Activity-Oriented Clustering Techniques in Large Process and Compliance Rule Repositories

Stefanie Rinderle-Ma[1], Sonja Kabicher[1], Linh Thao Ly[2]

[1]University of Vienna, Austria
Faculty of Computer Science, Workflow Systems and Technology Group
{stefanie.rinderle-ma,sonja.kabicher}@univie.ac.at
[2]Ulm University, Germany
Institute of Databases and Information Systems
thao.ly@uni-ulm.de

Abstract. Organizations often have to deal with large collections of business process models and compliance rules. Particular challenges in this context are compliance checks, consistency checks, and the maintenance of the process and rule repositories. In case that a-priory knowledge about dependencies within the process base and the rule base is not available, compliance checking must be performed by verifying all rules for each process, which turns out to be very costly in a context of large process and rule repositories. In this paper we present activity-oriented clustering techniques for efficient compliance checking which are particularly applicable in process and rule repositories where no a-priori clustering is considered. Further it is shown how the proposed clustering techniques influence the complexity of consistency checks. Finally, qualitative and quantitative aspects of the presented clustering techniques are discussed. The techniques provide a first step to effective and efficient management of large business process and compliance rule repositories.

1 Introduction

Recently business process compliance has gained particular interest: enterprises are more and more forced to guarantee that their business processes are executed in accordance with certain compliance rules such as policies, regulations, or guidelines (e.g., Sarbanes-Oxley Act or Six Sigma). Hence several approaches to design, integrate, and verify compliance rules over business processes have been proposed, e.g., [1]. However, none of these approaches paid attention to the existence of large process and rule repositories, even though several case studies show, that the amount of business processes can reach from a small set to hundreds of business processes being subject to several hundred compliance rules [2]. This demands for effective and efficient mechanisms to manage and maintain process models, compliance rules, and their interconnections. Specifically, efficiency is important since verifying compliance of process models with imposed compliance rules as well as consistency checks within the compliance rule base are often complex and expensive. Hence, in this paper we address the following research questions:(a)How to determine and manage the interconnections between process models and compliance rules in an effective and efficient manner?, (b)How to accelerate compliance as well as consistency checks?, and (c) How to support the maintenance of compliance rule repositories?

Intuitively, instead of checking compliance of all process models for all compliance rules in the repository, it might be more effective to check only those process models for which the compliance rules are relevant. This *clustering* of compliance rules is already provided by approaches that model compliance rules for business process in a policy-oriented way [3]. The question remains whether we can find a clustering if no a-priori knowledge is available. Furthermore, it is necessary to evaluate the application of clustering techniques for compliance rule and business processes (only apply clustering if beneficiary!).

In this paper we present activity-oriented clustering techniques for compliance rules and process models. These techniques can be applied independently of any a-priori knowledge such as policies associated to compliance rules and independently of any process meta model. We discuss the effectiveness of the different techniques based on performance considerations as well as on their effects on compliance rule consistency and maintenance. Exemplarily, for conflict-freeness of the compliance rule base we introduce a theorem that reduces the number of necessary consistency checks. The techniques are illustrated based on the IT Baseline Security use case as well as evaluated in a quantitative and qualitative way. The presented techniques provide a first step towards effective and efficient management of large business process and compliance rule repositories. Since we cluster compliance rules and process models, in this paper we use the term clustering instead of indexing. However, the clustering techniques could be also combined with further modeling approaches.

2 Use Case and Background Information

In this section the use case 'IT baseline security' [4] is presented and serves in the following sections as exemplification of the basic concepts and the techniques presented in this paper. Assume that the organization ORG works with business process models stored in process repository, and a number of compliance rules that affect the execution of the process models and which are stored in a rule repository. In Fig. 4, ORG's business process repository includes six business processes that refer to password protection (P1), screen lock protection (P2), protection against internet services (P3), malware scan of the data base (P4), malware scan of outgoing data (P5), and malware scan of incoming data (P6).

