Formal Concept Analysis for Trace Clustering in Process Mining

Salah Eddine Boukhetta, Marwa Trabelsi

Formal Concept Analysis for Trace Clustering in Process Mining

Salah Eddine Boukhetta[1] and Marwa Trabelsi[1]

University of La Rochelle, L3i laboratory, La Rochelle, France
salah.boukhetta@univ-lr.fr
marwa.trabelsi@univ-lr.fr

Abstract. Modeling user interaction in information systems (IS) using Process Mining techniques is an intriguing requirement for designers looking to optimize the use of various IS functionalities and make stored resources more accessible. Discovered models can thus be used in future work to present a set of recommendations to IS users. However, the large number of generated logs or user's traces result in complex models. To address this problem, in this paper, we propose a new methodology for grouping user traces prior to modeling using Formal Concept Analysis. The clustering method relies on the `GALACTIC` framework to generate relevant concepts, which are then used to select a specific concept for each trace using a distance measure. Considering a trace as a sequence, the proposed method generate concepts based on maximal common subsequences. The experimental part shows that our method successfully found the original clusters on a simulated dataset.

Keywords: Formal Concept Analysis, Trace clustering, Process mining, `GALACTIC`

1 Introduction

The practical need to investigate how users interact with information systems by examining their digital traces is growing significantly. Indeed, companies' business processes through these systems are fragmented, leaving users to determine their own path to achieve their objectives. In such a context, the user's journey to perform a task (purchase of a product on a website, search for a document in a digital library, etc.) corresponds to an "unstructured process". The meaning, structure and results depend on the user's skills.

In this work, we focus on the case of digital libraries in particular. Digital libraries (DLs) are complex and advanced information systems that attract a multitude of users for various information retrieval tasks. They store and manage a large amount of digital documents and objects. They provide many services to their users, such as the information retrieval system, personalization, etc [25].

To model the different paths leading DLs users to the resolution of their information retrieval task, we chose to use process mining techniques. Process mining proposes a suite of methods for discovering and modeling human behavior from

digital traces generated during the interaction with information systems [1]. The main advantage of these techniques is their ability to handle the whole information retrieval process (the sequence of activities carried out by users from the beginning to the end of their navigation). Therefore, a model depicting the users' interactions can help system designers to answer users' practical requirements, on the one hand, and to present a set of recommendations to them, on the other hand. Moreover, modeling user's interactions could be helpful to optimize systems' design and improve the most used features

However, due to the increasing use of DLs, very large and complex event logs are produced. These complex event logs pose additional challenges for process mining techniques. The large number of generated logs leads to complex models, commonly referred to as spaghetti models. These models can be challenging to interpret and may not fulfill all users' objectives [1,28]. Trace clustering techniques are recommended to address this issue [8]. These techniques group event logs based on similarities in executed activity sequences. Clustering approaches, on the other hand, face many challenges. When applied to user interactions in DLs, for example, numerous DLs users may have comparable sub-processes in their navigation despite conducting a different type of research or pursuing the same goal. Furthermore, despite the number of target clusters being uncertain, created process models based on clustering must present disjoint models to identify users' journeys.

In this paper, we propose a new trace clustering method for grouping user interactions prior to modeling using Formal Concept Analysis (FCA for short), a new FCA application. FCA is a data analysis method that focuses on the relationship between a set of objects and a set of attributes in data. A concept lattice, which is the main structure of FCA, gives us valuable insights from a dual viewpoint based on the objects and the attributes. In this work, FCA can be a solution to comprehend the users' navigations, group them, and extract the characteristics shared by this type of navigation. One limitation of the FCA framework is the generation of a large number of concepts, which makes extracting information from data more complex at times. To overcome this limitation, we attempt to combine the FCA framework with traditional clustering methods, beginning with generating a concept lattice and then extracting clusters based on the generated concepts.

The rest of the paper is structured as follows. After introducing FCA and process mining basics in Section 2 and 3 respectively, we discuss the related work in Section 4. Section 5 describes the proposed approach. Then, Section 6 describes experiments on the synthetic dataset. We consecutively present the event data and the experimental results. Finally, Section 7 concludes the paper and offers directions for future work

2 Formal Concept Analysis

Formal Concept Analysis (FCA) is a field of data analysis for identifying relationships in the data set. It appears in 1982 [27], then in the Ganter and Wille's 1999 work [12], it is issued from a branch of applied lattice theory that first

appeared in the book of Barbut and Monjardet in 1970 [4]. The lattice property guarantees both a hierarchy of clusters, and a complete and consistent navigation structure for interactive approaches [9]. FCA is classically designed to deal with data described by sets of attributes, thus binary data. The formalism of pattern structures [11,16] and abstract conceptual navigation [9] extend FCA to deal with non-binary data such as sequences, where patterns describe data. Inspired by pattern structures, the NextPriorityConcept algorithm [7] proposes a pattern mining approach for heterogeneous and complex data. The `GALACTIC` platform implements the NextPriorityConcept algorithm and offers an ecosystem of extensions for data processing. In a recent work, sequence data analysis was proposed and implemented in the `GALACTIC` platform [6].

