Computing and Informatics, Vol. 40, 2021, 341–367, doi: 10.31577/cai_2021_2_341

WEAKLY COMPLETE EVENT LOGS IN PROCESS MINING

Julijana Lekić

Faculty of Technical Sciences, University of Pristina in Kosovska Mitrovica
Kneza Milosa 7, 38220 Kosovska Mitrovica, Serbia
e-mail: julijana.lekic@pr.ac.rs

Dragan Milićev

Faculty of Electrical Engineering, University of Belgrade
Bulevar kralja Aleksandra 73, 11000 Belgrade, Serbia
e-mail: dmilicev@etf.bg.ac.rs

Abstract. Many information systems have a possibility to record their execution, and, in this way, to generate a trace about events describing the real system behaviour. From behaviour example records in traces of the event log, the α-algorithm automatically generates a process model that belongs to a subclass of Petri nets, known as workflow nets. One of the basic limiting assumptions of α-algorithm is that the event log needs to be complete. As a result of attempting to overcome the problem of completeness of the event log, we introduced the notion of weakly complete event logs, from which our modified technique and algorithm can produce the same result as the α-algorithm from complete logs on parallel processes. Thereby weakly complete logs can be significantly smaller than complete logs, considering the number of traces they consist of. Weakly complete logs were used for the realization of our idea of interactive parallel business process model generation.

Keywords: Process mining, business process model discovery, block-structured parallel process models, α-algorithm, $\alpha^{\|}$-algorithm, complete log, weakly complete event log

1 INTRODUCTION

Extraction of knowledge from traces recorded in event logs which are available in today's information systems, and use of that knowledge for discovering, monitoring and improvement of bussiness proces models, are basis for occurrence of different techniques of process mining (PM) areas [1, 2, 3]. α-algorithm is able to discover a large class of workflow (WF) nets [4] based on the behaviour recorded in event logs, with the main limiting assumption that the event log is complete. The property of completeness of the log often implies the necessity of having a large number of traces in the log on which the "representative" model for the behaviour seen in the log has to be constructed. Therefore, our challenge was to find logs with potentially much lower number of traces, which may not be complete, but are sufficiently valid so that, using the appropriate algorithm based on the evidences recorded in such logs, a "representative" model can be obtained.

To achieve this, we have partially modified the technique of process model discovering, and also the α-algorithm itself [1, 2, 3] by introducing the relation of *indirect precedence* as another basic relation between the activities recorded in the event log [5, 6]. Within that goal we firstly defined the so called *causally complete* event logs, which do not fulfil conditions of completeness, and yet using our modified algorithm $\alpha^{||}$-algorithm) we can reconstruct the original network of a parallel bussiness process. Obtaining block-structured parallel business process model with our modified method based on the event logs which fulfil the requirements of casual completeness is presented in [5].

The discovery of the parallel bussiness process model from event logs with as few recorded traces as possible was a part of our research, and the ultimate goal was finding ways for those models to be created interactively. With defining of causally complete event logs that goal was not achieved in full. Namely, even though parallel bussiness process models could be successfully discovered from causally complete event logs based on significantly smaller number of traces than from complete event logs, still we could not achieve the interactive model creation. Further research in that direction leads to defining of the so called *weakly complete* event log by which our goals could be achieved. Weakly complete event logs may contain a significantly smaller number of traces needed for obtaining the appropriate parallel business process model than in case with complete or causally complete logs.

The property of $\alpha^{||}$-algorithm, where its application on the weakly complete event logs may lead to the model discovery, was succesfully used for the creation of the parallel business process models by demonstration. For this purpose, its own demonstration graphical user interface has been created, which enables user to perform different scenarios of activity execution process using direct manipulation. The graphical user interface we created is a tool that visually shows steps of $\alpha^{||}$-algorithm. Such tool could serve as a learning tool and playground for those who want to learn more about the general α-algorithm, which is based on the same principles like its modified version $\alpha^{||}$-algorithm, and how it works. This has been

achieved through the idea in which a user enters log entries (scenario) step by step and observes a current model which is shown and obtained based on the log entries entered.

The created demonstration user interface contains the components of artificial inteligence which are reflected in the fact that the system itself suggests the order of process activitiy performance (in order to discover the process model as soon as possible) and "infers" some relations between the actvities which were not performed during the demonstration procedure. In order to find and infer relations which were not performed (inferred relations), the system uses the event log footprint[1] and certain rules described later in this paper. The results and ways of realization of this idea are presented in our paper [7].

Besides defining weakly complete event logs, this paper will present the results of experimental analysis carried out on examples of real business processes with the goal to discover the size of weakly complete event logs for observed processes expressed in the number of traces. The obtained results will be compared to the results of the experimental analysis carried out for the same processes on complete and causally complete event logs, which is presented in detail in [5]. In that way we will be able to show the improvement we made on weakly complete event logs in comparison to causally complete event logs [5] in regard of obtaining the valid model of parallel business processes based on the least possible number of traces from the event logs.

Our assumptions and preconditions for process models (that have to be block-structured parallel models), to which our algorithm is applicable, may look as a serious restriction. It really is for real-world process models in general. However, our solution still covers a respectably wide subclass of process models and represents a first step in a more ambitious attempt to solve the very serious problem of log completeness. Although having a strong limitation, we still deem that even a partial solution to such a serious problem can be beneficial for future research and is worth reporting. In our future research, we will try to expand our work to other categories of processes.

The paper is organized as follows. The next section provides an overview of the existing literature from the area of interest for our work and this paper. Section 3 gives some preliminaries about modified PM technique, modified $\alpha^{||}$-algorithm and weakly complete event logs necessary for someone to further follow the content of this paper. The problem with dangling nods in WF-net that is encountered during the discovery of the business process model from weakly complete event logs as well as the way of resolving that problem are presented. An example of the $\alpha^{||}$-algorithm application on the weakly complete event log is presented at the end of this section. Section 4 describes the ProM framework for applying the $\alpha^{||}$-algorithm on weakly complete event logs, within which the plug-ins we have developed for the needs of the

[1] Footprint of the event log is a matrix in which the defined relations between any two activities can be represented [2, 3, 5].

experimental analysis are presented. The results of the application of α-algorithm and $\alpha^{||}$-algorithm on weakly complete log which is used as a running example in the paper are also shown at the end of the section. In Section 5 we present the results of the performed experimental analysis. Section 6 contains some conclusions and guidelines for the future work and research.

2 RELATED WORK

The process mining idea is not a new one. One of the earliest examples of usage of PM in the context of workflow management, based on workflow graphs, was presented in [8]. Cook and Wolf have investigated similar issues in the context of software engineering, looking in particular sequential [9] and parallel behaviour of the process [10]. Dealing with sequential processes, in [9] they describe three methods for detection, one of which uses a neural network, while the other one is entirely based on the algorithmic approach, and the third one uses a Markovian approach. In [10], the same authors extended their work to concurrent processes, where they propose specific metrics to discover models out of event streams, but they do not provide an approach to generate explicit process models. Herbst also dealt with the No. of PM in the context of workflow management [11, 12] using an inductive approach, observing sequential [12] and parallel models [11] separately. A notable difference between his work and other approaches is that the same task can appear multiple times in the workflow model, i.e., the approach allows for duplicate tasks. Schimm [13] has developed a mining tool suitable for discovering hierarchically structured workflow processes.

Although many researchers have dealt with the ideas of process mining, the most comprehensive study is presented in the works of W. M. P. van der Aalst and his collaborators [1, 2, 3, 14, 15, 16, 17]. In [2, 3] a detailed description and formalization of techniques for discovering processes from workflow logs is presented, the α-algorithm for extracting process models from such logs is defined, and a representation of the model obtained in the form of a *sound*[2] WF-net is shown. An introduction to PM and an overview of existing techniques for discovering processes, and the problems which have been encountered in the application of the α-algorithm have been most fully presented in [1]. An overview of best practices and challenges is presented in [18], which is the work of a group of experts that was created in order to promote research, development and understanding of process mining as well as its implementation and evolution.

To discover process models from traces recorded in workflow logs, many techniques have been proposed [2, 3, 8, 9, 16, 19]. Many of these techniques use Petri nets in the process of discovering and presenting the discovered process model. However, other very different approaches are also used for the same purpose.

[2] Soundness corresponds to liveness and safeness of the corresponding short-circuited net [2, 3]. The set of all sound WF-nets (SWF) is denoted with `W`.

Although the original α-algorithm is able to discover a large class of WF-nets, there are problems it cannot cope with: incompleteness of the event log, rare behaviours, complex routing constructions and others [1]. As a consequence, there is a large number of algorithms that overcome lacks of the basic α-algorithm [20]. Some of them are variants of the original α-algorithm, such as, for example, the α^+ algorithm [15] and α^{++} algorithm [17], while others use a completely different approach, such as: heuristic mining [9], genetic mining [16], fuzzy mining [21], process mining from a basis of regions [19] or flexible heuristics miner [22]. The inductive miner (IM) [23], aims to discover block-structured process models that are sound and fitting to the behavior represented in the event log.

