Directly Follows-Based Process Mining: Exploration & a Case Study

Sander J.J. Leemans, Erik Poppe, Moe T. Wynn
Queensland University of Technology
Brisbane, Australia
{s.leemans, e.poppe, m.wynn}@qut.edu.au

***Abstract*—Many organisations now seek to analyse and improve their processes using event logs from various IT systems supporting their operations. Process mining aims to obtain insights from such process data, using process discovery, conformance checking and performance measures. While many commercial process mining tools feature user-friendly directly follows-based process maps, they typically do not offer a way to assess the quality of the model, leaving users with potentially unreliable insights, which could lead to the wrong conclusion being drawn from these insights. In contrast, academic tools typically provide verifiable results, but are often difficult to use and understand for stakeholders, sometimes overgeneralising behaviour to fit more extensive process model formalisms. In this paper, we bridge this well-known gap between commercial and academic tools by combining sound process discovery, conformance checking and performance capabilities with user-friendly directly follows-based process models. We implemented these techniques in a new process mining tool and applied them to analyse several business processes in a Queensland Government department. We discovered sound directly follows-based process models from their logs, compared them with prescribed models and analysed the performance of these processes. In particular, our conformance checking techniques allowed to pinpoint deviations between prescribed processes and actual recorded behaviour. The outcomes of this case study are now being used to document, review, improve and automate processes.**

I. Introduction

Organisations store large amounts of data nowadays, recorded from process executions in workflow systems, web applications or tracking technologies. From such transaction data (e.g., purchase order, customer journey, insurance claim, travel booking), an event log can be generated that captures the process steps executed for a trace. Process mining techniques and tools aim to derive meaningful information from such recorded event logs [1].

A typical first step in a process mining study is to automatically discover a process model from an event log, that shows the process steps (activities) that were performed and their order. Such a process model can then be used to compute performance measures, and identify bottlenecks and staff workload. Thus, it is imperative that a process model reflects the 'reality' found in an event log as much as it can. In many cases, process discovery techniques make trade-offs between the four quality dimensions of fitness, precision, simplicity and generalisation. One way to measure the deviations between the log and a discovered model is by performing conformance checking analysis.

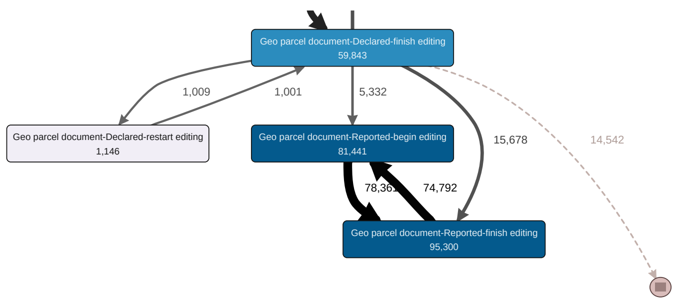

Geo parcel document-Declared-finish editing
59,843
1,009
1,001
5,332
15,678
14,542
Geo parcel document-Declared-restart editing
1,146
Geo parcel document-Reported-begin editing
81,441
78,361
74,792
Geo parcel document-Reported-finish editing
95,300

Fig. 1. Excerpt of a directly follows-based model.

Academic process mining tools typically use modelling formalisms with well-defined semantics, such as Petri nets, BPMN models and process trees, and support complex constructs such as concurrency, interleaving and inclusive choices. These semantics and constructs make them suitable for complex processes, but reduce simplicity [2]. For rather simple business processes, traditional process discovery techniques might need to overgeneralise in order to fit their representational bias [3] or return models with deadlocks or other anomalies (unsound models) [4].

In contrast, many commercial process mining tools exist that typically discover process models in the form of directly follows-based process maps. That is, their visualisations show which activities can follow one another directly. These commercial tools also project time and frequency based performance measures such as execution times, wait times and average or maximum activity counts on these maps. Due to the simplicity of these maps and the interactive filtering options offered by these commercial tools, these tools have captured a large market share[1].

Unfortunately, those same characteristics can make the interpretation of these maps and the related performance measures inaccurate. Some of these limitations are illustrated in Figure 1. Figure 1 shows an example of a directly follows-based model that has been filtered: cases start at the top of the model and progress down through activities to the circle at the bottom. As a result of filtering, the semantics of this model are not clear, as there is no way to reach the end circle from the two bottom-most blue activities, and the numbers, indicating how often each activity and edge has been executed, seem to be inaccurate, as 29000 cases start and only 14500 end.

[1]See https://www.gartner.com/doc/3870291/market-guide-process-mining

Without a clear understanding of the effects of filtering on such numbers, it is easy to draw the wrong conclusions from filtered models.

Conformance checking analysis on directly follows-based models can benefit the stakeholders in evaluating the quality of discovered models. However, although conformance checking has been part of academic process mining tools for some time, support in commercial tools is rather sparse to date. Furthermore, none of the mentioned commercial tools allows the manual import of a process map, for instance representing an authoritative model, against which the behaviour in the log can be compared.

This paper bridges the gap between current directly follows-based maps produced by commercial tools and well-founded but complex process models generated by academic tools. We show how a directly follows-based process mining tool can support process discovery with well-defined semantics, conformance checking and reliable performance measures. We present a case study where this tool has been used to analyse multiple processes from a Queensland Government department. It is our hope that these findings will inspire commercial tools to adopt a similar notion in future release of these tools.