2.1 Definitions

We present some definitions related to FCA elements here, and then we briefly present the NextPriorityConcept algorithm.

Let $\langle G, M, I\rangle$ be a *formal context* where G is a non-empty set of objects, M is a non-empty set of attributes and $I \subseteq G \times M$ is a binary relation between the set of objects and the set of attributes. Let $(2^G, \subseteq) \underset{\alpha}{\overset{\beta}{\leftrightarrows}} (2^M, \subseteq)$ be the corresponding *Galois connection* where:

- $\alpha : 2^G \to 2^M$ is an application which associates a subset $B \subseteq M$ to every subset $A \subseteq G$ such that $\alpha(A) = \{b \ : \ b \in M \wedge \forall a \in A, aIb\}$;
- $\beta : 2^M \to 2^G$ is an application which associates a subset $A \subseteq G$ to every subset $B \subseteq M$ such that $\beta(B) = \{a \ : \ a \in G \wedge \forall b \in B, aIb\}$.

A concept is a pair (A, B) such that $A \subseteq G$, $B \subseteq M$, $B = \alpha(A)$ and $A = \beta(B)$. The set A is called the *extent*, whereas B is called the *intent* of the concept (A, B). There is a natural hierarchical ordering relation between the concepts of a given context which is called the subconcept-superconcept relation:

$$(A_1, B_1) \leq (A_2, B_2) \iff A_1 \subseteq A_2 (\iff B_2 \subseteq B_1)$$

The ordered set of all concepts makes a complete lattice called the *concept lattice* of the context, that is, every subset of concepts has an infimum (meet) and a supremum (join).

2.2 The NextPriorityConcept algorithm

The NextPriorityConcept algorithm [7] computes concepts for heterogeneous and complex data for a set of objects G. It is inspired by Bordat's algorithm[5], also found in Linding's work [20], that recursively computes the immediate predecessors of a concept, starting with the top concept $(G, \alpha(G))$ containing the whole set of objects, until no more concepts can be generated.

Descriptions as an application generating predicates The algorithm introduces the notion of *description* δ as an application to provide predicates describing a set of objects $A \subseteq G$. Each concept $(A, \delta(A))$ is composed of a

subset of objects A and a set of predicates $\delta(A)$ describing them, corresponding to their pattern. Such generic use of predicates makes it possible to consider heterogeneous data as input, i.e., numerical, discrete or more complex data. A concept $(A, \delta(A))$ can be interpreted as a generalized convex hull, where each border of the hull corresponds to a predicate, and the elements inside the hull correspond to the objects A that verify all the predicates. Unlike classical pattern structures, predicates are not globally computed in a preprocessing step, but locally for each concept as the border lines of a convex hull.

Strategies as an application generating selectors The algorithm introduces the notion of *strategy* σ to provide predicates called *selectors* describing candidates for an object reduction of a concept $(A, \delta(A))$ i.e., for predecessors of $(A, \delta(A))$ in the pattern lattice. A selector proposes a way to select a reduced set $A' \subset A$ of objects and the concept $(A', \delta(A'))$ is candidate to be a predecessor of $(A, \delta(A))$. Several strategies can generate predecessors of a concept, going from the naive strategy classically used in FCA that considers all the possible predecessors, to strategies allowing to obtain few predecessors and smaller lattices. Selectors are only used for the predecessors' generation, they are not kept either in the description or in the final set of predicates. Therefore, choosing or testing several strategies at each iteration in a user-driven pattern discovery approach would be interesting.

The main result in [7] states that the NextPriorityConcept algorithm computes the formal context $\langle G, P, I_P \rangle$ and its concept lattice (where P is the set of predicates describing the objects in G, and $I_P = \{(a, p),\ a \in G, p \in P\ :\ p(a)\}$ is the relation between objects and predicates) if description δ verifies $\delta(A) \sqsubseteq \delta(A')$ for $A' \subseteq A$.

2.3 GALACTIC

`GALACTIC` is a new platform for computing patterns from heterogeneous and complex data that extend the approach of pattern structures [11] and logical concept analysis [10]. It's a development platform for a generic implementation of the NextPriorityConcept [7] algorithm allowing easy integration of new plugins for characteristics, descriptions, strategies and meta-strategies.

The `GALACTIC` platform allows the analysis of binary, numerical and categorical data. Sequences handling have been added to the platform recently [6] and multiple descriptions and strategies are available for simple, temporal, and interval sequences. Simple Sequences have three descriptions: Maximal Common Subsequences, Prefixed Common Subsequences and K-Common Subsequences. Simple Sequences also have two strategies: Naive Strategy and Augmented Strategy. They aim to extract subsequences from a sequence in various ways.