The above mentioned algorithms and techniques have overcome some of problems, but the problem of incompleteness of logs, to the best of our knowledge, has not been overcome. This fact still remains a challenge for future research.

The subject of our research presented in this paper is the problem of completeness of event logs [1]. Part of this research was preliminarily announced in our previous paper [24], and has been presented in [5], where we also dealt with logs that do not meet the requirement of completeness. In this paper, the more detailed description of weakly complete log is given than in [24], based on the definitions and rules. Moreover, the comparison of results obtained by applying our algorithm [5] with the results of other process mining algorithms on weakly complete logs was done. Finally, another big difference from the paper [24] is that this paper brings an extensive experimental analysis and its results are presented here. Unlike [5] where we dealt with causally complete event logs, this paper focuses on weakly complete event logs, that we obtained in the attempt to improve the characteristics of causally complete event logs. One of those improved characteristics is a smaller size of event logs, which was a basic parameter for the experimental analysis that was performed. In this analysis, values of the minimal sizes of complete logs, causaly complete logs and weakly complete logs are compared for 100 real examples of parallel business processes. In this paper, plug-ins are also presented in the existing ProM framework designed for the needs of the experimental analysis.

Another improvement that we have achieved with weakly complete event logs compared to causally complete event logs is the property of weakly complete event logs that based on them, the models of parallel business processes can be interactively (using demonstration) generated, which is presented in detail in our paper [7].

3 PRELIMINARIES

In this section, we give some definitions of concepts used throughout this paper. We must note that the detailed description of the concepts that will be mentioned, as well as how they were obtained, is shown in our paper [5], therefore we will here provide only the short review necessary for someone to further follow the content of this paper.

3.1 Modified Technique and Algorithm for Discovering Process Models

For the purposes of this paper, we will show only the basic characteristics of our modified PM technique and modified α-algorithm.

The log-based relations that are used to indicate the relevant patterns in the log in our modified technique of discovering process models are defined by Definition 1.

Definition 1. (Log-based ordering relations, in the modified PM technique of discovering process models). Let L be an event log over $\mathcal{A}$[3], i.e., $L \in \mathcal{P}\ (\mathcal{A}^*)$[4]. Let $a, b \in \mathcal{A}$ be two activities. Then, by definition:

- $a >_L b$ if and only if there is a trace $\sigma = \{t_1, t_2, t_3, \ldots, t_n\}$ and $i \in \{1, \ldots, n-1\}$ such that $\sigma \in L$ and $t_i = a$ and $t_{i+1} = b$,
- $a \gg_L b$ if and only if there is a trace $\sigma = \{t_1, t_2, t_3, \ldots, t_n\}$ and there are $i, j \in \{1, \ldots, n\}$ such that $i + 2 \leq j$, where $\sigma \in L$ and $t_i = a, t_j = b$, and it is not that $a >_L b$,
- $a \rightarrow_L b$ if and only if $a >_L b$, and it is not $b >_L a$, and it is not $b \gg_L a$,
- $a \Rightarrow_L b$ if and only if $a \gg_L b$, and it is not $b >_L a$, and it is not $b \gg_L a$,
- $a \sharp_L b$ if and only if it is not $a >_L b$, and it is not $b >_L a$, and it is not $a \gg_L b$, and it is not $b \gg_L a$,
- $a ||_L b$ if and only if $a >_L b$ and $b >_L a$, or $a >_L b$ and $b \gg_L a$, or $a \gg_L b$ and $b >_L a$, or $a \gg_L b$ and $b \gg_L a$.

The defined relations can be represented with a matrix, which represents a footprint of the event log.

The $\alpha^{||}$-algorithm, where "||" reflects the fact that the algorithm targets parallel business processes, can be described by Definition 2.

Definition 2. ($\alpha^{||}$-algorithm). Let L be an event log over $T \subseteq \mathcal{A}$. $\alpha^{||}(L)$ is defined as follows:

1. $T_L = \{t \in T \mid (\exists \sigma \in L)\ t \in \sigma\}$,
2. $T_I = \{t \in T \mid (\exists \sigma \in L)\ t = \mathit{first}(\sigma)\}$,
3. $T_O = \{t \in T \mid (\exists \sigma \in L)\ t = \mathit{last}(\sigma)\}$,
4. $X_L = \{(A, B) \mid A \subseteq T_L \wedge A \neq \emptyset \wedge B \subseteq T_L \wedge B \neq \emptyset \wedge (\forall a \in A)(\forall b \in B)(a \rightarrow_L b)\}$,
5. $P_L = \{p_{(A,B)} \mid (A, B) \in X_L\} \cup \{i_L, o_L\}$,

[3] $\mathcal{A}$ is the set of business process activities.

[4] $\mathcal{A}^*$ is the set of all finite sequences (traces) of the elements of $\mathcal{A}$, and $\mathcal{P}(\mathcal{A}^*)$ is the powerset of $\mathcal{A}^*$.

6. $F_L = \{(a, p_{(A,B)}) \mid (A, B) \in X_L \wedge a \in A\} \cup \{(p_{(A,B)}, b) \mid (A, B) \in X_L \wedge b \in B\} \cup \{(i_L, t) \mid t \in T_I\} \cup \{(t, o_L) \mid t \in T_O\}$,
7. $\alpha^{||}(L) = (P_L, T_L, F_L)$.

A necessary condition for discovering the original network by the α-algorithm is that the log on which the algorithm is applied needs to be complete, where the condition of completeness is based on the relation $>_L$ [1, 2, 3]. The condition of completeness in our modified PM technique for model discovering is related to the causality relation $\rightarrow_L$.

For a particular process model to be discovered, there may be a large (in general, an unlimited) number of different complete logs. However, all these complete logs have the same footprint, i.e., the same causality relation. We call this relation the *basic causality relation.*

Definition 3. (The basic causality relation). Let $N = (P, T, F)$[5] be a sound WF-net, i.e., $N \in \mathcal{W}$, and let L be a complete workflow log of N. $\rightarrow^B{}_N$ is the basic causality relation of network N iff $\rightarrow^B{}_N = \rightarrow_L$.

On the other hand, there may exist other logs for the same process model that are not complete or causally complete [5], but which have the same causality relation obtained from those logs. We are focused on investigating such logs, which we refer to as *weakly complete logs.* Obviously, the idea is to find weakly complete logs that may be, in general, significantly smaller than fully complete logs (in the terminology of the original α-algorithm) and causally complete logs (in the terminology of the modified $\alpha^{||}$-algorithm).

3.2 Weakly Complete Event Logs

As it has already been shown in [5], by using the modified PM technique on parallel processes we can obtain the original network from any event log in which $\rightarrow_L = \rightarrow^B{}_N$. Therefore, the main task is to find a log with the causality relation that is equal to the basic causality relation, and then apply the $\alpha^{||}$ algorithm, which leads to the original network of the parallel processes. There are event logs in which the causality relation is not equal to $\rightarrow^B{}_N$, but from the footprint of that log we can subsequently conclude the elements of the causality relation on whose joining the causality relation of the log becomes equal to $\rightarrow^B{}_N$. The log with such property has been named *weakly complete event log – L_w.*

Examples we have analyzed show that the original network can be discovered from an incomplete log L for which the following holds: $\rightarrow^B{}_N \subset (\rightarrow_L \cup \Rightarrow_L)$, and $\rightarrow_L \subset \rightarrow^B{}_N$; if the causality relation of that log is joined by the elements of causality relation which can be inferred from the footprint of the log. Such a log we call a weakly complete log (L_w) [24].

[5] Tuple (P, T, F) originates from the Place/Transition net [2, 3] definition.

Definition 4. (Weakly complete event log). Let $N = (P, T, F)$ be a sound WF-net, i.e., $N \in \mathcal{W}$, and let $\rightarrow^B{}_N$ be the basic causality relation of N. L_w is a *weakly complete* workflow log of N iff:

1. $\rightarrow^B{}_N \subset (\rightarrow_{L_w} \cup \Rightarrow_{L_w})$, and $\rightarrow_{L_w} \subset \rightarrow^B{}_N$, and
2. for any $t \in T$ [6] there is $\sigma \in L_w$ so that $t \in \sigma$.

