Privacy-Preserving Business Process Fragmentation for Reusability*

Mohamed Anis Zemni[1], Salima Benbernou[1], Soror Sahri[1]

[1]LIPADE, Université Paris Descartes, France
{mohamed-anis.zemni,salima.benbernou,soror.sahri}@parisdescartes.fr

Abstract. There is a growing need for the ability to fragment ones business processes in an agile manner, and be able to reuse these fragments for building new processes. Decomposition aims at clustering workflow activities into fragments according to business constraints. Additionally, individuals are becoming more and more concerned about the privacy of their personal data. In this paper, we present a fragment identification approach that is aware of privacy concern. Our fragmentation approach is based on the so-called formal concept analysis approach, while integrating a technique for avoiding the association of sensitive information.

Keywords: Privacy, Sensitive Association, Formal Concept Analysis, Galois Lattice, Business Process, Reuse, Clustering, Fragmentation.

1 Introduction

With the advent of modern Web technologies, organizations and business entities are more and more focused on improving the quality of their services. At the same time, they experience a need to maintain a high degree of efficiency, with respect to delivery deadlines and productivity; all this, within the context of a continuously increasing competition. A key strategy, companies use to cope with these pressures, consists of organizing their *Business Processes* [1] in a reconfigurable and agile manner.

Designing new processes or updating existing ones, is still a time consuming, costly and error prone activity. These same obstacles also apply to the construction of inter-organizations processes, which serve to integrate the businesses of several companies. It is more efficient to reuse existing processes or fragments of a whole business process while constructing new processes, rather than build the new process from the scratch. Nevertheless, the reuse of complete process remains difficult and may not even be interesting. In general, developers are keen to only use fragments of existing processes, constrained with fewer business rules, and integrate them to produce new processes, with different business rules and constraints. Existing literature proposes to split processes into multiple clusters of activities, in order either to enhance process execution [2, 3], or to secure the access to data that is enclosed within the process, through the

* The research leading to these results has received fund from the European Community's Seventh Framework Program FP7/2007-20013 under grant agreement 215483 (S-CUBE)

use of role hierarchies[4]. Moreover, nowadays, the individuals are becoming more and more concerned about the privacy of their personal data. A person's privacy may be seriously violated by inferring information compiled from multiple sources through the associations of some data belonging to the published fragments. Typically these sensitive associations need to be formed within process boundaries for the purpose of performing key process functions.Therefore, avoiding this problem, in the context of process fragmentation and reuse, should require, among things, a good design of the fragmentation boundaries, for the published fragments. Our specific contributions are articulated as follows:

1. We propose a model to represent processes using *Formal Concept Analysis* (FCA) [5]. This aims at splitting a process into independent fragments. Our approach is a *two-phases* fragmentation.
2. We define two types of constraints that are needed to make the fragmentation: (1) Process dependencies: They are applied on activities'dependencies to preserve the structure semantic. (2) Privacy-preserving constraints: They are applied on data dependencies to provide *privacy-preserving* fragmentation.
3. We ensure the privacy while performing the fragmentation to avoid sensitive association disclosure. Thus, the resulted fragments are privacy-preserving.

The proposed framework deals only with privacy-aware fragmentation for reusing these fragments. The reusing process is out of the scope of this paper. The rest of the paper is organized as follows. In Section 2, we propose a motivating example to introduce our approach handling privacy requirements and concerns. In Section 3, we show how to model process with FCA for the purpose of generating activity clusters. We extend FCA, through the integration of privacy-preserving constraints in Section 4. In Section 5, we provide an algorithm for grouping privacy-aware fragments, which are structurally correct. We review the related literature in Section 6. Our conclusion, as well as some indications of our future research directions are provided in Section 7.

2 Motivating Example

Our work focuses on how to fragment business processes in privacy-preserving way, for the purpose to reuse the resulted fragments. We start with an example that we will use to highlight our work throughout the paper. Suppose there are three processes: Drug Prescription, Medical Research Center and Searching for Pharmacy (cf. (Figure 1)).