In particular, this paper contributes:

- A definition of *directly follows models* (DFMs), their semantics and their soundness;
- The application of conformance checking to DFMs;
- A new way to discover DFMs by filtering infrequent behaviour while guaranteeing soundness and a minimum fitness level;
- A description of reliable performance measures for DFMs;
- A user-friendly implementation (tool) that includes process discovery, conformance checking and performance measures for DFMs; and
- A case study in which we applied conformance checking and performance measures to (hand-made and discovered) directly follows models and event logs from a Queensland Government department.

We discuss related work in Section II and introduce basic concepts in Section III. In Section IV we discuss how DFMs can be discovered from event logs, in Section V how standard conformance checking techniques can be applied and their results visualised, and in Section VI how performance measures can be computed reliably. In Section VII, we describe the case study performed at the Queensland Government, and we conclude the paper in Section VIII.

II. Related Work

This section briefly summarises recent work on process model formalisms, process mining tools, types of process mining analysis (such as conformance checking and performance analysis) and case studies.

a) Process Model Formalisms: In [2], three drivers for directly follows-based models in commercial tools are given: simplicity (Petri nets or BPMN models with too many gateways are considered un-interpretable), vagueness (ability to express relationships that cannot be interpreted in a precise manner), and scalability (ability to handle logs with millions of events interactively).

The most basic process modelling notation is a transition system which consists of states and transitions between states that correspond to activities being executed. All atomic (i.e. steps do not take time) business processes, such as the higher-level Petri nets [5], some BPMN models [6], process trees [3] and Declare models [7], can be expressed as transition systems.

Directly follows models (DFMs) are variants of transitions systems and lie in between transition systems and higher-level languages: while in a transition system the execution of an activity is atomic, in DFMs this takes time. However, in contrast to the higher-level languages, DFMs do not support concurrency. Furthermore, there is a difference in focus between transition systems and DFMs: while DFMs focus on the ordering of activities, transition systems focus on the state of a process.

BPMN is a standard that provides both graphical representations and execution semantics for processes. BPMN supports many advanced behaviour features such as concurrency, event handling and hierarchy and has found significant adoption by industry, but its complexity means that it requires training for stakeholders to be correctly understood.

While in [2] a new hybrid process modelling concept is proposed that combines formal (Petri nets) with informal ("sure" and "unsure") parts, our approach aims to illustrate the possibilities for directly follows-based models as used by commercial tools. That is, we keep the simplicity and address the vagueness by providing formal semantics for DFMs, defining soundness and provide reliable performance measures, although scalability might suffer from our use of alignments (see Section III).

b) Process Mining Techniques and Tools: In this paper, we consider three types of process mining techniques: process discovery (discovering a process model from an event log), conformance checking (comparing a process model with an event log) and performance measuring.

Discovering directly-follows models, while simple, will often produce large models, so many directly follows-based discovery tools attempt to reduce complexity through abstraction and aggregation. For instance, Heuristic Miner [8] removes behavioural relations based on occurrence frequency. Fuzzy Miner [9] similarly removes directly follows relations based on several heuristics, but supports abstraction by grouping activities in addition. Both approaches, however, can produce unsound models, as they remove behaviour that could be necessary to reach the end state of the process, thus potentially leaving the semantics of the produced models unclear.

Discovery algorithms typically generate process models of higher-level process model formalisms, containing high-level process models with constructs such as concurrency, inclusive choices and interleaving. However, some algorithms may return unsound models (α [1], Split Miner [4], BPMN miner [10], Fodina [11]), require considerable computations times (Evolutionary Tree Miner [12]) or overgeneralise behaviour (Inductive Miner [3]). Furthermore, some techniques generalise transition systems to higher-level process models [13], [14].

Commercial process mining tools, such as Fluxicon Disco (FD) [15], Celonis Process Explorer (CPE), ProcessGold (PG) and myInvenio (MI) [2], typically generate directly follows-based "models". Also the academic Fuzzy Miner (FM) [9] and other approaches [16] focus on directly follows-based models. In order to reduce the complexity of models, these tools seem to filter based on edges (FD, FM, CPE) and/or traces (CPE, PG), which may lead to models with soundness issues (see Section IV). The models are simplistic and do not capture concurrency. However, they can be interpreted easily by domain experts and are thus more easily accepted by users.

In this paper, we define soundness for DFMs and introduce a discovery technique that guarantees DFM-soundness.

c) Conformance Checking and Enhancement Techniques: Conformance checking techniques provide ways to compare behaviour in event logs with behaviour described by process models, which can be either discovered or created manually. As the expressiveness of a process modelling language has a direct effect on the simplicity of models and analysis capabilities, conformance checking techniques are mostly based on process models represented as a Petri net or process tree. Alignments [17], [7], explained in Section III, can be used to identify deviating behaviour. Existing work on conformance checking using directly follows process models has been limited to rule based, outlier (low-frequency behaviour) based or graph-based approaches (as in MI), in which the user has to find visual differences across multiple models. CPE supports conformance checking, however in a disjoint component: it supports a different syntax (BPMN), discovery technique and concepts, and might yield unintuitive results[3].

Process enhancement techniques derive additional information such as performance data using a process model and an aligned event log, see for instance [18], [19], [20]. In Section VI, we will show that existing tools might yield unintuitive performance results.

Our approach highlights deviations between log and model based on alignments and provides reliable performance metrics by projecting alignments onto directly-follows models.

d) Case Studies: Process mining techniques have been used in a number of process mining case studies in various domains such as insurance [18], healthcare [21] education [22] and government services [23]. In a number of such case studies, comparing the performance of different process variants is one of the key activities. When such comparative performance analysis is conducted on directly follows maps without careful attention paid to the abstractions made to the underlying process models, the results are not comparable and reliable. Our approach realigns the log with the abstracted model, making the performance measures reliable and comparable.