Meta-strategies act as filters for other strategies. The *LimitFilter* meta-strategy selects predecessors whose measure is above/below a threshold. It is possible for example to use confidence, support and cardinality measures to limit the generation of concepts to those who respect a threshold of these measures.

3 Process Mining

The idea of process mining was introduced by Aalst in 2004 [2]. Process mining is a data analytics technique that extracts knowledge from execution traces in today's information systems. These techniques provide novel methods for discovering (**Process discovery**), evaluating (**Conformance checking**), and improving processes (**Process enhancement**) in a wide range of application domains [1]. The growing interest in process mining is justified, on the one hand, by the large number of recorded traces that provide detailed information about the process history and, on the other hand, by the ability of these techniques to deal with the entire process (a complete process having a start activity and an end activity).

CaseId	**User**	**Date**	**Activity label**
1	$user_1$	2016-01-12T10:34:25	home index
2	$user_2$	2016-01-12T10:36:25	home index
1	$user_1$	2016-01-12T10:34:26	home languages
1	$user_1$	2016-01-12T10:34:28	language selection
3	$user_3$	2016-01-12T10:36:26	home index
3	$user_3$	2016-01-12T10:36:27	catalog show

Table 1: Sample of DL event logs

Event logs and process models are two fundamental artifacts used in process mining [1]. An event log corresponds to the set of execution traces (*i.e.* process instances) that delivers a specific service or product. For example, to make an online purchase, a user has to subscribe, select a product and proceed to payment. All of these activities are a specific trace of the main process (online purchase). For example, as shown in Table 1, for DLs, each user could be a case that follows a research process. The sequence of events related to a particular case is called a trace. Each row records an executed event, which contains information such as the identifier of each event (**CaseId**), the userId, the **activity label**, the **timestamp** (i.e. day, hour, minute and second) and some additional attributes regarding the event. Formally, an **event logs** $L = \{t_1, t_2, ..., t_k\}$ is a set of k **traces** where each trace t_i $(1 \leq i \leq k)$ is a set of n_i consecutive events $t_i =< e_{i1}, e_{i2}, ...e_{in_i} >$ made by the same CaseID [1].

Process models are destined to represent the whole of event logs. They depict the sequence of activities, decision points, and flow of information or resources within the process. Process models can be created using various notations, such as BPMN (Business Process Model and Notation), Petri nets or Directly follows graphs [1].

Many process discovery methods have been proposed to generate process models in the literature. For digital library users' interactions, *Trabelsi et al.* [26] studied the contribution of Process Mining techniques to analyze the digital library users' behaviors and to thus generate effective models from such unstructured processes. The authors executed the most-known process discovery techniques through two sets of event logs produced by users researching documents

in a digital library. Then, they presented and evaluated the models generated by best performing Process Mining techniques. Results showed that the Inductive Miner algorithm [17] achieves the best scores for both datasets. To evaluate the quality of the resulting process models of each method, authors used four typical metrics [3]: **Fitness**, which indicates the accordance of the model with the event log. An existing way to calculate the Fitness, is to determine how well the event log aligns with the model when replaying the traces on it [3]. **Precision** corresponds to the rate of activities in the event logs compared to the total of activities enabled in the process model. **Generalization**: it is related to the unseen behavior. This criterion aims to measure the ability of the model to generalize the behavior seen in the logs. A suitable model has to find a balance between these metrics [1]. Finally, **Simplicity** is related to a process model's complexity by capturing the simplicity dimension.

However, many other process mining works showed that discovering a single process model for an entire log is unsuitable, especially over a large dataset. Discovery algorithms usually lead to complex and/or overfitted models such as the well-known spaghetti or flower model [1]. Furthermore, various types of users behaviors can be included in the overall event logs. To tackle this issue, existing works proposed trace clustering methods prior to modeling [8].

Trace clustering in process mining is a collection of techniques for grouping sequences with similar characteristics to extract a model for each. Four approaches have been developed: trace-based clustering groups similar traces based on their syntax similarity; Feature-based clustering converts each trace into a vector of features based on its characteristics; Model-based clustering uses the process model's properties as input for clustering; and hybrid-based clustering combines previous methods [24,28,25]. Our work in this article, belongs to the trace-based approach since we consider the syntax similarity of traces for the clustering.

4 Related works

This section introduces and discusses related works on the application of FCA in clustering and the enhancement of process models.

In the context of **clustering**, authors in [23] demonstrated how FCA techniques could be used for clustering categorical data. A global support value was used to specify which concepts can be candidate clusters. The best cluster for each object was then determined using a score function. Furthermore, to assist museum researchers in analyzing and evaluating item placement and visiting styles, authors in [15] proposed an FCA approach comprised of two independent steps: clustering and trajectory mining. A specific dataset was concerned with the trajectories of visitors. Each trajectory is made up of a series of visited items. Given that the trajectory dataset can be regarded as a sequential dataset, a proper sequence clustering method is used where the distance between any two sequences is obtained from the number of their common subsequences. On the other hand, the mining of trajectory patterns is performed by two methods based on FCA. These patterns are then used to find the characteristic behavior of each cluster.