The first process (1.a), which we develop throughout the rest of the paper, depicts a "Drug Prescription Process" that is part of a large automated *Medical Management* system. Activity a_1 receives patient's identifier (x). Using this identifier, activity a_2 accesses the corresponding patient's personal data, (z), consisting of the patient's name, age, ..., etc. At the same time, activity a_3 retrieves the set of ordered drugs, (y). Activity a_4 extracts the patient's medical history (h). Activity a_5 checks for any incoherences of (y) with (h) and updates the history, (h), representing the symptoms and the illnesses. Once a_5 and a_2 have completed, a_6 generates the drug prescription, (o), and issues it to the pharmacy. Activity a_7 checks whether all prescribed drugs are available or not, (v), and returns the amount of the drugs, (d). Activity a_8 computes the reimbursement

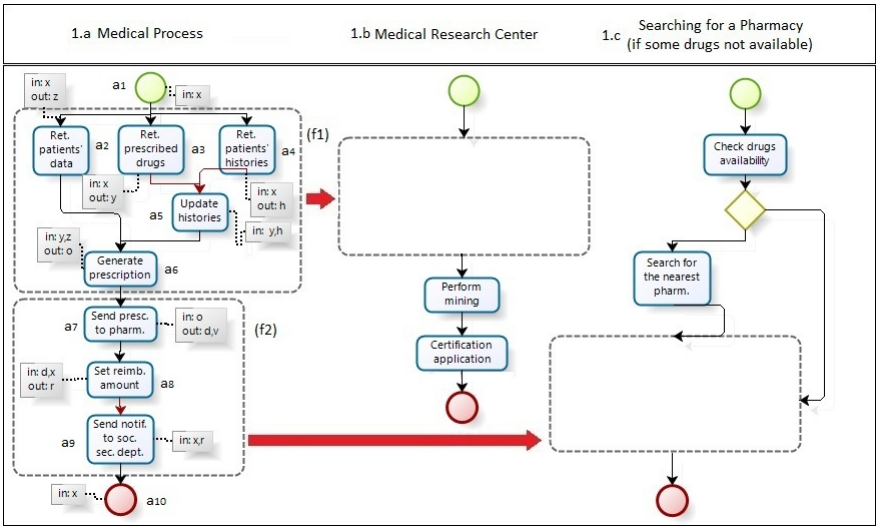

Fig. 1: Business Process Fragments Reutilization without Considering Privacy Concerns

amount, (r), based on the drugs amount, (d). Activity a_9 sends a notification to the insurance, with the reimbursement amount. Finally, activity a_{10} marks the prescription as properly processed.

The second process (1.b) represents a "Medical Research Center" collaborating with the hospital that owns the "Drug Prescription Process". The center wants to take advantage of the patients' consulting to work on to discover new drugs.

The third process (1.c) represents a "Searching for pharmacy" process. The pharmacy checks the availability of the drugs. Otherwise, searches for the nearest pharmacy having the drugs available and processes the reimbursement task.

Our objective is to split the "Drug Prescription Process" into independent and reusable fragments. One possible fragmentation could be done according to the main tasks involved in the process. Intuitively, the process can be fragmented into two main parts: (f_1) and (f_2), displayed in dashed boxes in (Figure 1) (1.a). (f_1) deals with the delivery of the prescriptions (o), and (f_2) processes the reimbursement of the prescription cost.

This fragmentation, however, while it provides coherent fragments, each processing generic functions that may be needed in other similar processes e.g. "Medical Research Process" and "Searching for a Pharmacy", is not necessarily *privacy-preserving*. Indeed, a fragment can be integrated in a new business process, linking it to a malicious activity that discloses some sensitive information. By inspection, we can easily determine that *sensitive associations* may be revealed. For instance, the association between z, the personal patient's data, and h, the patient's medical history, should not be disclosed; i.e, it should not be delivered as a single entity, as a parameter across fragment boundaries. Thus, the fragmentation process needs to be carefully designed to ensure that it satisfies similar privacy requirements apart other requirements. For this aim,

the fragmentation approach follows two phases: (1) split the process into *low-grained* privacy-preserving clusters, (2) group the resulted clusters to obtain coarse-grained and reusable fragments. This will be detailed in the following sections.

3 Modeling Processes with FCA

FCA can be used as a search space for selecting attributes to classify an object a amongst a set of objects A, and results into a set of concepts related by inclusion, each representing a cluster of objects that are semantically close w.r.t. attributes [5, 6]. This forms a concept graph with the properties of a lattice, called Galois Lattice. Objects and attributes are organized in the form of a discretized table called *formal context*. In the present work, we model processes with FCA such that the objects correspond to the activities. The attributes are represented by the activities properties e.g. data that are handled within the activities. We call these data *data items* in the rest of the paper.

In the following, we consider the example in Section 2 (1.a).

Definition 1 (Formal Context). *A Formal Context is $C = (E, A, R)$. E is a set of attributes, A is a set of activities, and $R \subseteq A \times E$, is a relation capturing the data items, used by each activity; i.e., $(e, a) \in R$, if the activity $a \in A$ uses the the data item $e \in E$. R is depicted by (x) when a formal context will be represented by a table where arrows are activities and columns are data items.*

For instance, $E = \{x, y, z, h, o, d, r, v\}$ is the set of data items and $A = \{a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}\}$ is the set of activities.