III. Preliminaries

a) Event Logs: An *event log* is a collection of *traces*, each consisting of the process steps to be executed. We refer to the steps as *activities* and their execution in a trace as *activity instances*. Each activity instance might traverse a certain life cycle. In this paper, we assume a simple cycle of an optional start and a completion. That is, each trace is a sequence of *events*, which represent the life cycle transitions of activity instances. For instance, $[\langle a_s, a_c, b_c, c_s, c_c\rangle, \langle a_s, a_c, c_s, c_c, b_c\rangle]$ is an event log consisting of two traces: in the first trace, first an instance of the activity a started, then it completed, then an instance of b completed (the start of this instance of b was not recorded) and finally c was started and completed. We assume that in each trace, each start event has a corresponding completion event, but that completion events do not need a corresponding start event. We chose this asymmetry as the event logs we encountered in practice typically contain completion events but rarely contain start events [24]. Similarly, a *language* is a set of traces, and the empty trace is denoted by ϵ.

b) Petri Nets: A Petri net is a graph consisting of *places* that can contain *tokens*, the presence of tokens determines the state of the net, as well as *transitions* which change the state of the net by consuming and producing tokens from/to connected places [5] and emitting an associated activity by *firing*. In a *workflow* (Petri) net, there is a single place without incoming (source) and a single place without outgoing (sink) transitions, and each place and transition is on a directed path between these two places. A workflow net is *sound* if all its transitions can be fired, and from each reachable state the end state (having the only token in the net in the sink) is reachable [1].

c) Alignments: An *alignment* is a combination of a trace and a path through the model that produces that trace. If the trace cannot be produced by the model, then a flawless combination is not possible. Two types of deviations exist: skipping events in the trace (a *log move*) or skipping activities in the model (a *model move*) [17]. An alignment is *optimal* if it has the least cost of all possible alignments, which is typically chosen to be the number of deviations.

For instance, consider the DFM shown in Figure 2 and the trace $\langle a_s, a_c, c_s, c_c\rangle$. Then an optimal alignment is:

log	a_s	a_c	c_s	c_c	-
model	a_s	a_c	-	-	b_c

The first two moves are *synchronous*, while the third and fourth moves are log moves on c, and the last move is a model move on b.

d) Directly Follows Models: Syntactically, a *directly follows model* (DFM) is a directed graph in which the nodes are annotated with either an activity, $start$ or end:

[2]See https://www.celonis.com/, https://processgold.com/en/, https://www.my-invenio.com/

[3]See https://community.celonis.com/t/unexpected-results-in-conformance-checking/1183.

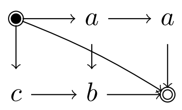


Fig. 2. Example of a directly follows model (DFM).

Definition 1 (Directly follows model - syntax)**.** *Given an alphabet Σ such that $start \notin \Sigma$ and $end \notin \Sigma$, a* directly follows model *is a directed graph (N, E), such that $N\colon \Sigma \cup \{start, end\}$ is a set of nodes and $E\colon N \times N$ is a set of edges. Additionally, start has no incoming and end has no outgoing edges: $\neg\exists_n (n, start) \in E$ and $\neg\exists_n (end, n) \in E$.*

Semantically, the language of a DFM consists of all traces that can be obtained by starting in the $start$-node and walking along the edges, adding events for each activity encountered, to the end-node. Similar to event logs, a start event is optional while a completion event is mandatory for each activity to ensure the activity is actually executed.

Definition 2 (Directly follows model - semantics)**.** *Let $D = (N, E)$ be a DFM. Then, the language of D ($\mathcal{L}(D)$) is $\{\langle a_{1s}, a_{1c}, \dots a_{ns}, a_{nc}\rangle \mid a_1 \dots a_n \in N \wedge (start, a_1) \in E \wedge \forall_{1 \leq i < n} (a_i, a_{i+1}) \in E \wedge (a_n, end) \in E\} \cup \{\epsilon \mid (start, end) \in E\}$, where all the $_s$-events are optional.*

For instance, Figure 2 shows a graphical representation of a DFM. The language of this DFM is $\{\epsilon, \langle a_s, a_c, a_s, a_c\rangle, \langle a_s, a_c, b_s, b_c\rangle, \langle c_s, c_c, b_s, b_c\rangle\}$.

Notice that the DFM formalism supports both the empty trace ($\{\epsilon\}$) as well as the empty language ($\emptyset$). However, concurrency is not supported by DFMs. For instance, there is no DFM that can represent the trace $\langle a_s, b_s, b_c, a_c\rangle$, as DFMs inherently cannot represent concurrency.

IV. Df-Process Discovery

Many commercial and some academic tools offer DFM-discovery, however they might produce unsound models. In this section, we first define soundness for DFMs, after which we introduce a discovery method that results in sound DFMs and offers a user-chooseable fitness guarantee.

A. Soundness

As stated in [1], unsound workflow nets might be useful for limited manual human analysis, but issues such as unexecutable activities, livelocks and deadlocks challenge both human and in particular automatic analysis.

In existing directly follows-based tools, a DFM is constructed, after which edges are filtered out based on a user-chosen parameter to reduce the complexity of the model. However, this may lead to soundness issues, as illustrated in Figure 1, which shows a DFM-like model. In this model, from the two middle-bottom activities, the end state cannot be reached. Despite the visual suggestion, these activities cannot be part of a trace that starts and ends properly, so the language of the model remains unclear.