In the context of **process model enhancement** in process mining, authors in [13], proposed to apply the FCA to process enhancement which is one of the main goals of process mining. Process enhancement consists of analyzing a discovered process from an event log, and improving its efficiency based on the analysis. For process enhancement, by FCA, authors defined subsequences of events whose stops are fatal to the execution of a process as weak points to be removed. In their method, the extent of every concept is a set of event types, and the intent is a set of resources for events in the extent, and then, for each extent, its weakness is calculated by taking into account event frequency. They also proposed some ideas to remove the weakest points. In this line, authors in [22], demonstrated that FCA can provide additional insights in situations closely related to potential value leaks in processes.

A few amounts of works focused on the analysis of digital traces. For instance, authors in [14], proposed a new method for automatically extracting smartphone users' contextual behaviors from the digital traces collected during their interactions with their devices. Their goal was to understand the impact of users' context (eg, location, time, environment, etc.) on the applications they run on their smartphones. Based on the presented works, it is clear that FCA techniques can extract relevant information from event logs, allowing analysts to gain insights into the process and formulate and validate its hypotheses. Trace clustering using FCA can help process mining and allows detecting profiles to have more explainable user behaviors. In this work, we propose a new trace clustering method for grouping execution traces before process modeling.

5 Proposed method: Trace clustering using FCA

The proposed approach can be divided into two parts. The first part consists of generating a concept lattice using the NextPriorityConcept algorithm. Then, in the second part, trace clustering is performed based on the generated clusters in the lattice. The main idea is to use the knowledge carried by the traces to construct a concept lattice. Each concept (cluster) then contains a set of traces with their description.

For this purpose, we use the `GALACTIC` platform described above. The description of a set of traces can be of different forms. For event logs, we use Maximal Common Subsequence description (MCS), where traces are transformed into sequences of activities and then described by their maximal common subsequences. Moreover, the Numerical description describes a set of traces by their maximal and minimal size. Table 2 shows an example of three traces, with their description. As we mentioned in Section3, a trace is a sequence of activities (or events) and a description is a set of predicates. For example, in this table, we have a sequential predicate that represents the common subsequence of the three traces and two numerical predicates defining the maximal and minimal length of the traces.

Therefore, we use these two descriptions and the naive strategies to generate the concept lattice. We also use the *LimitFilter* as meta-strategy that limits the number of generated predecessors to those of cardinality greater than a given

Trace id	Trace	Description
t1	⟨ home_index, home_topics, home_show_topic_selection, catalog_show ⟩	chain match ('home_index', 'catalog_show') and 'nb_activities'>=3 and 'nb_activities'<=5
t2	⟨ home_index, home_topics, catalog_show ⟩	
t3	⟨ home_index, catalog_index_query, catalog_show, catalog_index_filter, catalog_show ⟩	

Table 2: Example of description of three traces

threshold. Our problem here is that a trace may be present in many concepts as FCA generates overlapping clusters. As it seems to be a drawback for FCA-based clustering, it may lead to better results. First, clustering methods are based on the distance between traces which is a kind of distance calculation, whereas using FCA, the concepts/clusters discovery is based on sequences and their common subsequences (symbolic approach). Second, existing clustering methods may be robust, but they generate a local minimum for each execution. Last, the lattice generated by FCA contains by default all the clusters, using FCA will certainly generate the exact clusters we are seeking alongside with other clusters, we just have to find the right ones from the lattice.

One solution to get a disjoint clustering is to assign a score function that evaluates the distance between a trace and all its concepts, and then select the best concept for this trace. The score function was introduced firstly by [23]. The data used by the authors was binary data, and the score function uses the frequency of the attributes. In this work, we propose a score function that uses the distance between a trace and the concept description.

Let $A = \{t_1, t_2, ...t_k\}$ be a set of traces where each trace t_i is a set of events $t_i = \langle e_{i1}, e_{i2}, ...e_{in_i} \rangle$, and $c = (A, \delta(A))$ be a concept, where the description $\delta(A)$ is defined by:

$$\delta(A) = \delta^{Seq}(A) \cup \delta^{Num}(A)$$

Where $\delta^{Seq}(A)$ is a set of predicates of sequential data, and $\delta^{Num}(A)$ is a set of predicates of numerical data. We define $Score$ as a function that gives a distance value between a trace t and a concept c where $t \in A$ by:

$$Score(t, c) = \frac{d^{Seq}(t, \delta^{Seq}(A)) + d^{Num}(t, \delta^{Num}(A)))}{2} \text{ , where:}$$

- $d^{Seq}(t, \delta^{Seq}(A))$ is the *edit distance* between two sequences: the trace sequence, and the predicate sequence of $\delta^{Seq}(A)$ that is a subsequence of t. The edit distance is the minimum number of insertions, deletions, and substitutions required to transform one sequence to another [19].
- $d^{Num}(t, \delta^{Num}(A))$ is simply the mean of absolute values of the difference between the size of t and the numeric values of the predicates in $\delta^{Num}(A)$.