Definition 2 (Galois Correspondence). *A Galois Correspondence involves two functions, Θ and Δ, generated from a given Formal Context. Θ is defined over the subsets of activities, and gives the data item that are commonly used by them. For a given $A_i \subseteq A$, $\Theta(A_i) = \{e \in E | (a, e) \in R, \forall a \in A_i\}$. Δ is defined over the subsets of data items, and gives the activities that commonly use them. For a given $E_j \subseteq E$, $\Delta(E_j) = \{a \in A | (a, e) \in R, \forall e \in E_j\}$.*

For instance, $\Theta(\{a_1, a_2, a_3, a_4, a_8, a_9, a_{10}\}) = \{x\}$, and, $\Delta(\{y\}) = \{a_3, a_5, a_6\}$.

Definition 3 (Formal Concept/Clusters). *A Formal Concept is the couple (E_j, A_i), where $E_j \subseteq E$ is a set of data items and $A_i \subseteq A$ is a set of activities such as $\Theta(A_i) = E_j$ and $\Delta(E_j) = A_i$. A* cluster *corresponds to the set of activities A_i involved in the formal concept.*

For example, $(\{a_3, a_5, a_6\}, \{y\})$ is a formal concept, and $\{a_3, a_5, a_6\}$ is its corresponding cluster.

Definition 4 (Galois Lattice). *A Galois Lattice for a Formal Context $C = (E, A, R)$ is a pair $(\zeta(C), \leq)$, where $\zeta(C)$ is the set of concepts over C and $\leq$ is the order relation on $\zeta(C)$. For some $(E_i, A_i) \in \zeta(C)$ and $(E_j, A_j) \in \zeta(C)$, $i \neq j$, we have $(E_i, A_i) \leq (E_j, A_j)$ iff $E_j \subseteq E_i$ (equivalent to $A_i \subseteq A_j$)*

By modeling the process with a classical FCA, we aim to generate the set of clusters, $\bigcup A_i$, where $A_i \subseteq A$. These clusters will constitute parts of the resulted fragments; some of them can eventually represent the fragments themselves. Notice that until this stage of process modeling, the privacy requirements are not, yet, considered.

For instance, consider the association between the data items z, the patient's data, and h, the patient's history. We can all agree that such an association is *sensitive*, and that the history should never be tagged in an intended open context with the private data of the patient. Indeed, with the cluster $\{a_3, a_5, a_6\}$, one can disclose this sensitive association, as it encloses both of the data items z and h. Consequently, any derived fragment from this cluster can externalize this sensitive association. In this context, we propose to extend our model with constraints that can derive *privacy-preserving* fragments.

Our privacy-aware fragmentation framework for reusing will follow two phases:

- Phase I: We will define privacy-preserving constraints that will be handled by the FCA in order to generate low-grained privacy-aware clusters.
- Phase II: Since the previous phase generates low-grained clusters and in order to make fragments coarse-grained, we introduce grouping rules.

4 Privacy-Preserving Clustering

In this section, we extend our FCA model in order to handle privacy-preserving constraints. First, we will define privacy-preserving constraints in order to avoid sensitive association disclosure when grouping clusters (in phase II). Then, we incorporate them in the FCA logic.

4.1 Privacy-Preserving Constraints

As we have already tackled in the motivating example, when reusing the resulted fragments, the association between some data items of these fragments can disclose some sensitive information e.g. name/illness. To avoid such sensitive associations, we propose to incorporate privacy-preserving constraints to our fragmentation approach. We denote by C_N the set of privacy-preserving constraints within sensitive associations. Formally, a privacy-preserving constraint, that avoids a *sensitive association* between two data items, $e_i^*, e_j^* \in E^*$, is expressed as $(e_i^* \multimap e_j^*)$, e.g. $(z \multimap h)$.

Consider three clusters, $A_i, A_j, A_k \subseteq A$. We assume that A_i and A_j, use respectively the data items, e_i^* and e_j^*; and $A_i \cap A_j = \phi$ (no common activities between A_i and A_j). Let A_k can be used by a malicious part to disclose sensitive information. If ever A_i and A_j are reused in a new Business Process that already includes A_k. Then, A_k will allow to guess this sensitive association with pretty good confidence.