We define soundness for DFMs, and show the connection between soundness of DFMs and soundness of workflow nets.

Definition 3 (DFM soundness)**.** *Let (N, E) be a DFM. Then, the DFM is* sound *if every node $\in N$ is on a path from start to end:*

$$\forall_{x \in N} \exists_{a_1 \dots a_n \in N} a_1 = start \wedge a_n = end \wedge \exists a_j = x$$
$$\wedge \forall_{1 \leq i < n} (a_i, a_{i+1}) \in E$$

As a consequence of this definition, the DFM without any edges is not sound, as the $start$ and end nodes are not on a path as requested by the definition. This implies that a sound DFM with the empty language does not exist, which is analogous to workflow nets: a workflow net with the empty language cannot be sound.

To further establish the link between sound DFMs and sound workflow nets, we show that Definition 3 is sufficient to translate a DFM to a sound workflow net:

Lemma 1. *A sound DFM can be translated to a language-equivalent sound workflow net.*

We will prove this lemma in Section V-A.

B. Trace-Based DFM Discovery

In order to discover a DFM from an event log, as DFMs do not support concurrency, first all start events need to be removed from the traces. Then, a DFM can be obtained from an event log straightforwardly by sequentially following the events for traces in the log and adding arcs to the DFM accordingly. However, for a typical real-life process, this yields complex and unreadable DFMs.

Therefore, existing tools filter the DFM, either by removing infrequent *traces* before discovery or infrequent *edges* after discovery (next to removing activities). Tools that remove edges do so using certain criteria, typically removing the least-occurring edges. However, this strategy might yield unsound DFMs, as shown in Figure 1, which, when translated to Petri nets (Section V-A), might not result in sound workflow nets.

Tools that aim to filter infrequent behaviour by removing traces from the event log have to figure out which traces represent infrequent behaviour and should be removed. In complex processes, many or all traces might occur rarely, thus one needs to find the infrequent behaviour to decide which traces to remove [25]. *Trace-based DFM discovery* combines these two approaches:

1) First, a DFM is constructed by adding, for each trace, nodes and edges accordingly, while keeping track of how often each edge occurs in the log;
2) Second, the least frequent edges of the DFM are selected;
3) Third, all traces that use the selected edge are removed from the event log;
4) This procedure is repeated until a user-chosen threshold of removed traces is about to be exceeded. That is, the user-chosen threshold indicates how many traces will at most be filtered from the DFM.

Consequently, the DFM returned fits at least *threshold* of the traces in the event log (if only completion events are considered), which is a guarantee not offered by other process discovery techniques.

V. DF-Conformance Checking

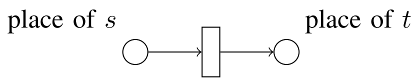
Conformance checking techniques compare an event log with a process model, which can be constructed either manually (to indicate prescriptive behaviour) or by process discovery techniques. The state-of-the-art technique to perform detailed conformance checking is to compute alignments on sound Petri nets (see Section III). In this section, we describe how alignments can be applied to DFMs and how the results can be visualised on DFMs, in order to show them in DF-based [commercial] tools. That is, we first describe how DFMs can be translated to Petri nets, and that sound DFMs yield sound workflow nets. Second, we introduce directly follows-specific alignment visualisation concepts.

A. Translation to Petri Nets

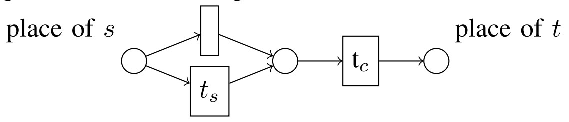
In Definition 2, we defined the language semantics of DFMs. To enable the easy use of DFMs in other process mining tools, we provide a translation to standard Petri (workflow) nets. In this translation, each transition is modelled as a combination of a start and a completion event, such that performance measures can be computed. For a DFM (N, E), the translated Petri net consists of:

- A place for each node $\in N$ (this includes places belonging to $start$ and end);
- For each edge $(s, t) \in E$, a subgraph connecting the place belonging to s to the place belonging to t. This subgraph depends on whether t is end.

  If t is end, then this subgraph is a silent transition connecting s to t:

  If t is not end, then this subgraph executes t_s and t_c in sequence, where t_s is optional:

  This last step ensures that start and completion events are accounted for in the alignments, while keeping the start events optional (as these are not always present).

For instance, Figure 3 shows the Petri net as translated from the DFM in Figure 2. Given this translation procedure, we can show that if a sound DFM is translated, a sound workflow net is obtained:

Proof of Lemma 1. Consider the translation of Section V-A: By the $start$- and end-nodes (Definition 1) and Definition 3, the translated Petri net is a workflow net. As the translation does not introduce transitions with multiple outgoing or incoming edges, the workflow net is 1-bounded (safe). Hence, the translated workflow net is sound. Language equivalence follows by construction of the translation. Hence, the translated Petri net is a language-equivalent sound workflow net. □

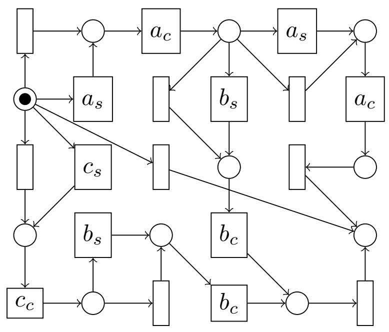


Fig. 3. Example of the DFM of Figure 2 translated to a Petri net.