Even though in weakly complete event logs it cannot be inferred based on log traces that $\rightarrow_{L_w} = \rightarrow^B{}_N$ (but $\rightarrow_{L_w} \subset \rightarrow^B{}_N$), the elements of the causality relation which are not in $\rightarrow_{L_w}$, but are in $\rightarrow^B{}_N$, may be subsequently inferred from the log footprint. Those elements create causality relation called the *inferred causality relation*, denoted with $\rightarrow^i$. The causality relation that inserted the final appearance of the log footprint (denoted with $\rightarrow_{Lf}$), and on which the $\alpha^{||}$-algorithm is applied, becomes: $\rightarrow_{Lf} = \rightarrow_{Lw} \cup \rightarrow^i$ by which we get that $\rightarrow_{Lf} = \rightarrow^B{}_N$. Finding of the *footprint causality relation* $\rightarrow_{Lf}$ which equals to basic causal relation $\rightarrow^B{}_N$ is the condition enough for discovering the original network based on $\rightarrow_{Lf}$, using the $\alpha^{||}$-algorithm in a manner shown in [24].

3.3 The Dangling Nodes Problem

A network obtained from a weakly complete log often contains dangling nodes, i.e., activities (node in Petri net) without predecessors and/or successors. The occurrence of dangling nods in the network obtained based on the traces recorded in the weakly complete event log is due to a large number of activities that can be performed simultaneously and due to the rapid detection of parallelism by modified PM technique. Besides that, the occurrence of dangling nods in the network is also due to the property of weakly complete logs that all the causality relation elements cannot be discovered from the record written in the log traces, alone.

Definition 5. (Dangling node). Let $N = (P, T, F)$ be a network with initial place i and ending place o, and let a be such an activity that $a \in T$ and $a \notin \{i, o\}$. Activity $a \in T$ is a *dangling* node in network N if there is no activity $b \in T$ such that $a\bullet$ [7] $\cap \bullet b \neq \emptyset$ or there is no activity $b \in T$ such that $b \bullet \cap \bullet a \neq \emptyset$.

If activity $a \in T$ is a dangling node in network N and if there is no activity $b \in T$, such that $a \bullet \cap \bullet b \neq \emptyset$, then it can be said that activity a does not have its *successor* in network N. If activity $a \in T$ is a dangling node in network N and there is no activity $b \in T$ such that $b \bullet \cap \bullet a \neq \emptyset$, than it can be said that activity a does not have its predecessor in network N.

[6] $T \subseteq \mathcal{A}$, $\mathcal{A}$ is a set of activities, T is a finite set of transitions in the Petri net [4], σ is a trace such that $\sigma \in L$ [2, 3].

[7] The concepts of marking and tokens $\bullet$ are well known for Petri nets and precise definitions can be found in [2, 3].

The definition of a WF-net [4], includes an assumption of network connectivity, which means connectivity of all nodes in the network, and which prohibits the existence of dangling nodes. Therefore, the network obtained based only on the records written in weakly complete event log which contains dangling nods in itself is not a proper SWF-net, and it is not equal to the original network. To overcome the problem of dangling nodes, we observed the relations in footprints, and based on those we have defined the rules of inference of the direct successors and predecessors from the indirect ones. Thus, for each activity that is a dangling node, a successor and/or predecessor can be found.

Network obtained based on weakly complete event log which contains dangling nods has in the event log footprint at least one activity that in its table row does not have relation $\rightarrow$ (if the activity has no successor) or relation $\leftarrow$ (if the activity has no predecessor) [24].

Rule 1 (Determining the inferred causality relation $\rightarrow^i$ when activity has no successor). Let a be activity which in its log footprint row has no relation $\rightarrow$ then by definition: $a \rightarrow^i c$ iff in footprint $a \Rightarrow c$, and there is b such that $b \rightarrow c$, where $a||b$.

Rule 2 (Determining the inferred causality relation $\leftarrow^i$ when activity has no predecessor). Let a be activity which in its log footprint row has no relation $\leftarrow$, then by definition: $a \leftarrow^i c$ iff in footprint $a \Leftarrow c$ and there is b such that $b \leftarrow c$, where $a||b$.

Using Rule 1 and Rule 2, the elements of the inferred causality relation $\rightarrow^i$ ($\leftarrow^i$) can be determined based on the footprint of the event log. This contributes to the discovery of those causality relations which cannot be discovered from the weakly complete event log traces alone. The causality relation which enters the final log footprint appearance and on which $\alpha^{||}$-algorithm applies becomes: $\rightarrow_{L_f} = \rightarrow_{L_w} \cup \rightarrow^i$. In that way $\rightarrow_{L_f}$ becomes equal to the basic causality relation, i.e. $\rightarrow_{L_f} = \rightarrow^B{}_N$, which allows the discovery of the original network. The larger the number of elements $\rightarrow^i$ in relation $\rightarrow_{L_f}$, the poorer the weakly complete event log on basis of which the model can be constructed, i.e., with a smaller number of the traces recorded.

3.4 An Example of the $\alpha^{||}$-Algorithm Application on the Weakly Complete Event Log

Let us consider a parallel process model shown in Figure 1, and a log L with records obtained after several executions of the process.

$$L = [\langle a, d, b, e, c, f, h, g, k\rangle^4, \langle d, e, a, c, b, f, g, h, k\rangle^3].$$

The basic causality relation for this example is:

$$\rightarrow^B{}_N = \{(a, b), (a, c), (d, e), (b, f), (c, f), (e, f), (f, g), (f, h), (g, k), (h, k)\}.$$

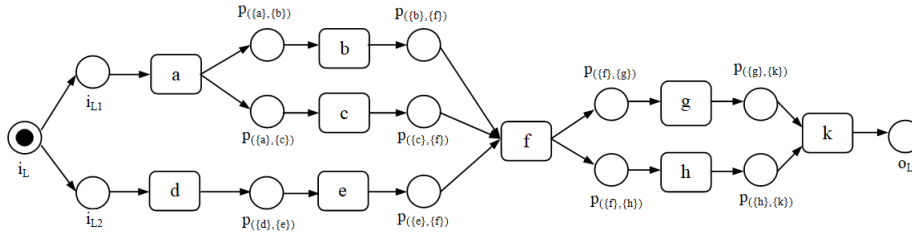

Figure 1. Example of a block-structured parallel process model

The log-based ordering relations for this example are:

$$>_L = \{(a,b),(b,c),(c,d),(d,e),(e,f),(f,g),(g,h),(h,k),(d,e),(e,a),(a,c),(c,b),(b,f),(f,h),(h,g),(g,k)\},$$

$$<_L = \{(b,a),(c,b),(d,c),(e,d),(f,e),(g,f),(h,g),(k,h),(e,d),(a,e),(c,a),(b,c),(f,b),(h,f),(g,h),(k,g)\},$$

$$\gg_L = \{(a,d),(a,e),(a,f),(a,g),(a,h),(a,k),(b,d),(b,e),(b,g),(b,h),(b,k),(c,e),(c,f),(c,g),(c,h),(c,k),(d,f),(d,g),(d,h),(d,k),(e,g),(e,h),(e,k),(f,k),(d,b),(d,f),(d,h),(d,g),(d,k),(e,c),(e,b),(e,h),(e,g),(e,k),(a,f),(a,h),(a,g),(a,k),(d,a),(d,c),(c,f),(c,h),(c,g),(c,k),(b,h),(b,g),(b,k),(f,k)\},$$

$$\ll_L = \{(d,a),(e,a),(f,a),(g,a),(h,a),(k,a),(d,b),(e,b),(g,b),(h,b),(k,b),(e,c),(f,c),(g,c),(h,c),(k,c),(f,d),(g,d),(h,d),(k,d),(g,e),(h,e),(k,e),(k,f),(a,d),(c,d),(b,d),(f,d),(h,d),(g,d),(k,d),(c,e),(b,e),(h,e),(g,e),(k,e),(f,a),(h,a),(g,a),(k,a),(f,c),(h,c),(g,c),(k,c),(h,b),(g,b),(k,b),(k,f)\},$$

$$||_L = \{(b,c),(c,b),(c,d),(d,c),(g,h),(h,g),(e,a),(a,e),(a,d),(d,a),(b,d),(d,b),(b,e),(e,b),(c,e),(e,c)\},$$

$$\rightarrow_L = \{(a,b),(d,e),(e,f),(f,g),(h,k),(a,c),(b,f),(f,h),(g,k)\},$$

$$\leftarrow_L = \{(b,a),(e,d),(f,e),(g,f),(k,h),(c,a),(f,b),(h,f),(k,g)\},$$

$$\Rightarrow_L = \{(a,f),(a,g),(a,h),(a,k),(b,g),(b,h),(b,k),(c,f),(c,g),(c,h),(c,k),$$
$$(d,f),(d,g),(d,h),(d,k),(e,g),(e,h),(e,k),(f,k)\},$$

$$\Leftarrow_L = \{(f,a),(g,a),(h,a),(k,a),(g,b),(h,b),(k,b),(f,c),(g,c),(h,c),(k,c),$$
$$(f,d),(g,d),(h,d),(k,d),(g,e),(h,e),(k,e),(k,f)\},$$

$$\sharp_L = \{(a,a),(b,b),(c,c),(d,d),(e,e),(f,f),(g,g),(h,h),(k,k)\}.$$

The footprint of the event log L is given in Table 1.