In order to group the resulting clusters, in phase II, without disclosing sensitive associations, we propose to combine privacy-preserving constraints such that a cluster can be directly combined with similar clusters w.r.t. E_i^*, or with clusters that use neither of the data items in E^*. For this, we introduce the *combination operator*, $\bigotimes$, to

combine privacy-preserving constraints. Therefore, given two privacy-preserving constraints, $(e_i^* \multimap e_j^*)$, and, $(e_k^* \multimap e_l^*)$,

$$\mathrm{C}_N = (e_i^* \multimap e_j^*) \bigotimes (e_k^* \multimap e_l^*) = \begin{cases} (\mathrm{e}_i^* e_k^* \multimap e_j^* e_l^*), (e_i^* e_l^* \multimap e_j^* e_k^*) & \text{,if} \quad \mathrm{e}_i^* \neq e_k^*, e_l^* \\ & \text{and} \ \ \mathrm{e}_j^* \neq e_k^*, e_l^* \\ (\mathrm{e}_i^* e_k^* \multimap e_j^* e_l^*), (e_i^* e_l^* \multimap e_j^* e_k^*) & \text{,if} \quad \mathrm{e}_1^* = e_3^* \end{cases}$$

For instance, consider, $(z \multimap h)$ and $(z \multimap v)$, two privacy-preserving constraints. Then, $(z \multimap h) \bigotimes (z \multimap v) = (z \multimap hv)$.

Consequently, in case we have many privacy-preserving constraints, we have to apply the combination operator $\bigotimes$ on them when performing the extended FCA technique. Thus, a privacy-preserving constraint, C_{N_k}, is expressed as $(E_i^* \multimap E_j^*)$, where $C_{N_k} \in C_N$.

4.2 Extending FCA with Privacy-Preserving Constraints

In the previous section, we have presented privacy-preserving constraints, C_N. In this section, we incorporate them in the FCA technique so that the relation between activities and data depends also on the data items, $E_i^* \subseteq E^*$ that the activities use. This can be formulated as follows:

Definition 5 (Privacy-Aware Formal Context). *A privacy-aware formal context is* $(E, A, C_N, R, \bigotimes)$. *If an activity* $a_i \in A$ *uses a data item* $e_j \in E$, E_k^* *is involved in the combination of the constraints* $\bigotimes_{C_N}$, *the relation is* $(e_j, a_i, E_k^*) \in R$.

Definition 6 (Privacy-Aware Galois Correspondence). *The common data used by the activities in* A_i *related to the data items* E_k^* *is defined as a function* $\Theta(A_i, E_k^*) = \{e | (e, a, E_k^*) \in R, \forall e \in E, \forall a \in A_i, \forall E_k^*$ *involved in the constraints* $C_N\}$. *Similarly, the common activities to a set of data items* E_i *related to the data items* E_k^* *is defined as a function* $\Delta(E_j, E_k^*) = \{a | (e, a, E_k^*) \in R, \forall e \in E_j, \forall a \in A, \forall E_k^*$ *involved in the constraints* $C_N\}$.

The two functions, Θ and Δ, for Privacy-Aware Galois Correspondences are executed to form Privacy-Aware Formal Concepts.

Definition 7 (Privacy-Aware Formal Concept). *A privacy-aware formal concept, related to a privacy-aware formal context* $(E, A, C_N, R, \bigotimes)$ *is a tuple* (E_j, A_i, E_k^*) *where* $E_j \subseteq E$, $A_i \subseteq A$ *is a cluster of activities and* E_k^* *is involved in the constraints* C_N *with* $\Theta(A_i, E_k^*) = (E_j)$, *and,* $\Delta(E_j, E_k^*) = (A_i)$.

All the generated privacy-aware formal concepts are organized in a tree structure, called privacy-aware Galois lattice.

Definition 8 (Privacy-Aware Galois Lattice). *A privacy-aware Galois Lattice for a privacy-aware formal context* C=$(E, A, C_N, R, \bigotimes)$ *is a pair* $(\zeta(C), \leq)$. $\zeta(C)$ *is the set of concepts generated from* C, *and* $\leq$ *is the order relation on* $\zeta(C)$ *such as* $(E_i, A_i, E_i^*) \in \zeta(C)$ *and* $(E_j, A_j, E_j^*) \in \zeta(C)$, $(E_i, A_i, E_i^*) \leq (E_j, A_j, E_j^*)$ *iff the following inclusion properties are well respected:* $E_j \subseteq E_i \subseteq E$ *(equivalent to* $A_i \subseteq A_j \subseteq A$ *and* $E_i^* \subseteq E_j^*$ *involved in the constraints* C_N).

The following example illustrates the notions related to FCA extended with privacy-preserving constraints.