Once a sound DFM has been translated into a workflow net, any standard Petri net-based conformance checking technique can be applied. For instance, fitness [17], [19], precision [26], [19] and simplicity can be measured, which makes Trace-Based DFM discovery a "full" discovery technique, which could be compared with other process discovery techniques. However, such a qualitative comparison is outside the scope of this paper.

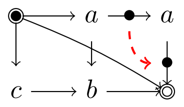
B. Alignment Visualisation

Using a Petri net and the event log, alignments can be computed. Alignments have been visualised on Petri nets [17] and process trees [20]. For DFMs, one could use the Petri net alignment visualisation, however this would require users to familiarise themselves with a new visualisation of their DFM.

Alignments result in two types of deviations: model moves and log moves. We discuss the visualisation of both types of deviation in more detail. Model moves are visualised with an edge bypassing a node in the model, indicating that an event should have been executed according to the model, but it was not seen in a trace of the event log. For instance, consider the DFM of Figure 2 and the trace $\langle a_s, a_c \rangle$. Then, an optimal alignment is

log	a_s	a_c	-
model	a_s	a_c	a_c

, which can be visualised as follows on the DFM:



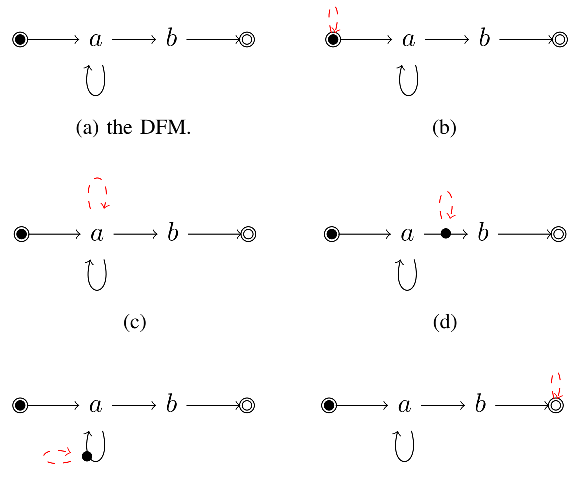
Log moves indicate that an event appeared in the log while it was not allowed at that point by the model. We identified five positions on which a log move can occur, and we illustrate them using the following DFM as shown in Figure 4a.

(1) At the start of a trace, which we visualise by a loop on the start node. For instance, the trace $\langle l_c, a_s, a_c, b_s, b_c \rangle$ yields an alignment that is visualised in Figure 4b.

(2) During execution of a, which we visualise by a loop on a. For instance, the trace $\langle a_s, l_c, a_c, b_s, b_c \rangle$ yields an alignment that is visualised in Figure 4c.

(3) In between the executions of a and b, which we visualise by a loop on the edge from a to b. For instance, the trace $\langle a_s, a_c, l_c, b_s, b_c \rangle$ yields an alignment that is visualised in Figure 4d.

(4) In between two executions of a, which we visualise by a loop on the model's loop on a. For instance, the trace



(a) the DFM. (b) (c) (d) (e) (f)

Fig. 4. A DFM and log move positions on it.

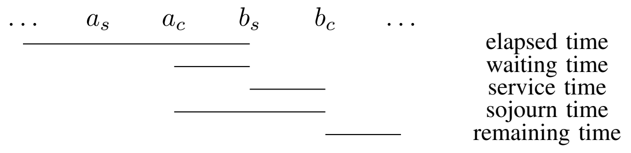
$\langle a_s, a_c, l_c, a_s, a_c, b_s, b_c \rangle$ yields an alignment that is visualised in Figure 4e.

(5) At the end of a trace, which we visualise by a loop on the end node. For instance, the trace $\langle a_s, a_c, b_s, b_c, l_c \rangle$ yields an alignment that is visualised in Figure 4f.

By construction, each model or log move corresponds to one of these types.

VI. DF-Performance Measures

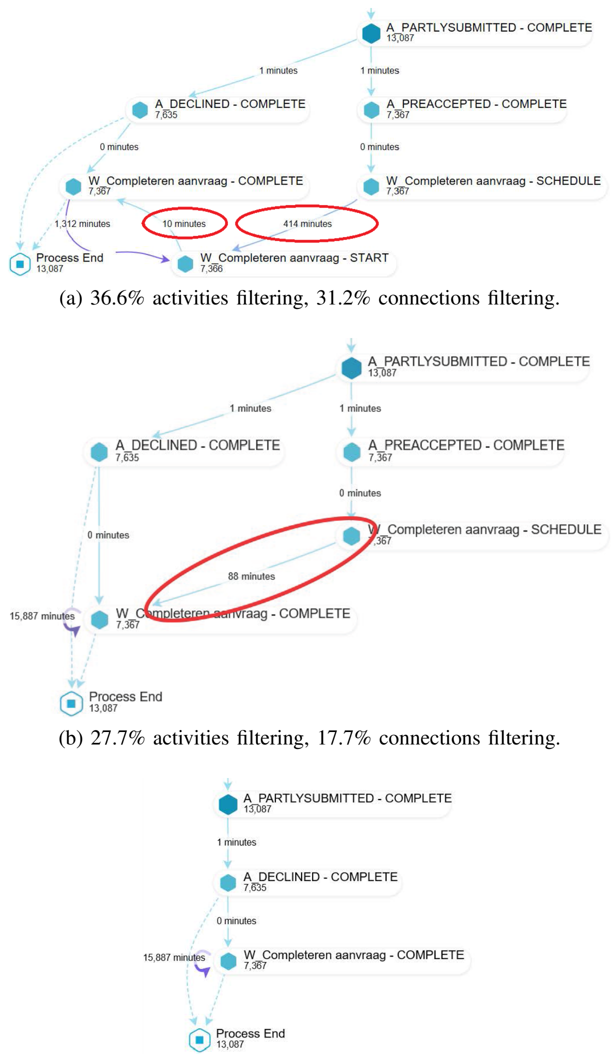
Another perspective that is often used in process mining is the performance perspective. DFMs enable the computation of most commonly used performance indicators, such as service time, waiting time and sojourn time [27]. In addition, the time elapsed and remaining in a case can be of interest:



Synchronisation time [1] is not defined on DFMs, as DFMs cannot represent concurrent activities.

Many process mining tools provide performance measures. However, measures can be unreliable without the use of sound models and alignments. That is, existing approaches, while removing edges from a directly follows-based model, also remove the time that was associated with that edge from the model. As a result of this, the times shown do not add up correctly to overall case durations and times seem to disappear.

For instance, consider the directly follows-based models shown in Figure 5: these models are derived from the same event log using different levels of filtering in CPE. In these models, consider the path taken through `W_Completeren aanvraag-schedule` via `-start` to `-complete`. In the least-filtered model (Figure 5a), this path takes more than 400 minutes. In a bit more filtered model (Figure 5b), this path takes 88 minutes, while in the most-filtered model (Figure 5c), it does not seem to take



A_PARTLYSUBMITTED - COMPLETE 13,087
1 minutes
1 minutes
A_DECLINED - COMPLETE 7,635
A_PREACCEPTED - COMPLETE 7,367
0 minutes
0 minutes
W_Completeren aanvraag - COMPLETE 7,367
W_Completeren aanvraag - SCHEDULE 7,367
1,312 minutes
10 minutes
414 minutes
Process End 13,087
W_Completeren aanvraag - START 7,366
88 minutes
15,887 minutes

(a) 36.6% activities filtering, 31.2% connections filtering.

(b) 27.7% activities filtering, 17.7% connections filtering.

(c) 22% activities filtering, 13.3% connections filtering.

Fig. 5. BPI Challenge 2012 log in CPE 4.4 at different levels of filtering.

any time. In fact, in the most-filtered model, seemingly the majority of cases takes only 1 minute, which could lead to the false conclusion that barely any time is spent in the process. A further consideration found similar issues in the other commercial process mining tools.

By using sound Petri nets and alignments to compute performance measures, such issues can be avoided. For instance, the time between `-schedule` and `-start` would be added to the time between `-schedule` and `-complete` if `-start` would be filtered out [27], [19], [20]. This highlights the importance of reliable performance measures *and conformance checking*, which could be of aid in signalling that the path `A_PARTLYSUBMITTED-complete` via `A_PREACCEPTED-complete` to `W_Completeren aanvraag-complete` has been removed from the model by filtering[4].

Directly Follows visual Miner: To convey the concepts of DFM discovery, conformance checking and performance measuring, these concepts have been implemented in a new

[4]CPE possesses conformance checking capabilities, but not on its DFM.

tool, the *Directly Follows visual Miner* (DFvM). Given an event log, DFvM first discovers a DFM, translates it to a Petri net, aligns this net, visualises deviations and animates the event log on the DFM, all without user interaction. Users can change parameters, filter the log, change the visualisation, etc., after which DFvM will recompute the necessary steps automatically, enabling a truly exploratory approach to process mining. Figure 7 shows DFvM, which is available as a plug-in of the ProM framework [28] and shares its code base with the Inductive visual Miner [20]. Additionally, the plug-in *Visualise deviations on Directly Follows Model* allows for easy comparison of a DFM model and an event log, that is, this plug-in entails DFvM without the discovery parts but with a given model.

VII. Case Study at the Queensland Government

One of the motivators for this approach was a case study that was performed with a major government organisation in Australia. The organisation relies on an IT service management system to automatically capture activities performed by operational teams. The case study focused on financial and human resources processes support by this system. Documentation for some of these processes existed and but for others only outdated or no models existed. During each of the calendar years, 2017 and 2018, the organisation instigated several initiatives to improve the quality of their service performance.

The organisation had three objectives: i) to have complete and up-to-date documentation of their processes, ii) to check whether processes are executed correctly and iii) to identify performance bottlenecks in their processes. To achieve these objectives we extracted an event log for one month of operations covering four areas of operations with up to 53 distinct workflows each. The log contained 142896 active cases of which 80883 were completed and used in the analysis.

To achieve the first objective, the new approach was used to mine direct-follows process models from the event log. As all of the processes are sequential and do not contain concurrent tasks, direct-follows models appropriately document the processes for which documentation was missing or outdated.

To achieve the second objective, DFMs were automatically constructed from Excel tables specifying the intended process as lists of direct-follows-relations between process activities. As existing approaches could not do conformance checking for DFMs, we used the new approach to check and annotate the constructed models with conformance information, as shown in Figure 6. This example model shows several unexpected transitions (red dashed lines) between activities which were missing in the documentation and proved that the existing documentation is not up-to-date.