	a	b	c	d	e	f	g	h	k
a	$\sharp$	$\rightarrow$	$\rightarrow$	$\|\|$	$\|\|$	$\Rightarrow$	$\Rightarrow$	$\Rightarrow$	$\Rightarrow$
b	$\leftarrow$	$\sharp$	$\|\|$	$\|\|$	$\|\|$	$\rightarrow$	$\Rightarrow$	$\Rightarrow$	$\Rightarrow$
c	$\leftarrow$	$\|\|$	$\sharp$	$\|\|$	$\|\|$	$\Rightarrow$	$\Rightarrow$	$\Rightarrow$	$\Rightarrow$
d	$\|\|$	$\|\|$	$\|\|$	$\sharp$	$\rightarrow$	$\Rightarrow$	$\Rightarrow$	$\Rightarrow$	$\Rightarrow$
e	$\|\|$	$\|\|$	$\|\|$	$\leftarrow$	$\sharp$	$\rightarrow$	$\Rightarrow$	$\Rightarrow$	$\Rightarrow$
f	$\Leftarrow$	$\leftarrow$	$\Leftarrow$	$\Leftarrow$	$\leftarrow$	$\sharp$	$\rightarrow$	$\rightarrow$	$\Rightarrow$
g	$\Leftarrow$	$\Leftarrow$	$\Leftarrow$	$\Leftarrow$	$\Leftarrow$	$\leftarrow$	$\sharp$	$\|\|$	$\rightarrow$
h	$\Leftarrow$	$\Leftarrow$	$\Leftarrow$	$\Leftarrow$	$\Leftarrow$	$\leftarrow$	$\|\|$	$\sharp$	$\rightarrow$
k	$\Leftarrow$	$\Leftarrow$	$\Leftarrow$	$\Leftarrow$	$\Leftarrow$	$\Leftarrow$	$\leftarrow$	$\leftarrow$	$\sharp$

Table 1. Footprint of the log L

It can be noted that the causality relation of log L is not equal to the basic causality relation, i.e., $\rightarrow_L \neq \rightarrow^B{}_N$, but it holds: $\rightarrow^B{}_N \subset (\rightarrow_L \cup \Rightarrow_L), \rightarrow_L \cup \rightarrow^B{}_N$, which makes L a weakly complete log. It can also be seen from the footprints that the activity c does not have its direct successors (the rows c do not have $\rightarrow$), which means that there will be dangling nodes in the network.

According to the Rule 1, i.e., the rule of inference of direct from indirect successors in networks with dangling nodes, we obtain:

$$c \Rightarrow_L f, \quad b \rightarrow_L f, \quad c \|_L b$$

then

$$c \rightarrow^i{}_L f,$$

or

$$c \rightarrow_L f, \quad e \rightarrow_L f, \quad c \|_L e$$

then

$$c \rightarrow^i{}_L f$$

i.e.,

$$f \leftarrow^i{}_L c.$$

Thus: $\leftarrow^{i}{}_{L} = (c, f)$, i.e.:

$$\rightarrow_{L_f} = \rightarrow_L \cup \rightarrow^{i}{}_{L} = \{(a,b),(a,c),(d,e),(b,f),(c,f),(e,f),(f,g),(f,h),$$
$$(g,k),(h,k)\}.$$

It can be noted that now it holds $\Rightarrow_{Lf} = \rightarrow^{B}{}_{N}$. By applying the $\alpha^{||}$-algorithm to the given log L, we obtain the following:

1. $T_L = \{a, b, c, d, e, f, g, h, k\}$,
2. $T_I = \{a, d\}$,
3. $T_O = k$,
4. $X_L = \{(\{a\},\{b\}),(\{a\},\{c\}),(\{d\},\{e\}),(\{b\},\{f\}),(\{c\},\{f\}),(\{e\},\{f\}),(\{f\},\{g\}),(\{f\},\{h\}),(\{g\},\{k\}),(\{h\},\{k\})\}$,
5. $P_L = \{p(\{a\},\{b\}),p(\{a\},\{c\}),p(\{d\},\{e),p(\{b\},\{f\}),p(\{c\},\{f\}),p(\{e\},\{f\}),p(\{f\},\{g\}),p(\{f\},\{h\}),p(\{g\},\{k\}),p(\{h\},\{k\}),i_{L1},i_{L2},o_L\}$,
6. $F_L = \{(a,p_{(\{a\},\{b\})}),(p_{(\{a\},\{b\})},b),(a,p_{(\{a\},\{c\})}),(p_{(\{a\},\{c\})},c),(d,p_{(\{d\},\{e\})}),(p_{(\{d\},\{e\})},e),(b,p_{(\{b\},\{f\})}),(p_{(\{b\},\{f\})},f),(c,p_{(\{c\},\{f\})}),(p_{(\{c\},\{f\})},f),(e,p_{(\{e\},\{f\})}),(p_{(\{e\},\{f\})},f),(f,p_{(\{f\},\{g\})}),(p_{(\{f\},\{g\})},g),(f,p_{(\{f\},\{h\})}),(p_{(\{f\},\{h\})},h),(g,p_{(\{g\},\{k\})}),(p_{(\{g\},\{k\})},k),(h,p_{(\{h\},\{k\})}),(p_{(\{h\},\{k\})},k),(i_{L1},a),(i_{L2},d),(k,o_L)\}$,
7. $\alpha^{||}(L) = (P_L, T_L, F_L)$.

4 PROM FRAMEWORK FOR APPLYING THE $\alpha^{||}$-ALGORITHM

For the need of discovering original networks of block-structured parallel business processes by the modified PM method and the $\alpha^{||}$-algorithm based on causally complete [5] and weakly complete logs, we have developed a plug-in *Alpha||-algorithm* (Figure 5) for the existing ProM framework [26]. The program code of the plug-in is located in a separate, new package, *alpha_parallel_algorithm* and is located at address [27].

At the same address there is a program code of the *Alpha||-algorithm – helper plug-in* (Figure 5), which is given in a separate package *alpha_parallel_algorithm_basic_causal_relation*. The mentioned plug-in is created for the purpose of extraction of basic causal relations from a complete event log.

It should be noted that this utility of extracting the basic causality relation is not used by the $\alpha^{||}$-algorithm. As it has been explained, the point is that our algorithm is guaranteed to discover the original process model if the input log is causally or weakly complete, just as the original α-algorithm is guaranteed to restore the original model if the log is fully complete; on the opposite, none of these algorithms can guarantee that the obtained model is the original one if the log is not causally, weakly or fully complete, respectively. The plug-in is just an independent, helper utility that can be used during experimentation to check whether the given log

is weakly complete or not, for the given known process model. Of course, in the procedure of process discovery, the model is unknown.

In the procedure of discovering original networks from weakly complete event logs performance of additional operations by Rules 1 and 2 is needed, which allows inferring of direct successors based on indirect ones or predecessors respectively, in network with dangling nods. For each indirect relation a check is being made to determine if there is such an event in the set of events which meets any of the conditions from the above mentioned rules. If any such event is found the newly discovered causality relation is added to the *inferred causal_relations* collection.

Inferring direct successors based on indirect successors is implemented in the *weakly_completed_logs_find_casual_from_indirect_succesor* function, as shown in Appendix 1, which is located at address [27]. In addition, inferring of direct predecessors based on indirect predecessors is implemented in the *weakly_completed_logs_find_casual_from_indirect_predecessor* function as shown in [27, Appendix 1].

Figure 2 shows an example of a block-structured model of a parallel business process [5, 25], presented in a form of a Petri net, which represents our running example in this paper (as well as in [5] and [24]). Discovering the original model from Figure 2 from a weakly complete event log $L_w = [\langle a, b, c, d, e, f, g, h\rangle^3, \langle a, f, g, c, e, d, b, h\rangle^2]$ is presented in detail in [24].

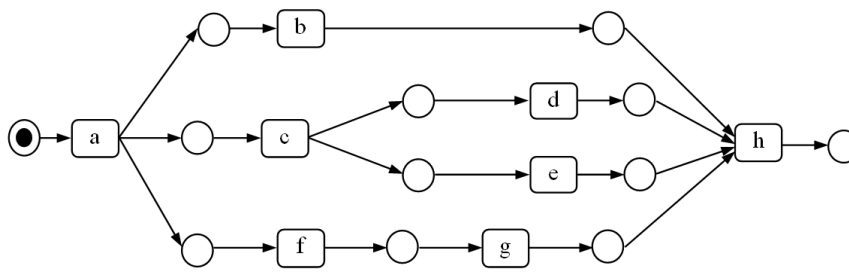

Figure 2. Example of a block-structured parallel process model

Figure 3 shows the $N^{||}$ [8] network obtained by applying the $\alpha^{||}$-algorithm over log $L_w = [\langle a, b, c, d, e, f, g, h\rangle^3, \langle a, f, g, c, e, d, b, h\rangle^2]$ and using the plug-in *Alpha||-algorithm*.