Example 1. Consider the drug prescription process described in Section 2 (1.a). Its corresponding privacy-aware formal context is, $C = (E, A, C_N, R, \bigotimes)$, where, $E = \{x, y, z, h, o, d, r, v\}$, $A = \{a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}\}$, $C_N = \{(z \multimap h), (z \multimap v)\}$. First of all, we apply the combination operator $\bigotimes$ on C_N, so that, $(z \multimap h) \bigotimes (z \multimap v) = (z \multimap hv)$. Consequently, $C_N = (z \multimap hv)$. This privacy-aware formal context C is illustrated in Figure 2. The two tables represent the data items involved in the constraint set C_N relatively to sensitive associations. Table 2.a represents the relations between the activities and the data items, w.r.t. the data item z. Table 2.b represents the relations between the activities and data items, w.r.t. the data items hv. Indeed, an activity using the data item z, represented by (x) in table 2.a in figure 2, could not use the sensitive data items hv, represented by (x) in table 2.b, and vice versa. If the relation exists in the two tables, so, the relative activities are called indifferent. That means they do not use neither y nor h nor v. We assume that the privacy-preserving constraints are correct and there are no activities using r, h and v in the same time.

C	x	y	z	h	o	d	r	v
a_1	x							
a_2								
a_3	x	x						
a_4	x			x				
a_5		x		x				
a_6								
a_7					x	x		x
a_8	x					x	x	
a_9	x						x	
a_10	x							

2.a Relation A x E w.r.t. "$h\ v$ "

C	x	y	z	h	o	d	r	v
a_1	x							
a_2	x		x					
a_3	x	x						
a_4								
a_5								
a_6		x	x		x			
a_7								
a_8	x					x	x	
a_9	x						x	
a_10	x							

2.b Relation A x E w.r.t. "z "

Fig. 2: Privacy-aware Formal Context C

From the privacy-aware formal context C depicted in (Figure 2), we generate the *Privacy-Aware Formal Concepts* $\zeta(C)$. For this purpose, we use the Privacy-Aware Galois Correspondence functions. For example, $(\{xy\}, \{a_3\}, \phi)$ is a Privacy-aware formal concept according to the correspondence functions.

On the other hand, $(\{or\}, \{a_7\}, zh)$ is not a privacy-aware formal concept because $\Delta(\{a_7\}, zh) = \{xor\}$.

Figure3 illustrates the corresponding privacy-aware Galois lattice that is obtained from the generated privacy-aware formal concepts.

From this phase, phase I, we have generated privacy-aware formal concepts $\zeta(C)$ and represented them within the privacy-aware Galois lattice. The lattice properties

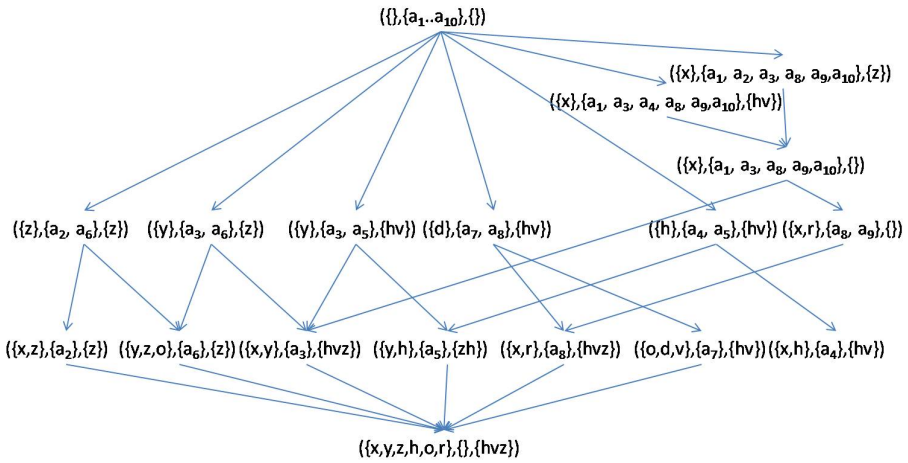

Fig. 3: Privacy-aware Galois Lattice with Privacy-aware Formal Concepts

are well respected, i.e. even if we add the privacy-awareness dimension, the inclusion properties, stated in definition 8, hold. Moreover, the privacy-preserving constraints that we have defined can be applied in some FCA algorithms available in the literature, as in [7].

From the generated concepts, we have derived clusters with no sensitive association disclosure. However, some of these clusters are not connected i.e. some of their activities are not linked. This is the case for the cluster $\{a_1, a_3, a_8, a_9, a_{10}\}$. Moreover, some other clusters are low-grained, i.e. they have no significance for reuse e.g. $\{a_3\}$. In the next section, we will focus on these problem.

5 Generating Privacy-Preserving Fragments

In this section, we will focus on phase II of our fragmentation approach. In a first time, we present process dependencies according to weak and strong links. Then, in a second time, we define rules on how to use these process dependencies in order to group clusters and obtain reusable fragments. Resulted fragments, more specifically those in conflict, should not be reused together within the same new business process.