The third objective was to identify performance bottlenecks in the analysed processes. Existing approaches produced performance data that was too complex without filtering and became inconsistent on filtering. Instead, we used our mined direct-follows models for the analysis. Figure 7 shows a model with performance annotations (sojourn times). Activities are

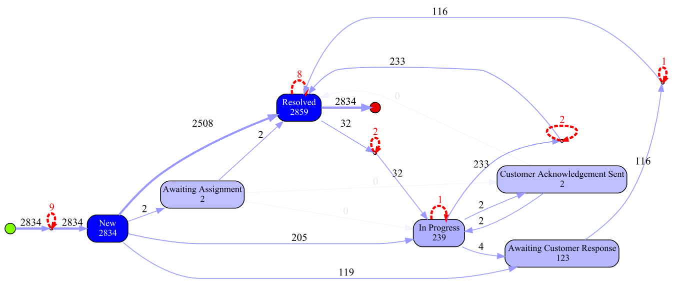


Fig. 6. DFM annotated with conformance and frequency information; labels are not intended to be readable.

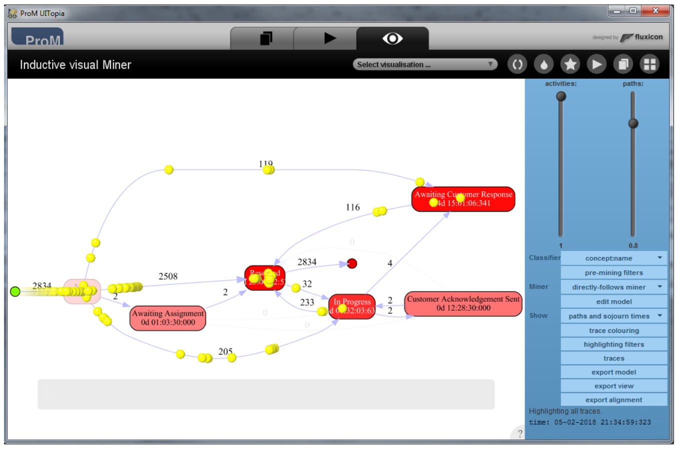


Fig. 7. Screenshot of the Directly Follows visual Miner (DFvM) in ProM, where animated tokens show cases flowing through the process. The DFM is annotated with performance information (sojourn times).

drawn in red to indicate durations. For example, cases involving interactions with external customers take significantly longer.

The stakeholders were also interested in comparing cases across various groupings such as request types, so annotated models were generated for each workflow and each of these groupings, resulting in a total of 4905 models. Data from these models was aggregated and some high-impact workflows were identified and analysed in more detail, and all the models were handed over to the organisation for internal documentation.

At the time of writing the organisation's business improvement team is incorporating the analysis results in their review of business process effectiveness and efficiency. When results were presented, directly-follows models highlighted deviations from documented process pathways. Further deviations became apparent to business improvement team members during the visual replays of some of the business process models using DFvM in ProM. Team members discussed likely explanations for such deviations including incorrect uses of the IT service management system, data quality issues and potential training issues. The business improvement team will use the conformance checking results to identify and rectify potential data processing errors and associated data quality issues. The results also highlighted high-level processes with long-running sub-processes. The business improvement team will review each of these process and its sub-processes for underlying causes. The business improvement team reviewed a sample of the new process documentation generated by process mining. The review highlighted that some process activities require additional effort to increase documentation.

VIII. Conclusion

While many process mining tools exist, many commercial tools lack conformance checking (quality of models cannot be verified), their performance measures might be unreliable, and academic tools lack the intuitiveness and simplicity of DFMs.

Our approach bridges this gap by showing how DFMs can be discovered from event logs, while guaranteeing soundness and a minimum fitness level chosen by the user. We introduced a DFM-to-Petri-net translation, which is used to align the model with a log. The alignment is then used to measure the quality of a discovered model, identify and visualise deviations, and provide reliable performance measures. Furthermore, we highlighted issues in performance measures of existing commercial process mining tools and showed that alignment-based measures do not suffer from the same issues.

We implemented discovery, conformance checking and performance measures based on DFMs in a process mining tool that enables exploring event logs by repeated discovery, conformance checking and filtering.

This tool was applied in a case study in the Queensland Government. In this study, we discovered DFMs from event logs, compared them to existing manually made DFMs and analysed the performance of processes; the outcomes of this study were used to document, review, improve and automate processes. Overall, we showed that conformance checking and reliable performance measures can be combined with directly follows-based models and that such models can provide guarantees such as soundness or a minimum fitness level. This tool demonstrates that conformance checking and transparent performance measures can be offered in typical process mining tools based on directly follows semantics. We hope that commercial parties will adopt these ideas to enable their customers to assess the quality of discovered models and to better understand performance measures.

In future work, it would be interesting to evaluate how our discovery technique compares to existing discovery techniques that support advanced constructs. Furthermore, discovered DFMs might be used as input to these techniques [29]. Furthermore, advanced infrequent behaviour filtering techniques might improve results or provide different insights [25]. Finally, our conformance checking concepts might be used to perform conformance checking on hybrid models [2].

Acknowledgement: We gratefully acknowledge funding and support from the Queensland Government during the conduct of this research.

References

[1] W. M. P. van der Aalst, *Process Mining - Data Science in Action, Second Edition*. Springer, 2016.

[2] W. M. P. van der Aalst, R. De Masellis, C. D. Francescomarino, and C. Ghidini, “Learning hybrid process models from events - process discovery without faking confidence,” in *BPM*, 2017, pp. 59–76.

[3] S. J. J. Leemans, D. Fahland, and W. M. P. van der Aalst, “Discovering block-structured process models from event logs containing infrequent behaviour,” in *BPM workshops*, 2013, pp. 66–78.