It can be noticed that the network $N^{||} = \alpha^{||}(L_w)$ in Figure 3 is equal to the original network in Figure 2, although it is obtained from a weakly complete log which is not complete and which is smaller than the complete log and causaly complete log, as it will be shown later in this paper.

[8] $N^{||} = (P, T, F)$ denotes a parallel process network [5].

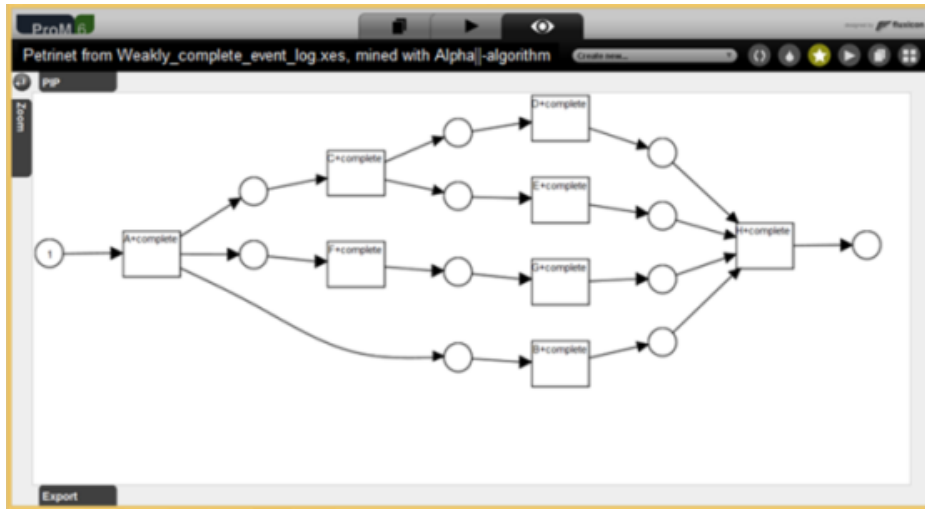

Figure 3. WF-net $N^{\|} = \alpha^{\|}(L_w)$

4.1 Comparison of Application of the $\alpha^{\|}$-Algorithm and Other Algorithms on Weakly Complete Event Logs

It can be seen from the above said that we deal with the problem of completeness which originates from the original α-algorithm and is present in all other its modifications or other algorithms for discovering process models. As other algorithms resulted mainly in the attempt to overcome some other problems, but not the problem of completeness, and as our algorithm was created by a modification of the basic α-algorithm, it would be most appropriate to compare it with the α-algorithm in the first place.

In order to evaluate the potential and effectiveness of our algorithm, we have applied several other algorithms to the same sample log $L_w = [\langle a, b, c, d, e, f, g, h\rangle^3, \langle a, f, g, c, e, d, b, h\rangle^2]$ and checked their ability to rediscover the original model. The sample log L_w is not complete, because there are missing elements in the relation $>_L$: $a > c, b > d, b > e, b > g, c > b, c > f, c > g, d > b, d > f, d > g, d > h, e > b, e > g, e > h, f > b, f > c, f > d, f > e, g > d$ and $g > e$, which could be potentially performed on the basis of the process model given in Figure 2, and the resulting WF-net $N^{\|}$.

When the original α-algorithm is applied on this weakly complete log L_w, the model shown in Figure 4 is obtained.

Due to the lack of elements of relations: $b > d$, $b > e$, $b > g$, $c > b$, $d > b$, $e > b$ and $f > b$, Alpha Miner was unable to detect that the activity b is parallel to the activities c, d, e, f and g. Due to the lack of elements of relations: $f > b$, $f > c$, $f > d$, $f > e$, $c > f$ and $d > f$, Alpha Miner was unable to detect that the activity f is parallel to the activities b, c, d and e. Due to the lack of elements of

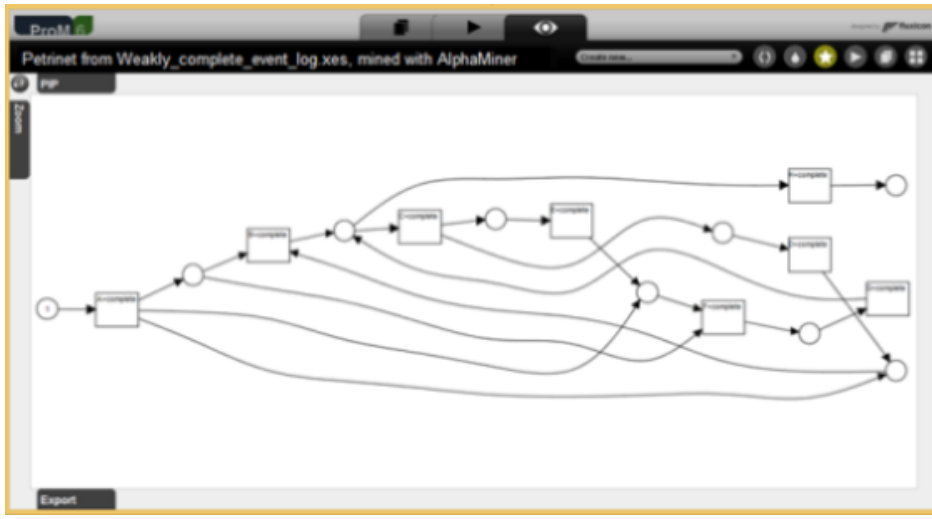

Figure 4. The result of the application of Alpha Miner to the weakly complete log L_w

relations: $c > b$, $c > f$, $c > g$ and $f > c$, Alpha Miner was unable to detect that the activity c is parallel to the activities b, f and g. Due to the lack of elements of relations: $b > g$, $c > g$, $d > g$, $e > g$, $g > d$ and $g > e$, Alpha Miner was unable to detect that the activity g is parallel to the activities b, c, d and e. For these reasons, the model obtained by Alpha Miner is so complex and different from the original network.

The Appendix 2 which is located at address [27] presents the results of the application of the available plug-ins for several other algorithms on the same given weakly complete log L_w: Alpha++ Miner, Heuristics Miner, Fuzzy Miner, Genetic Miner, ILP Miner, Mine transition system and Inductive Miner. As it can be seen from [27, Appendix 2], neither of the selected algorithms have succeeded to rediscover the original model from the given weakly complete log L_w. On the contrary, in most cases, the rediscovered models were rather complex and very far from the original model. We also give our opinion about the reasons of the inability to rediscover the original model for these algorithms.

5 EXPERIMENTAL ANALYSIS

Our experimental analysis was performed on real examples, where the size of minimal complete, minimal causaly complete and minimal weakly complete logs were compared. In order to achieve this within the existing ProM framework [21], another plug-in has been developed *Alpha‖-algorithm – minimal logs from complete log* which, from the given complete log extracts complete, causally complete and weakly complete logs with minimal possible number of traces, comparing their size

(Figure 5). The program code of the plug-in is located in a separate, new package, *alpha_parallel_algorithm_minimal_logs_from_complete_log*, and is located at address [27].

Figure 5. View of the ProM framework with an active plug-in Alpha||-algorithm – minimal logs from complete log

5.1 Procedure for Carrying Out Experimental Analysis

We will show the procedure that has been done in the experimental analysis in the example which model is shown in Figure 2. Let us observe the L event log with records obtained after several executions of the process in Figure 2.

$$\begin{aligned} L = [&\langle a,b,c,d,e,f,g,h\rangle, \langle a,f,g,b,c,e,d,h\rangle, \langle a,c,d,e,f,g,b,h\rangle,\\ &\langle a,b,f,g,c,d,e,h\rangle, \langle a,c,b,d,e,f,g,h\rangle, \langle a,c,b,e,d,f,g,h\rangle,\\ &\langle a,f,b,g,c,d,e,h\rangle, \langle a,c,f,b,g,e,d,h\rangle, \langle a,f,c,g,b,e,d,h\rangle,\\ &\langle a,f,g,c,d,b,e,h\rangle, \langle a,b,f,c,d,g,e,h\rangle, \langle a,c,e,b,d,f,g,h\rangle,\\ &\langle a,b,c,f,e,g,d,h\rangle, \langle a,c,f,d,g,e,b,h\rangle]. \end{aligned}$$

Only a variety of traces are displayed in the log, with no indication of the number of their occurrences, since the frequency of their occurrence is not relevant to this research. The log L has 14 traces and fulfils the conditions of completeness [2, 3] for the model in Figure 2.

When an active plug-in: Alpha||-algorithm – minimal logs from complete log (Figure 5), is started on the imported log L, we get the size and the appearance

of the minimal complete event log (has 14 traces), minimal causaly complete event log (has 6 traces) and minimal weakly complete event log (has two traces). The procedure shown is applied to all selected examples used in experimental analysis.