Algorithm 1 introduces the Lattice-Based Trace Clustering approach, which accepts a lattice L and a list of traces T as input and produces a list of disjoint concepts that represent a set of clusters for the traces given as input.

The algorithm starts by calculating each couple's scores (trace, concept) (lines 1, 2). Then it selects one concept with the highest score value for each trace (lines 3, 4). In the case of score equality between a trace and many concepts, the selection is then made by selecting the concept with the highest cardinality. Finally, we create the clusters with the selected concepts and their traces (lines 5-15).

Algorithm 1 Lattice Based Trace Clustering

Input: A concept lattice L and a list of traces T
Output: A set of clusters

1: *// Calculate the scores for each couple (trace, concepts)*
2: $TC \leftarrow$ triples of (Trace, Concept, Score)
3: *// Select the best concept for each trace*
4: $TC \leftarrow$ couples of (Trace, Concept) where the score is the highest
5: *// calculate the clusters*
6: $Clusters \leftarrow \emptyset$ *//* $(Concept, \emptyset)$ *list of clusters*
7: **for** $tc \in TC$ **do**
8: $Trace, Concept \leftarrow tc$
9: **if** $Concept \in Clusters$ **then**:
10: $Clusters[Concept].add(Trace)$
11: **else**
12: $Clusters.add(Concept, \{Trace\})$
13: **end if**
14: **end for**
15: **Return** $Clusters$

6 Experiments

6.1 Datasets

As mentioned in the introduction, we chose information systems conceived for Digital Libraries (DLs) as a case study. To validate our new trace clustering method based on the FCA, we simulated DLs users' search behaviors [1] by reproducing the characteristics of the main categories described by Marchionini [21]. This experimental strategy is simplified compared to the use of real data. Our objective is the validity of the approach. Its main goal is to assess the ability of the proposed approach to model users' behaviors. By leveraging a simulated dataset, the study benefits from the controlled environment and the availability of ground truth information, enabling a comprehensive evaluation of the FCA-based clustering method's performance.

The simulated dataset distinguishes three types of traces. **Lookup** traces where users can access precisely identified documents with few manipulations via the search engine or by browsing through the various document categories. **Borderline** traces where users can access documents within a well-defined subject area, using multiple searching methods and filtering results. Finally, **Ex-**

[1] based on `https://projectblacklight.org/`

ploratory traces, where users can access a wide range of documents in various fields and types, using more advanced search and filtering functions [21].

The simulated dataset contains quantitatively 100 traces distributed as 40 lookup traces, 30 borderline traces and 30 exploratory traces. Each trace is related to a user. Obviously, exploratory traces which are the most complex cover more than 10 types of events for a total of events (310). The borderline traces are made from 6 types of events (170 events) and the lookup traces are limited to 5 types of events for a total of 140 events.

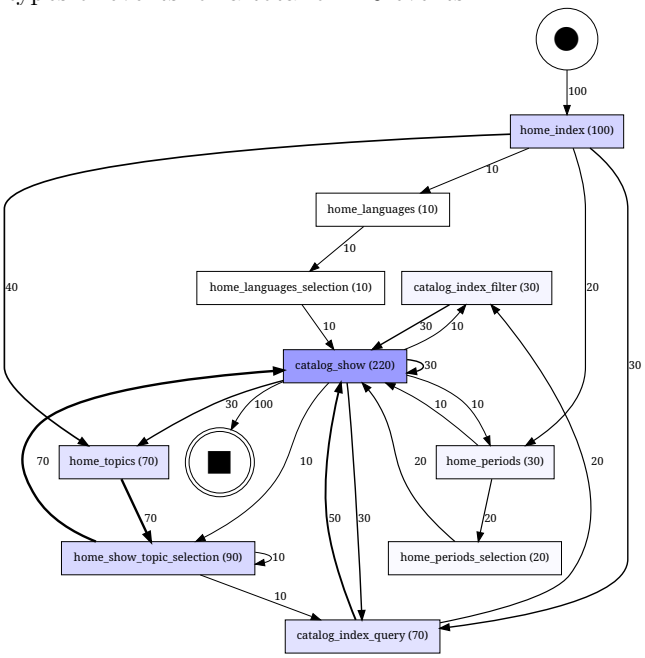

Fig. 1: Discovered Directly Follows Graph for the whole simulated event logs

The process model in Figure 1 is a Directly-Follows Graph (DFG) [2] discovered on the simulated dataset. The process discovery method utilized considers the events and their frequency in the event logs and the frequency of direct succession between events [18]. The event logs will be mapped to a DFG whose vertices are events and edges are the direct relations. However, as previously said, understanding a DFG incorporating many sorts of research is far from simple for designers. It is usually easier to find each kind in a distinct graph rather than from the entire graph (cf. Figure 1).