5.1 Process Dependencies

Process dependencies are constraints applied on the link between any two given activities. We separate these dependency links into two groups: (1) weak links, and (2) strong links, called C_F. The weak links, explicitly represented with the process flows, can be identified as those that can be broken without losing the comprehensiveness of each of the activities. The strong links, however, are fixed by the user and should never be severed. In our work, we are interested in only strong links. Consequently, process dependencies correspond to only strong links that we represent as $C_F = \{(a_i, a_j)|a_i, a_j \in$

$A\}$, where a_i and a_j should never be broken. For instance, in the motivating example (1.a) of Section 2, $C_F = \{(a_3, a_5), (a_4, a_5), (a_8, a_9)\}$ (strong links are represented by red arrows). We will use these process dependencies in the next section while grouping clusters.

5.2 Grouping Clusters

During this phase, phase II, we are interested in the clusters that are derived from the Formal Concepts. We will particularly, retrieve and identify the connected clusters. Connected clusters are the clusters whose activities are linked together. Next, we organize the connected clusters into groups according to their data items E_i^*. While grouping the clusters, we apply some rules to get reusable fragments. For this aim, we propose Algorithm 1. Below, we explain the main steps of the algorithm and illustrate them using example in Figure 2 (1.a).

1. The first step consists of identifying connected clusters (line 2 in Algorithm 1). We say that a cluster is connected if there is path between each two activities. Indeed, in order to generate reusable fragments, all cluster activities should be linked between each other to be duly executed. To retrieve connected clusters (L_{CC}) among all the clusters (L_{con}) represented in algorithm 1, we adopt a bottom-up approach to navigate the privacy-aware Galois lattice. We start from the cluster in the bottom of the privacy-preserving Galois lattice and navigate to the top. As child clusters are included in the parent clusters, if ever parent clusters are connected, then their child clusters are removed from (L_{CC}) and replaced by their parent clusters. Otherwise, if parent clusters do not respect the connectivity condition then their child clusters are kept. For instance, the cluster $\{a_6\}$ is connected. Its parent clusters $\{a_6, a_7\}$ and $\{a_2, a_6\}$ are connected which is not the case for $\{a_3, a_6\}$. Thus, the child cluster $\{a_6\}$ is replaced by only its connected parent clusters $\{a_6, a_7\}$ and $\{a_2, a_6\}$ into the list L_{CC}.
2. The second step consists of organizing connected clusters into groups (line 3 in Algorithm 1). Once the connected clusters are retrieved, they are organized into groups according to their data items E_i^*.Let's call each group L_{G_i} and the set of all the groups $L_G = \bigcup L_{G_i}$. Cluster grouping results into two kinds of grouping: (i) intra-group and (ii) inter-group. *Intra-group* consists of grouping clusters in between clusters using the same data items E_i^*. *Inter-group* consists of grouping clusters in between clusters belonging to one group and clusters using neither of the data items E^*. Clusters using different data items will never be grouped together. We illustrate cluster grouping in Figure 4, using the arrows in between the different groups.
3. The third step consists of applying rules on the resulted cluster groups. Once connected clusters are retrieved and placed into groups, we enhance them with functional rules. These rules allow to keep activities as close as possible in order to get reusable fragments. We define these rules on the basis of process dependencies C_F. We distinguish between intra-group functional rules and inter-group functional rules.

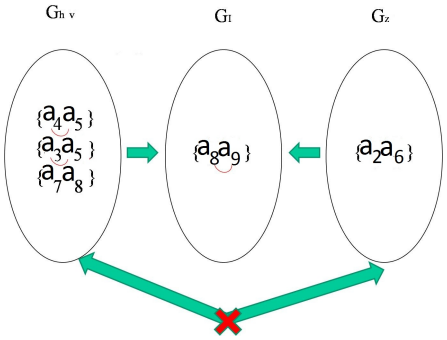

Fig. 4: Clusters grouping

- Intra-group rules (line 4-6 in Algorithm 1): This type of rules is performed only within the same group (L_{G_i}).
  - If two clusters share the same activity, they have to be grouped together. For instance, clusters $\{a_4, a_5\}$ and $\{a_3, a_5\}$ share the same activity a_5. Grouping them gives rise to the cluster $\{a_3, a_4, a_5\}$.
  - If two clusters have activities that are weakly linked in the process, they are combined together.
  - If an activity of a given cluster is close to an activity of another cluster of the same group L_{G_i}, i.e. linked via strong links C_F, they have to be combined. For instance, recall that $C_F = \{(a_3, a_5), (a_4, a_5), (a_8, a_9)\}$. The clusters $\{a_3, a_5\}$ and $\{a_4, a_5\}$ will be grouped together to generate $\{a_3, a_4, a_5\}$, because the link between activity a_3 of the first cluster and activity a_5 of the second cluster should not be broken according to C_F.
- Inter-group rules (line 7 of Algorithm 1): This type acts on the entire list of groups (L_G). Only strong links C_F promote this type of grouping. As the link between two activities in C_F cannot be broken, these activities should be merged together. Consider two different cluster groups L_{G_i} and L_{G_j}. If an activity of one cluster in L_{G_i} has strong link with an activity of another group in L_{G_j}, these two clusters have to be combined. Notice that one of the two groups should enclose clusters that use neither of the data items in E^*. For instance, the clusters $\{a_7, a_8\}$ and $\{a_8, a_9\}$ are grouped together because the link between a_8 of the cluster from the first group and a_9 of the cluster from the second group should not be broken.