The experimental analysis was performed on a sample of 100 real examples obtained by arbitrary manual search of the Internet and selecting publicly available models of business processes, which fulfill our conditions of block-structured models of parallel processes[9]. The considered examples with their .xes files complete, causaly complete and weakly complete logs can be found at the address given in [27].

Some characteristics that reflect the network structure and size of the analysed examples are given in Tables 2 and 3 in [5]. These characteristics are expressed by the total number of activities in the network and a number of branches in the network. As with block-structured parallel processes there is one input and one output [5, 25] and it can be presented by the structure of the tree, by "branch" we meant a direct route from the entrance to the exit of the network.

5.2 Analysis Results

Table 2 presents results of the performed comparative analysis of the minimal size of complete, causaly complete and weakly complete logs, needed for discovering original networks of the considered examples.

For easier understanding of Tables 2 and 3 as well as Figures 6, 7 and 8, the following notations are used:

- N_{mcl} denotes the number of traces in minimal complete logs,
- N_{mccl} denotes the number of traces in minimal causaly complete logs,
- N_{mwcl} denotes the number of traces in minimal weakly complete logs,
- N_a denotes the total number of activities in parallel branches,
- N_b denotes the number of parallel branches in the network.

The performed experimental analysis has shown that the size of weakly complete logs from which the original networks of the observed parallel business processes can be discovered are lower, or (in the worst case) equal to the size of complete and causaly complete logs.

From Table 2 it can be seen that in 99 examples (from the examined 100 examples), the size of the minimal weakly complete logs is less than the size of minimal complete logs is lower than the size of minimal complete by an average of 52.74 %, while in one example only their values are equal. Besides that, Table 2 also shows that the size of the minimal weakly complete logs in the observed examples is only 2 or 3 traces.

[9] The models were found by searching the Web for the keywords: block-structured parallel process, parallel business process, activity diagram, BPMN diagrams etc.

N	Number of Traces in Logs			N_{mwcl} Less Than N_{mcl}	N_{mwcl} Less Than N_{mccl}
	N_{mcl}	N_{mccl}	N_{mwcl}	%	%
1	2	2	2	0.00	0.00
12	3	2	2	33.33	0.00
13	4	2	2	50.00	0.00
39	4	3	2	50.00	33.33
1	4	3	3	25.00	0.00
5	5	2	2	60.00	0.00
6	5	3	2	60.00	33.33
3	5	4	2	60.00	50.00
4	6	2	2	66.67	0.00
3	6	3	2	66.67	33.33
1	6	5	2	66.67	60.00
3	7	3	2	71.43	33.33
3	8	4	2	75.00	50.00
1	9	4	2	77.78	50.00
1	9	4	3	66.67	25.00
2	10	3	3	70.00	0.00
1	10	5	3	70.00	40.00
1	11	3	2	81.82	33.33
Total				On average less by	
100				52.742 %	22.08 %

Table 2. Results of the comparative analysis of the minimal size of complete, causaly complete and weakly complete logs

It can also be seen from Table 2 that the minimal weakly complete event logs are in average smaller than the minimal causaly complete logs by 22.08 %, observed in a sample of 100 examples. In none of the observed examples the number of traces in the minimal causally complete event log is lower than the number of traces in the minimal weakly complete log.

In order to confirm that the hypothesis that the size of weakly complete logs is lower than the size of complete logs and causaly complete logs is statistically relevant, we have applied the Wilcoxon-Mann-Whitney rank-sum nonparametric test [28] on the results from Table 2.

5.2.1 Proof of the Hypothesis That the Minimal Weakly Complete Event Logs Are Smaller Than the Minimal Complete Event Logs

If we denote: X = size of minimal weakly complete logs, and Y_1 = size of minimal complete logs, it is needed to test the null hypothesis that distributions of these two marks (labels) are equal, i.e., $H_0 : F_X = F_{Y1}$, against the alternative hypothesis H_1: "The size of minimal weakly complete logs X is lower than the size

of minimal complete logs Y_1". The corresponding critical area C in this case is in Table 3.

H_0	H_1	C
$F_X = F_{Y1}$	X is lower than Y_1	$z_0 \leq -z_{0.5-\alpha}$

Table 3.

Applying the Wilcoxon-Mann-Whitney test on the obtained experimental analysis results, the following values are obtained:

$$n_1 = 100; \quad n_2 = 100; \quad n = n_1 + n_2 = 200;$$

$$V = 0; \quad E(V) = n_1 n_2/2 = 4\,900.5;$$

$$D(V) = E(V)(n+1)/6 = 162\,533.2;$$

$$z_0 = (V - E(V))/[D(V)]^{\frac{1}{2}} = -12.125.$$

For the level of significance $\alpha = 0.05$, the critical area of this test is $C = (-\infty, -1.645]$. Since the realized value of the test statistic z_0 belongs to the critical area C, the null hypothesis H_0 is rejected in favour of alternative hypothesis H_1. In other words, for the level of significance $\alpha = 0.05$, it can be concluded that the assertion that the size of minimal weakly complete logs is lower than the size of minimal complete logs is statistically significant.

5.2.2 Proof of the Hypothesis That the Minimal Weakly Complete Event Logs Are Smaller Than the Minimal Causally Complete Event Logs

If we denote: X = size of minimal weakly complete logs, and Y_2 = size of minimal causaly complete logs, it is needed to test the null hypothesis that distributions of these two marks (labels) are equal, i.e., $H_0 : F_X = F_{Y2}$, against the alternative hypothesis H_1: "The size of minimal weakly complete logs X is lower than the size of minimal causaly complete logs Y_2". The corresponding critical area C in this case is in Table 4.

H_0	H_1	C
$F_X = F_{Y2}$	X is lower than Y_2	$z_0 \leq -z_{0.5-\alpha}$

Table 4.

Applying the Wilcoxon-Mann-Whitney test on the obtained experimental analysis results, the following values are obtained:

$$n_1 = 100; \quad n_2 = 100; \quad n = n_1 + n_2 = 200;$$

$$V = 35 + 35 + 35 + 35 + 35 = 175; \quad E(V) = n_1 n_2/2 = 5\,000;$$

$$D(V) = E(V)(n+1)/6 = 167\,500;$$
$$z_0 = (V - E(V))/[D(V)]^{\frac{1}{2}} = -11.789.$$

For the level of significance $\alpha = 0.05$, the critical area of this test is $C = (-\infty, -1.645]$. Since the realized value of the test statistic z_0 belongs to the critical area C, the null hypothesis H_0 is rejected in favour of alternative hypothesis H_1. In other words, for the level of significance $\alpha = 0.05$, it can be concluded that the assertion that the size of minimal weakly complete logs is lower than the size of minimal causaly complete logs is statistically significant.

5.2.3 Influence of the Structure of Networks on the Event Log Size

Observing the structure of networks in the considered examples, the experimental analysis has shown that the size of the event log, from which the original networks can be discovered, can depend on the number of parallel branches in the network, as well as on the total number of activities in mutually parallel branches.

In Table 3 the results of the performed experimental analysis are presented, in which considered examples are grouped according to the total number of activities in parallel branches and the number of parallel branches in the network. Considering such groups of examples, sizes minimal complete, minimal causaly complete and minimal weakly complete logs are presented, as well as the difference between them, and the difference between the total number of activities in parallel branches and the number of parallel branches in the network.

From Figure 6 (and from Table 3) it can be seen that the size of minimal complete logs is proportional to the total number of activities in mutually parallel branches. It can also be seen that the number of activities in parallel branches does not affect the size of minimal causaly complete and minimal weakly complete logs.

From Figure 7 (and from Table 3) it can be seen that the number of parallel branches in the network does not affect the size of minimal complete and minimal weakly complete event logs. It can also be seen that the size of minimal causaly complete logs is proportional (nearly equal) to the total number of parallel branches in the network.

From Figure 8 (and from Table 3) it can be seen that the difference between the size of the minimal complete logs and the size of the minimal weakly complete logs, expressed in the number of traces, is proportional to the difference between the total number of activities in parallel branches and the number of parallel branches in the network. The difference between the total number of activities in parallel branches and the number of parallel branches in the network does not affect the relationship between the sizes of the minimal causaly complete and minimal weakly complete event logs.