6.2 Clustering results using FCA

In this part, we provide the experimental results related to the clustering on the simulated DL event logs. Our aim is to retrieve the three desired DL users'

[2] Along this paper, we use Directly-Follows Graph to show our models instead of Petri Net for simplicity.

groups described in the simulated data using the proposed clustering method based on FCA. Figure 2 shows the Hasse diagram of the generated lattice using the MCS and the Numerical descriptions and their respective Naive strategies. We use a *Cardinality* measure to limit the generation of concepts to those where the number of traces didn't exceed a given cardinality parameter (here 25). The generated lattice contains 20 concepts. In this figure, the $ symbol represents the *id* of the concept and the # symbol represents the number of traces of this concept. Each concept comprises two parts, the upper part, where we can see the description represented by a set of predicates, and the lower part where we can identify traces by their ids. The concept $14 here contains 30 traces described by two predicates:

- Chain predicates δ^{Seq}: *chain match ('home_index', 'catalog_show', 'catalog_show', 'catalog_show', 'catalog_show')*
- Numerical predicates δ^{Num}: *'nb_activities'*≥ 11

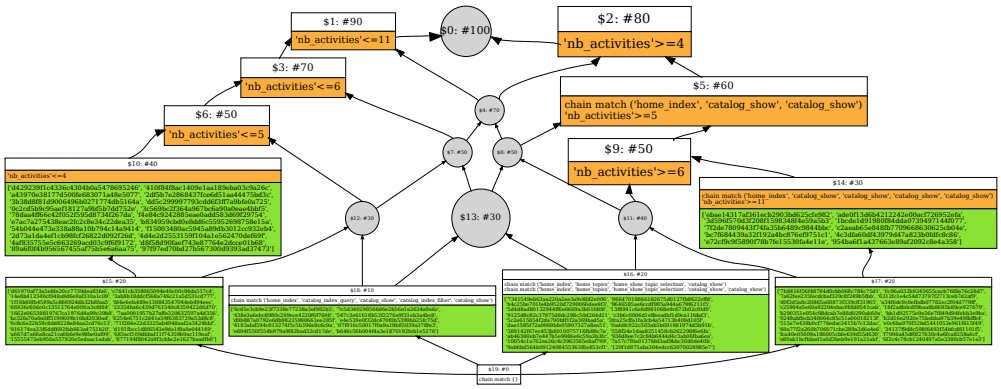

Fig. 2: Hasse diagram of the concept lattice generated by the MCS and Numerical descriptions and their respective Naive strategies with cardinality 25, with a zoom of concept $14

Remember that in this lattice, a trace may be present in different concepts. To perform a disjoint clustering, we use our method to select one concept for each trace. Table 3 shows the selected concepts, their description, and the number of traces in each class. We clearly can conclude that this is a perfect clustering as each concept represents the exact initial class.

Cluster	1	2	3
Description	chain match ('home_index', 'catalog_show') and 'nb_activities'>=3 and 'nb_activities'<=12	chain match ('home_index', 'catalog_show', 'catalog_show') and 'nb_activities'>=5 and 'nb_activities'<=12	chain match ('home_index', 'catalog_show', 'catalog_show', 'catalog_show', 'catalog_show') and 'nb_activities'>=11 and 'nb_activities'<=12
Lookup	40	0	0
Borderline	0	30	0
Exploratory	0	0	30

Table 3: The selected concepts with their description and the number of traces in each class

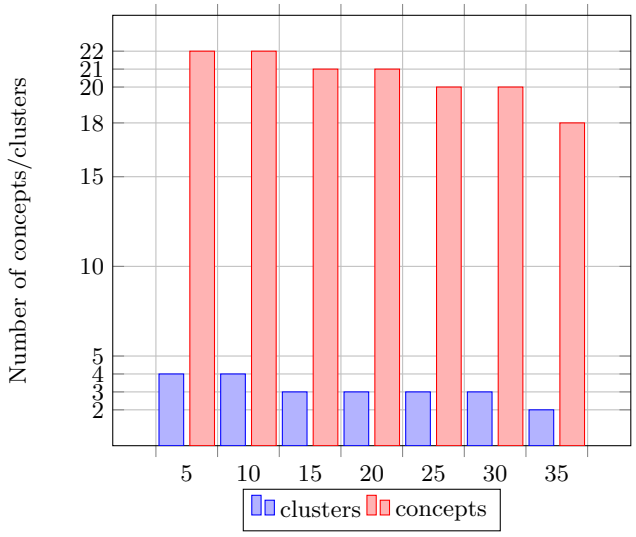

Fig. 3: The number of concepts and clusters according to cardinality change

However, this method only sometimes finds the exact number of classes we are looking for; if we change the cardinality parameter, we may have more/fewer concepts in the lattice and thus more/fewer resulting clusters. Figure 3 shows the number of clusters found according to the cardinality parameter.