The application of Algorithm 1 generates privacy-preserving and reusable fragments as shown in Figure 5 (1.a). This may not be the case through an intuitive fragmentation by inspection process, taking into account only functional constraints. Additionally, the activities involved in fragments are connected so that when these fragments are executed separately, their corresponding activities are duly and correctly performed.

Consequently, the "Medical Research Center" process can reuse the fragment f_2 and the "Searching for Pharmacy" process can use the fragment f_3 as shown in process

Algorithm 1 Grouping algorithm

Require: L_{con}:Set of Formal Concepts,C_F:Set of process dependencies

1: init L_{CC},L_G
2: $L_{CC} \leftarrow$ retrieveConnectedConcepts(L_{con})
3: $L_P \leftarrow$ splitConcepts(L_{CC})
4: **for** all L_G **do**
5: intraGroupGrouping(L_{G_i})
6: **end for**
7: interGroupGrouping(L_G, C_F)
8: **return** L_G

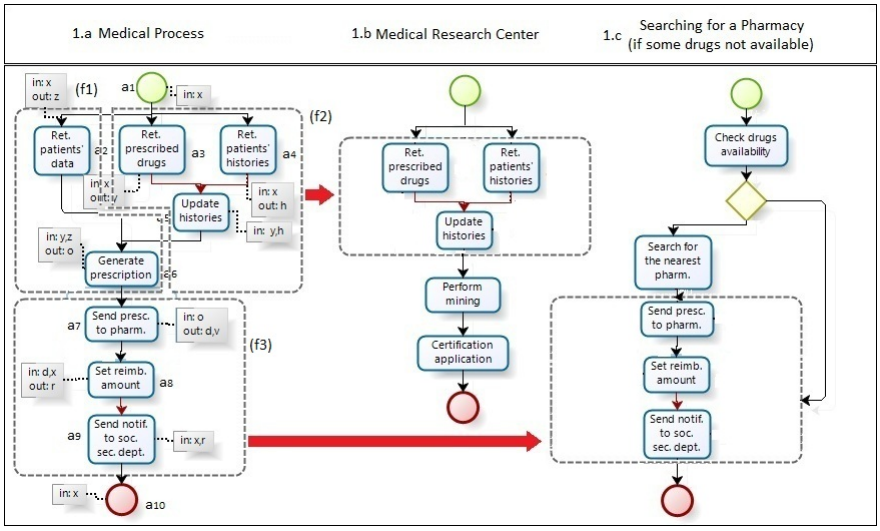

Fig. 5: Final Fragments

(1.b) and process (1.c) in Figure 5. Indeed, the sensitive associations between z and h and between z and v are avoided.

6 Related Work

Business process decomposition has been studied in [2, 3, 8]. In [2], a BPEL decomposition process results into several partitions assigned to different process engines. The partitions are defined at the design stage of a process where the activities are assigned to the execution sites manually. The aim of such approach is to enhance the execution of the original process. Similarly, an automatic approach is proposed in [3], to partition a centralized BPEL process into a set of coordinated processes. To do so, the authors foster the idea of decentralized orchestration and use an explicit meta-model and graph transformation system to supply the formal grounding to slice a single BPEL process. Then, they add the communication infrastructure among the newly created processes to

exchange data and propagate the execution flow.The idea of distributed processes execution has also been tackled for workflows. In [8], the authors separate one integrated workflow model into small partitions and allot them to different servers to be executed. Although these papers address business process fragmentation, reusing the resulted process fragments with privacy guarantee during fragmentation are not considered.

Writing processes or process fragments that can be reused in different places within a process or multiple processes has been considered in [9]. In this aim, a resolution of extending BPEL to support modularization and reuse has been proposed by using two different approaches: macros in WS-BPEL and the WS-BPEL on Event structure. [10] proposes a lifecycle model for reusing process fragment in business process modeling. The life cycle model consists of a set of phases. Each phase has one or more purposes that guide the business user in understanding and adopting the concepts of using process fragment in business process modeling. In [11], the authors present an approach based on p-calculus and ontologies to enhance business process description. The focus of this approach is using ontologies to specify business knowledge for better manipulating and reusing. The related work analyzed above focuses on reusing process fragments but none of them focuses on privacy concerns while decomposing and reusing process fragments.