N	N_a	N_b	$N_a - N_b$	N_{mcl}	N_{mccl}	N_{mwcl}	$N_{mcl} - N_{mwcl}$	$N_{mccl} - N_{mwcl}$
1	2	2	0	2	2	2	0	0
12	3	2	1	3	2	2	1	0
39	3	3	0	4	3	2	2	1
13	4	2	2	4	2	2	2	0
1	4	3	1	4	3	2	2	1
3	4	3	1	5	3	2	3	1
1	4	3	1	6	3	2	4	1
2	4	4	0	5	4	2	3	2
4	5	2	3	5	2	2	3	0
1	5	3	2	5	2	2	3	0
3	5	3	2	5	3	2	3	1
1	5	3	2	5	4	2	3	2
1	5	5	0	6	5	2	4	3
4	6	2	4	6	2	2	4	0
2	6	3	3	6	3	2	4	1
1	6	3	3	7	3	2	5	1
1	6	4	2	8	4	2	6	2
2	7	3	4	7	3	2	5	1
2	7	4	3	8	4	2	6	2
1	8	4	4	9	4	2	7	2
2	8	3	5	10	3	3	7	0
1	8	4	4	9	4	3	6	1
1	9	4	5	9	4	3	6	1
1	9	5	4	10	5	3	7	2
1	10	3	7	11	3	2	9	1

Table 5. The influence of the number of parallel branches in the network and the total number of activities in each parallel branch on the event log size

6 CONCLUSIONS

The examples show that with our modification of the PM discovering technique, we are able to reduce the problem of completeness of logs in parallel processes that occur in the basic α algorithm. That way, we can improve the efficiency of obtaining a block-structured process model, with the meaning that our algorithm can guarantee the discovery of the original model from significantly smaller logs, which do not satisfy the condition of completeness. For that reason, we first defined the causally complete logs [5] and then the weakly complete logs presented in this paper.

The contribution that we have made with the paper goes in two directions. The first is that by defining a new type of event logs – weakly complete event logs, we have made an improvement with regards to overcoming of the conditions of completeness and causall completeness. The weakly complete logs were cre-

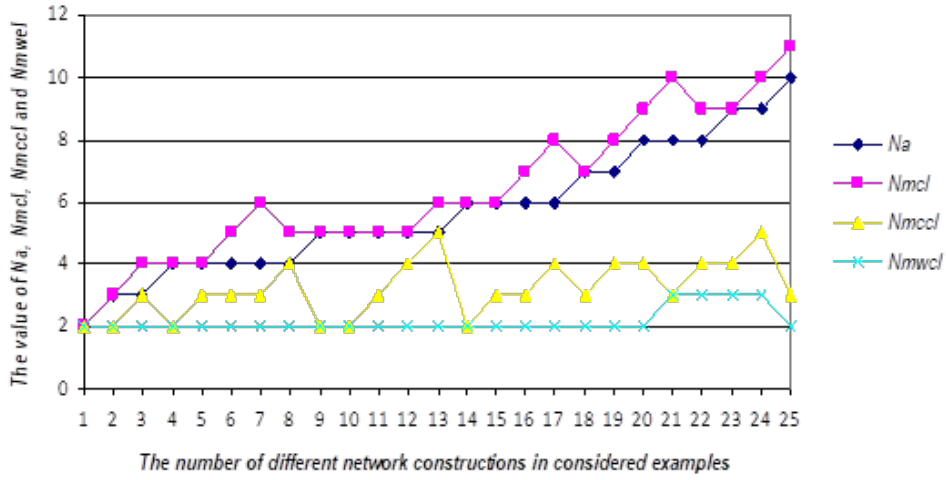

Figure 6. The relation between N_{mcl}, N_{mccl}, N_{mwcl} and N_a

ated as a consequence of our efforts to improve the properties of causally complete logs, primarily in terms of their size. The detailed experimental analysis presented in this paper has just shown that such an improvement has been realized.

Comparative analysis of the results showed that weakly complete event logs can be significantly smaller than both complete and causally complete event logs by the number of traces from which the process model can be discovered.

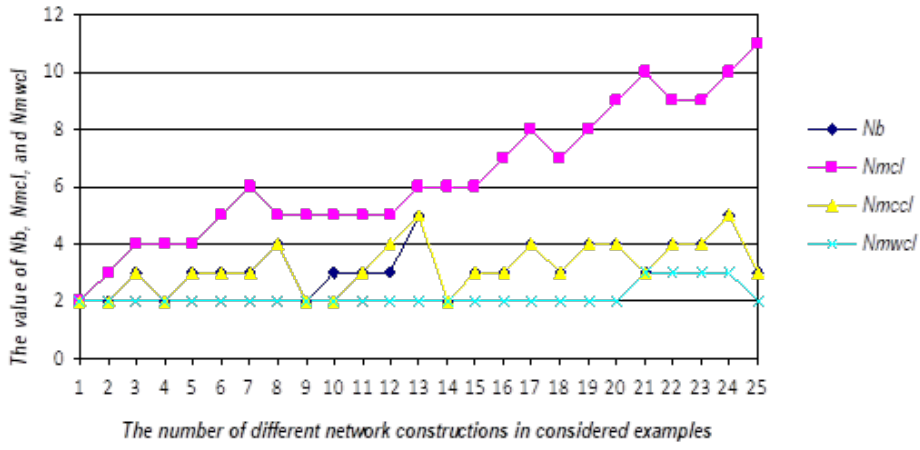

Figure 7. The relation between N_{mcl}, N_{mccl}, N_{mwcl} and N_b

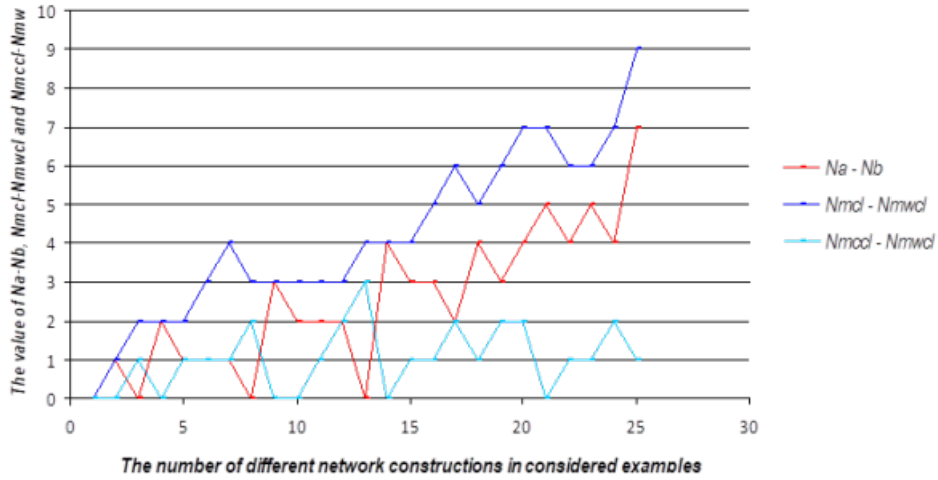

Figure 8. The relation between differences: $N_{mcl} - N_{mwcl}$, $N_{mccl} - N_{mwcl}$ and $N_a - N_b$

The other contribution of defining weakly complete event logs presented in this paper is that they enabled the interactive generation of parallel business process models by demonstration, which was our primary goal. To accomplish our goal, a graphical user interface (GUI) was created, through which the user demonstrates different scenarios of process execution. The graphical user interface that we created is a tool that visually shows steps of $\alpha^{\|}$-algorithm. Such tool could serve as a learning tool and playground for those who want to learn more about how the much better known and more general α-algorithm, which is based on the same principles, functions.

Our assumptions and preconditions for process models (that have to be block-structured parallel models), to which our algorithm and technique are applicable, may look as a serious restriction. However, our solution still covers a respectably wide subclass of process models and represents a first step in a more ambitious attempt to solve the very serious problem of log completeness. In our future research, we will try to expand our work to other categories of processes.

REFERENCES

[1] van der Aalst, W. M. P.: Process Mining: Discovery, Conformance and Enhancement of Business Processes. Springer-Verlag, Berlin, 2011, doi: 10.1007/978-3-642-19345-3.

[2] van der Aalst, W.—Weijters, T.—Maruster, L.: Workflow Mining: Discovering Process Models from Event Logs. IEEE Transactions on Knowledge and Data Engineering, Vol. 16, 2004, No. 9, pp. 1128–1142, doi: 10.1109/TKDE.2004.47.

[3] van der Aalst, W. M. P.—Weijters, A. J. M. M.—Maruster, L.: Workflow Mining: Which Processes Can Be Rediscovered? BETA Working Paper Series, Vol. 74, Eindhoven University of Technology, Eindhoven, 2002.

[4] van der Aalst, W. M. P.: Verification of Workflow Nets. In: Azéma, P., Balbo, G. (Eds.): Application and Theory of Petri Nets 1997 (ICATPN 1997). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 1248, 1997, pp. 407–426, doi: 10.1007/3-540-63139-9_48.

[5] Lekić, J.—Milićev, D.: Discovering Block-Structured Parallel Process Models from Causally Complete Event Logs. Journal of Electrical Engineering, Vol. 67, 2016, No. 2, pp. 111–123, doi: 10.1515/jee-2016-0016.