6.3 Process modeling Results

In this part, we provide the experimental results related to the process modeling step using the process discovery algorithm (the Inductive Miner algorithm). After discovering the Petri nets using the process discovery algorithm, we used the four metrics, that we mentioned previously in Section 3, to evaluate the models: the Fitness criteria, the Precision, the Generalization and the Simplicity.

Table 4 compares the process models discovered for each event log cluster (three clusters) and the process model mined from the complete simulated dataset. The results indicate that the FCA-based method consistently achieves the highest Precision scores for each discovered process model, demonstrating its effectiveness in accurately identifying process patterns compared to the discovered process models from the complete real logs. Additionally, this method shows a superior balance between Precision and Generalization, suggesting its ability to capture both specific and generalized process behaviors effectively in a more balanced manner. Moreover, the FCA clustering method outperforms others in terms of Simplicity, highlighting its capability to provide straightforward and easily understandable process representations.

	Real logs	**Cluster 1**	**Cluster 2**	**Cluster 3**
Fitness	1	1	1	1
Precision	0,406	**1**	**0,496**	**0,414**
Generalization	0.853	0.734	0.760	0.800
Simplicity	0.696	**0.779**	**0.700**	**0.739**

Table 4: Process models comparison metrics for the simulated DL event logs

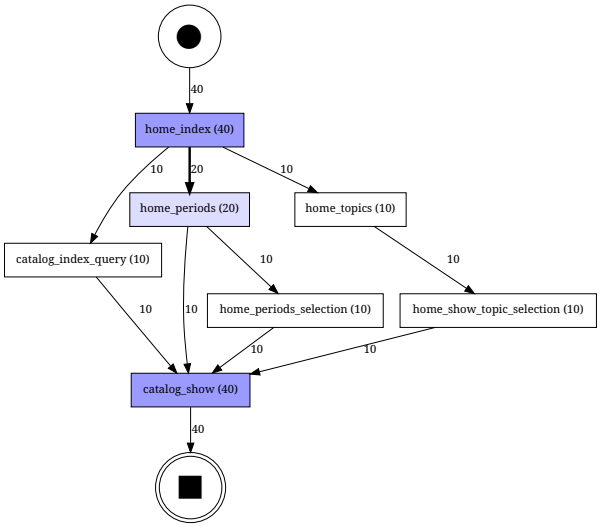

Fig. 4: Discovered DFG from the first cluster : lookup event logs

In addition to the aforementioned findings, Figure 4 provides visual evidence supporting the effectiveness of the proposed FCA clustering method in generating the expected event log clusters, namely Lookup, Borderline, and Exploratory traces. These clusters align with the outcomes mentioned in Section 6.2.

For example, the process model derived from the first cluster exhibits the capability to gather users who can access specific documents with minimal manipulation, indicating its suitability for lookup-based activities. This confirms the ability of the FCA clustering method to identify and group similar traces

based on their underlying characteristics, leading to the discovery of process models that align with the expected behavior and requirements.

7 Conclusion

This work focuses on extracting similar users' journeys in information systems with unstructured business processes. In this paper, we propose a new method to cluster traces (users interactions) in process mining using FCA. To the best of our knowledge, this work is the first process mining study processing users traces and generating models for users and tasks using FCA techniques. The FCA clustering method generates a lattice based on user traces and then uses a score function to select one concept for each trace. Results showed the effectiveness of our method for clustering users' traces and modeling their journeys. The generated models allow for the improvement of DL design by identifying unused features that may require more documentation and improving useful ones related to frequent events. Furthermore, based on the paths in models, designers can make practical recommendations to help new users achieve their goals by following the event sequences created by advanced users. For future work, it is desirable to work with a large and real dataset that covers the whole journeys of DLs uses. Also, we plan to explore more trace clustering algorithms to compare the obtained results. Furthermore, considering the temporal information in generating the lattice using the temporal sequence plugins of `GALACTIC` may be a promising approach.