Privacy has been studied in various contexts: data mining, social networks, statistical databases, etc. However, there has been very little investigation on privacy in business process area. In [4], the authors propose a fragment identification approach for workflows, based on a set of predefined policy constraints. With such approach, one can assign activities to a fragment by interpreting the sharing information obtained from the analysis according to the predefined policy constraints. In [12], there is an extension of workflows management systems to enforce data privacy by introducing the subject notion. Indeed, workflow management systems often process sensitive data related to subjects who demand that their data is properly protected. The proposed extension, in [12], allows workflows to be aware of the data subject's identity and consequently to retrieve the subject's privacy policy using a workflow data pull pattern. In [13], the authors investigate in scientific workflows where privacy concerns are particularly acute. They focus on the problem of preserving the privacy of module functionality in a workflow, i.e. the mapping between input and output values produced by the module. They mainly formalize the privacy concerns. [14] completes the work of [13] and proposes private module functionality based on the notion of l-diversity known in databases area. They abstractly model a workflow by a relation R and a privacy requirement. The owner of the workflow decides which data to hide, and provides the user with a view R which is the projection of R over attributes which have not been hidden.

The focus of almost all these works has been set on ensuring privacy in workflows and none of them addresses in particular business process fragmentation with preserving privacy to get reusable fragments. Our work is the first, to the best of our knowledge, to address this problem.

7 Conclusion

We have presented a privacy-aware framework for decomposing processes into multiple and independent privacy-preserving fragments for reusing. Our approach is based on the so-called formal concept analysis FCA. We also extended it with privacy-preserving constraints to avoid sensitive data association disclosure. As a consequence, the resulting fragments respect the given privacy requirements. Moreover, the resulting fragments can be reused without representing a risk for the disclosure of sensitive data.

As perspectives, our ongoing work is to automate the process dependency generation according to process flow logic. Moreover, our future work shall take into account the process activities that are omitted during cluster grouping.

References

1. F. Leymann and D. Roller. Business processes in a Web services world. *A quick overview of BPEL4WS. Retrieved*, pages 02–28, 2005.
2. Rania Khalaf and Frank Leymann. E role-based decomposition of business processes using bpel. In *ICWS*, pages 770–780, 2006.
3. Luciano Baresi, Andrea Maurino, and Stefano Modafferi. Towards distributed bpel orchestrations. *ECEASST*, 3, 2006.
4. Dragan Ivanovic, Manuel Carro, and Manuel V. Hermenegildo. Automatic fragment identification in workflows based on sharing analysis. In *ICSOC*, volume 6470 of *Lecture Notes in Computer Science*, pages 350–364, 2010.
5. Bernhard Ganter and Rudolf Wille. *Formal concept analysis - mathematical foundations*. Springer, 1999.
6. S. Decker H.G. Kim H.L. Kim, J.G. Breslin. Mining and representing user interests: the case of tagging practices. In *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans*, 2011.
7. P. Njiwoua and E.M. Nguifo. A parallel algorithm to build concept lattice. In *Proceedings of the 4th Groningen International Information Technology Conference for Students*, volume 107, 1997.
8. Wei Tan and Yushun Fan. Dynamic workflow model fragmentation for distributed execution. *Computers in Industry*, 58(5):381–391, 2007.
9. I. Trickovic. Modularization and reuse in ws-bpel. *SAP Developer Network*, 2005.
10. Z. Ma and F. Leymann. A lifecycle model for using process fragment in business process modeling. In *BPMDS'08: Proceeding of the 9th Workshop on Business Process Modeling, Development, and Support*, 2008.
11. Ivan Markovic and Alessandro Costa Pereira. Towards a formal framework for reuse in business process modeling. In *Business Process Management Workshops*, pages 484–495, 2007.
12. B. Alhaqbani, M. Adams, C. Fidge, and A.H.M. ter Hofstede. Privacy-aware workflow management. *BPM Center Report BPM-09-06, BPMcenter. org*, 2009.
13. Susan B. Davidson, Sanjeev Khanna, Val Tannen, Sudeepa Roy, Yi Chen, Tova Milo, and Julia Stoyanovich. Enabling privacy in provenance-aware workflow systems. In *CIDR*, pages 215–218, 2011.
14. S. Davidson, S. Khanna, T. Milo, D. Panigrahi, and S. Roy. Provenance views for module privacy.