[4] A. Augusto, R. Conforti, M. Dumas, and M. L. Rosa, “Split miner: Discovering accurate and simple business process models from event logs,” in *IEEE ICDM*, 2017, pp. 1–10.

[5] W. Reisig, *A primer in Petri net design*, ser. SCI. Springer, 1992.

[6] O. M. G. (OMG), “Business process model and notation (BPMN) version 2.0,” Object Management Group (OMG), Tech. Rep., jan 2011.

[7] M. De Leoni, F. M. Maggi, and W. M. P. van der Aalst, “Aligning event logs and declarative process models for conformance checking,” in *BPM*. Springer, 2012, pp. 82–97.

[8] A. Weijters, W. van der Aalst, and A. de Medeiros, “Process mining with the heuristics miner-algorithm,” Eindhoven University of Technology, BETA Working Paper series 166, 2006.

[9] C. W. Günther and W. M. P. van der Aalst, “Fuzzy mining - adaptive process simplification based on multi-perspective metrics,” in *BPM*, 2007, pp. 328–343.

[10] R. Conforti, M. Dumas, L. García-Bañuelos, and M. L. Rosa, “BPMN miner: Automated discovery of BPMN process models with hierarchical structure,” *Inf. Syst.*, vol. 56, pp. 284–303, 2016.

[11] S. K. L. M. vanden Broucke and J. D. Weerdt, “Fodina: A robust and flexible heuristic process discovery technique,” *Decision Support Systems*, vol. 100, pp. 109–118, 2017.

[12] J. C. A. M. Buijs, B. F. van Dongen, and W. M. P. van der Aalst, “A genetic algorithm for discovering process trees,” in *IEEE CEC*, 2012, pp. 1–8.

[13] V. Liesaputra, S. Yongchareon, and S. Chaisiri, “Efficient process model discovery using maximal pattern mining,” in *BPM*, 2015, pp. 441–456.

[14] A. Ehrenfeucht and G. Rozenberg, “Partial (set) 2-structures. part I: basic notions and the representation problem,” *Acta Inf.*, vol. 27, no. 4, pp. 315–342, 1990.

[15] C. W. Günther and A. Rozinat, “Disco: Discover your processes,” in *BPM demos*, 2012, pp. 40–44.

[16] V. Leno, A. Armas-Cervantes, M. Dumas, M. L. Rosa, and F. M. Maggi, “Discovering process maps from event streams,” in *ICSSP 2018*, 2018, pp. 86–95.

[17] W. M. P. van der Aalst, A. Adriansyah, and B. F. van Dongen, “Replaying history on process models for conformance checking and performance analysis,” *WIRDMKD*, vol. 2, no. 2, pp. 182–192, 2012.

[18] M. T. Wynn, E. Poppe, J. Xu, A. H. M. ter Hofstede, R. Brown, A. Pini, and W. M. P. van der Aalst, “Processprofiler3d: A visualisation framework for log-based process performance comparison,” *Decision Support Systems*, vol. 100, pp. 93–108, 2017.

[19] A. Adriansyah, “Aligning Observed and Modeled Behavior,” Ph.D. dissertation, Eindhoven University of Technology, 2014.

[20] S. J. J. Leemans, D. Fahland, and W. M. P. van der Aalst, “Process and deviation exploration with Inductive visual Miner,” in *BPM Demo Sessions 2014*, 2014, p. 46.

[21] A. Partington, M. T. Wynn, S. Suriadi, C. Ouyang, and J. Karnon, “Process mining for clinical processes: A comparative analysis of four australian hospitals,” *ACM TMIS*, vol. 5, no. 4, pp. 19:1–19:18, 2015.

[22] W. M. P. van der Aalst, S. Guo, and P. Gorissen, “Comparative process mining in education: An approach based on process cubes,” in *DDPDA*. Springer, 2013, pp. 110–134.

[23] R. P. J. C. Bose, A. Gupta, D. Chander, A. Ramanath, and K. Dasgupta, “Opportunities for process improvement: A cross-clientele analysis of event data using process mining,” in *ICSOC*, 2015, pp. 444–460.

[24] S. J. J. Leemans, D. Fahland, and W. M. P. van der Aalst, “Using life cycle information in process discovery,” in *BPM Workshops*, 2015, pp. 204–217.

[25] M. F. Sani, S. J. van Zelst, and W. M. P. van der Aalst, “Improving process discovery results by filtering outliers using conditional behavioural probabilities,” in *BPM workshops*, 2017, pp. 216–229.

[26] A. Adriansyah, J. Munoz-Gama, J. Carmona, B. F. van Dongen, and W. M. P. van der Aalst, “Measuring precision of modeled behavior,” *Inf. Syst. E-Business Management*, vol. 13, no. 1, pp. 37–67, 2015.

[27] M. Leemans, W. M. P. van der Aalst, and M. G. J. van den Brand, “Hierarchical performance analysis for process mining,” in *ICSSP*, 2018, pp. 96–105.

[28] B. F. van Dongen, A. K. A. de Medeiros, H. M. W. Verbeek, A. J. M. M. Weijters, and W. M. P. van der Aalst, “The ProM framework: A new era in process mining tool support,” in *Petri Nets*, 2005, pp. 444–454.

[29] S. J. J. Leemans, D. Fahland, and W. M. P. van der Aalst, “Scalable process discovery and conformance checking,” *SoSyM*, vol. 17, no. 2, pp. 599–631, 2018.