Leoni, W. M. P. van der Aalst, and B. F. van Dongen, "Data- and Resource-Aware Conformance Checking of Business Processes," in *Business Information Systems*. Springer, 2012, pp. 48–59.

[15] S. Suriadi, C. Ouyang, W. M. P. van der Aalst, and A. H. M. ter Hofstede, "Root Cause Analysis with Enriched Process Logs," in *Business Process Management Workshops*. Springer, 2013, pp. 174–186.

[16] R. Lorenz, J. Desel, and G. Juhás, "Models from Scenarios," in *Transactions on Petri Nets and Other Models of Concurrency VII*. Springer, 2013, pp. 314–371.

[17] K. B. Lassen and B. F. van Dongen, "Translating Message Sequence Charts to Other Process Languages Using Process Mining," in *Transactions on Petri Nets and Other Models of Concurrency I*. Springer, 2008, pp. 71–85.