[6] Sun, H.—Du, Y.—Qi, L.—He, Z.: A Method for Mining Process Models with Indirect Dependencies via Petri Nets. IEEE Access, Vol. 7, 2019, pp. 81211–81226, doi: 10.1109/ACCESS.2019.2923624.

[7] Lekić, J.—Milićev, D.—Stanković, D.: Generating Block-Structured Parallel Process Models by Demonstration. Applied Sciences, Vol. 11, 2021, No. 4, Art. No. 1876, doi: 10.3390/app11041876.

[8] Agrawal, R.—Gunopulos, D.—Leymann, F.: Mining Process Models from Workflow Logs. In: Schek, H. J., Alonso, G., Saltor, F., Ramos, I. (Eds.): Advances in Database Technology (EDBT '98). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 1377, 1998, pp. 467–483, doi: 10.1007/BFb0101003.

[9] Cook, J. E.—Wolf, A. L.: Discovering Models of Software Processes from Event-Based Data. ACM Transactions on Software Engineering and Methodology, Vol. 7, 1998, No. 3, pp. 215–249, doi: 10.1145/287000.287001.

[10] Cook, J. E.—Wolf, A. L.: Event-Based Detection of Concurrency. Proceedings of the Sixth International Symposium on the Foundations of Software Engineering (FSE-6), ACM SIGSOFT Software Engineering Notes, Vol. 23, 1998, No. 6, pp. 35–45, doi: 10.1145/291252.288214.

[11] Herbst, J.: Dealing with Concurrency in Workflow Induction. In: Baake, U., Zobel, R., Al-Akaidi, M. (Eds.): Proceedings of the European Concurrent Engineering Conference (ECEC 2000), Leicester, UK, 2000.

[12] Herbst, J.—Karagiannis, D.: Integrating Machine Learning and Workflow Management to Support Acquisition and Adaptation of Workflow Models. Intelligent Systems in Accounting, Finance, and Management, An International Journal, Vol. 9, 2000, No. 2, pp. 67–92, doi: 10.1002/1099-1174(200006)9:2<67::AID-ISAF186>3.0.CO;2-7.

[13] Schimm, G.: Process Miner – A Tool for Mining Process Schemes from Event-Based Data. In: Flesca, S., Greco, S., Ianni, G., Leone, N. (Eds.): Logics in Artificial Intelligence (JELIA 2002). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 2424, 2002, pp. 525–528, doi: 10.1007/3-540-45757-7_47.

[14] van der Aalst, W. M. P.—Stahl, C.: Modeling Business Processes: A Petri Net-Oriented Approach. MIT Press, Cambridge, MA, 2011, doi: 10.7551/mitpress/8811.001.0001.

[15] de Medeiros, A. K. A.—van der Aalst, W. M. P.—Weijters, A. J. M. M.: Workflow Mining: Current Status and Future Directions. In: Meersman, R., Tari, Z., Schmidt, D. C. (Eds.): On the Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE (OTM 2003). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 2888, 2003, pp. 389–406, doi: 10.1007/978-3-540-39964-3_25.

[16] de Medeiros, A. K. A.—Weijters, A. J. M. M.—van der Aalst, W. M. P.: Genetic Process Mining: An Experimental Evaluation. Data Mining and Knowledge Discovery, Vol. 14, 2007, No. 2, pp. 245–304, doi: 10.1007/s10618-006-0061-7.

[17] Wen, L.—van der Aalst, W. M. P.—Wang, J.—Sun, J.: Mining Process Models with Non-Free-Choice Constructs. Data Mining and Knowledge Discovery, Vol. 15, 2007, No. 2, pp. 145–180, doi: 10.1007/s10618-007-0065-y.

[18] van der Aalst, W.—Adriansyah, A.—de Medeiros, A. K. A.—Arcieri, F.—Baier, T. et al.: Process Mining Manifesto. In: Daniel, F., Barkaoui, K., Dustdar, S. (Eds.): Business Process Management Workshops (BMP 2011). Springer, Berlin, Heidelberg, Lecture Notes in Business Information Processing, Vol. 99, 2012, pp. 169–194, doi: 10.1007/978-3-642-28108-2_19.

[19] Carmona, J.—Cortadella, J.—Kishinevsky, M.: A Region-Based Algorithm for Discovering Petri Nets from Event Logs. In: Dumas, M., Reichert, M., Shan, M. C. (Eds.): Business Process Management (BPM 2008). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 5240, 2008, pp. 358–373, doi: 10.1007/978-3-540-85758-7_26.

[20] van Dongen, B. F.—Alves de Medeiros, A. K.—Wen, L.: Process Mining: Overview and Outlook of Petri Net Discovery Algorithms. In: Jensen, K., van der Aalst, W. M. P. (Eds.): Transactions on Petri Nets and Other Models of Concurrency II. Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 5460, 2009, pp. 225–242, doi: 10.1007/978-3-642-00899-3_13.

[21] Günther, C. W.—van der Aalst, W. M. P.: Fuzzy Mining – Adaptive Process Simplification Based on Multi-Perspective Metrics. In: Alonso, G., Dadam, P., Rosemann, M. (Eds.): Business Process Management (BPM 2007). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 4714, 2007, pp. 328–343, doi: 10.1007/978-3-540-75183-0_24.

[22] Weijters, A. J. M. M.—Ribeiro, J. T. S.: Flexible Heuristics Miner (FHM). Proceedings of the 2011 IEEE Symposium on Computational Intelligence and Data Mining (CIDM 2011), Paris, France, 2011, pp. 310–317, doi: 10.1109/CIDM.2011.5949453.

[23] Leemans, S. J. J.—Fahland, D.—van der Aalst, W. M. P.: Discovering Block-Structured Process Models from Incomplete Event Logs. In: Ciardo, G., Kindler, E. (Eds.): Applications and Theory of Petri Nets and Concurrency (PETRI NETS 2014). Springer, Cham, Lecture Notes in Computer Science, Vol. 8489, 2014, pp. 91–110, doi: 10.1007/978-3-319-07734-5_6.

[24] Lekic, J.—Milicev, D.: Discovering Models of Parallel Workflow Processes from Incomplete Event Logs. In: Hammoudi, S., Pires, L. F., Desfray, P., Filipe, J. (Eds.): Proceedings of the 3rd International Conference on Model-Driven Engineering and Software Development (MODELSWARD 2015), Angers, Loire Valley, France, 2015, pp. 477–482, doi: 10.5220/0005242704770482.

[25] Ling, J. M.—Zhang, L.—Feng, Q.: An Improved Structure-Based Approach to Measure Similarity of Business Process Models. Proceedings of the 26th International Conference on Software Engineering and Knowledge Engineering (SEKE 2014), 2014, pp. 377–380.

[26] van der Aalst, W. M. P.—van Dongen, B. F.—Günther, C. W.—Mans, R. S.—Alves de Medeiros, A. K.—Rozinat, A.—Rubin, V.—Song, M.—Verbeek, H. M. W.—Weijters, A. J. M. M.: ProM 4.0: Comprehensive Support for Real Process Analysis. In: Kleijn, J., Yakovlev, A. (Eds.): Petri Nets and Other Models of Concurrency – ICATPN 2007. Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 4546, 2007, pp. 484–494, doi: 10.1007/978-3-540-73094-1_28.

[27] Weakly Complete Event Logs. `https://drive.google.com/drive/u/0/folders/1TKM-6T03i4qZNYZ6a-6yAB7K1x5GHsDD`.

[28] Popovic, C. B.: Mathematical Statistics. Faculty of Sciences and Mathematics, University of Niš, 2009 (in Serbian).

Julijana B. Lekić received her dipl. Eng. degree in electrical engineering from the Faculty of Electrotechnical Engineering, University of Priština (1989), M.Sc. degree from the Faculty of Electrotechnical Engineering, University of Belgrade (1998), and Ph.D. from the Faculty of Technical Sciences, University of Priština (2016), Serbia. She is Assistant Professor at the Faculty of Technical Sciences, University of Priština (temporarily displaced in Kosovska Mitrovica). Her current research interests are in the field of computer science, particularly information systems, software engineering, and business process modeling and mining.

Dragan S. Milićev is Professor at the University of Belgrade, Faculty of Electrical Engineering. He received his dipl. Eng. degree in 1993, M.Sc. in 1995, and Ph.D. in 2001, all from the University of Belgrade. He is specialized in software engineering, model-based engineering, model-driven development, UML, software architecture and design, information systems, and real-time systems. He is a member of the Editorial Board of Springer's Software and Systems Modeling journal (SoSyM). He authored three books on object-oriented programming and UML, published in Serbian, and a book in English, published by Wiley/Wrox, entitled "Model-Driven Development with Executable UML". With thirty years of extensive industrial experience in building complex commercial software systems, he has been serving as the chief software architect, project manager, or consultant in a number of international projects.