8 Aknwoledgement

This article is supported by the ANR SmartFCA project Grant ANR-21-CE23-0023 of the French National Research Agency

References

1. Van der Aalst, W.: Process mining: data science in action. Springer (2016)
2. Van der Aalst, W.M., Weijters, A.J.: Process mining: a research agenda. Computers in industry **53**(3), 231–244 (2004)
3. Adriansyah, A.: Aligning observed and modeled behavior. Ph.D. thesis, Technische Universiteit Eindhoven (2014)
4. Barbut, M., Monjardet, B.: Ordres et classifications : Algèbre et combinatoire. Hachette, Paris (1970), 2 tomes
5. Bordat, J.P.: Calcul pratique du treillis de Galois d'une correspondance. Mathématiques et Sciences humaines **96**, 31–47 (1986)
6. Boukhetta, S.E., Demko, Ch., Richard, J., Bertet, K.: Sequence mining using fca and the NextPriorityConcept algorithm. In: Concept Lattices and Their Applications 2020. vol. 2668, pp. 209–222 (2020)
7. Demko, Ch., Bertet, K., Faucher, C., Viaud, J.F., Kuznetsov, S.O.: NextPriorityConcept: A new and generic algorithm computing concepts from complex and heterogeneous data. Theoretical Computer Science **845**, 1 – 20 (2020)
8. Diamantini, C., Genga, L., Potena, D.: Behavioral process mining for unstructured processes. Journal of Intelligent Information Systems **47**(1), 5–32 (2016)

9. Ferré, S.: Reconciling expressivity and usability in information access from file systems to the semantic web (2014)
10. Ferré, S., Ridoux, O.: A logical generalization of formal concept analysis. vol. 1867, pp. 371–384 (03 2000)
11. Ganter, B., Kuznetsov, S.O.: Pattern structures and their projections. In: LNCS of International Conference on Conceptual Structures (ICCS'01). pp. 129–142 (2001)
12. Ganter, B., Wille, R.: Formal Concept Analysis, Mathematical foundations. Springer Verlag, Berlin (1999)
13. Ikeda, M., Otaki, K., Yamamoto, A.: Formal concept analysis for process enhancement based on a pair of perspectives. In: CLA. pp. 59–70 (2014)
14. Jaffal, A., Le Grand, B.: Towards an automatic extraction of smartphone users' contextual behaviors. In: 2016 IEEE Tenth International Conference on Research Challenges in Information Science (RCIS). pp. 1–6. IEEE (2016)
15. Juniarta, N.: Mining complex data and biclustering using formal concept analysis. Ph.D. thesis, Université de Lorraine (2019)
16. Kaytoue, M., Codocedo, V., Buzmakov, A., Baixeries, J., Kuznetsov, S.O., Napoli, A.: Pattern structures and concept lattices for data mining and knowledge processing. In: In Proceedings of ECML-PKDDl (2015)
17. Leemans, S.J., Fahland, D., van der Aalst, W.M.: Discovering block-structured process models from event logs containing infrequent behaviour. In: International conference on business process management. pp. 66–78. Springer (2013)
18. Leemans, S.J., Poppe, E., Wynn, M.T.: Directly follows-based process mining: Exploration & a case study. In: 2019 International Conference on Process Mining. pp. 25–32. IEEE (2019)
19. Levenshtein, V.I., et al.: Binary codes capable of correcting deletions, insertions, and reversals. In: Soviet physics doklady. vol. 10, pp. 707–710. Soviet Union (1966)
20. Linding, C.: Fast concept analysis. In: Working with Conceptual Structures-Contributions to ICC. pp. 235–248 (2002)
21. Marchionini, G.: Exploratory Search: From Finding to Understanding. Commun. ACM **49**(4), 41–46 (Apr 2006)
22. Peters, E.M., Dedene, G., Poelmans, J.: Empirical discovery of potential value leaks in processes by means of formal concept analysis. In: 2013 IEEE 13th International Conference on Data Mining Workshops. pp. 433–439. IEEE (2013)
23. Saquer, J.M.: Formal concept analysis based clustering. In: Encyclopedia of Data Warehousing and Mining, pp. 514–518. IGI Global (2005)
24. Song, M., Günther, C.W., Van der Aalst, W.M.: Trace clustering in process mining. In: International Conference on Business Process Management. pp. 109–120. Springer (2008)
25. Trabelsi, M., Suire, C., Morcos, J., Champagnat, R.: A new methodology to bring out typical users interactions in digital libraries. In: 2021 ACM/IEEE Joint Conference on Digital Libraries (JCDL). pp. 11–20. IEEE (2021)
26. Trabelsi, M., Suire, C., Morcos, J., Champagnat, R.: User's behavior in digital libraries: Process mining exploration. In: International Conference on Theory and Practice of Digital Libraries. pp. 388–392. Springer (2019)
27. Wille, R.: Restructuring lattice theory : an approach based on hierarchies of concepts. Ordered sets pp. 445–470 (1982), i. Rival (ed.), Dordrecht-Boston, Reidel.
28. Zandkarimi, F., Rehse, J.R., Soudmand, P., Hoehle, H.: A generic framework for trace clustering in process mining. In: 2020 2nd International Conference on Process Mining (ICPM). pp. 177–184 (2020). https://doi.org/10.1109/ICPM49681.2020.00034