In this paper, we do not restrict our considerations to a certain process meta model or language. Hence, we introduce process models based on the set of activities N and set of edges E they consist of, i.e., a process model P is defined as $P := (N, E)$. To each node $n \in N$ either an activity type AT from the domain of interest $\mathcal{A}$ or a connector type $CT \in \{ANDSplit, ANDJoin, XORSplit, XORJoin\}$

is assigned to[1]. Thus, a node is either an element within the process graph and the activity type defines which activity is invoked at this point or based on the node and its connector a certain process pattern is defined. Note that in this paper we abstract from data flow issues and leave this to future work.

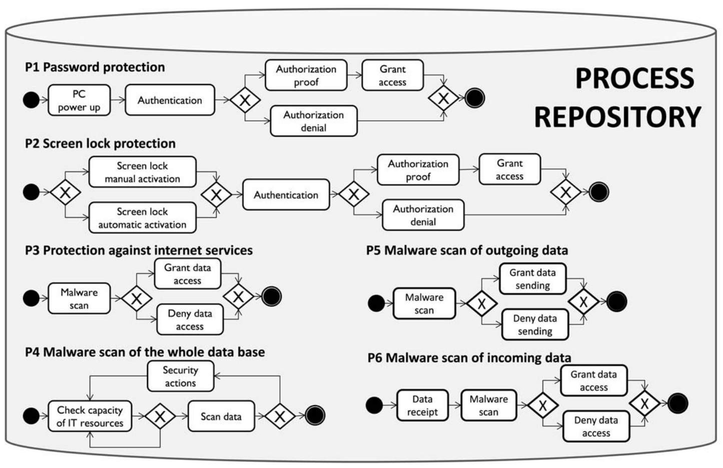


Fig. 1. IT Baseline Security: Process Models (in BPMN Notation)

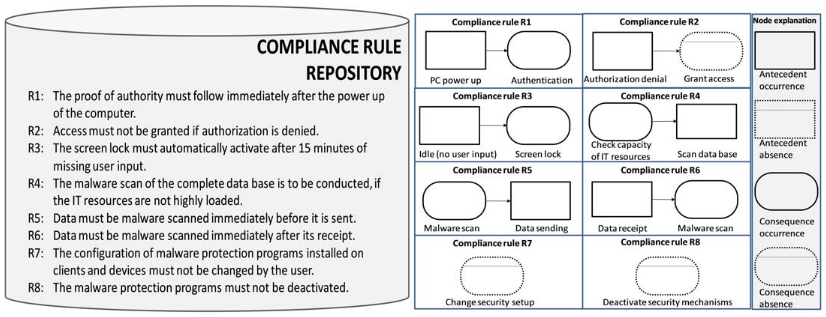
Furthermore, there are eight compliance rules stored in the ORG's compliance rule repository, as illustrated in Fig. 2. Compliance rules are visualized as compliance rule graphs (CRGs) introduced in the SeaFlows approach [5]. Note that we use the SeaFlows formalism in this paper since due to the set-based definition of the CRGs (cf. Def. 1) it can be easily determined whether a compliance rule refers to a process model or not.

Definition 1 (Compliance Rule Graph (CRG)). *A compliance rule graph is a 7-tuple $R = (N_A, N_C, E_A, E_C, E_{AC}, nt, p)$ where:*

- N_A *is a set of nodes of the antecedent graph of R,*
- N_C *is a set of nodes of the consequence graph of R,*
- E_A *is a set of directed edges connecting nodes of* N_A*,*
- E_C *is a set of directed edges connecting nodes of* N_C*,*
- E_{AC} *is a set of directed edges connecting nodes of the antecedent and the consequence graph of R,*
- $nt : N_A \cup N_C \rightarrow \{ANTEOCC, ANTEABS, CONSOCC, CONSABS\}$ *is a function assigning a node type to the nodes of R, where*
- $ANTEOCC/ANTEABS$ *denotes occurring/absent antecedent nodes in CRG,*
- $CONSOCC/CONSABS$ *denotes occurring/absent consequence nodes in CRG,*
- *p is a function assigning a set of properties (e.g., activity type, data conditions) to each node of R.*

[1] CT might be extended by further connector types such as *ORSplit*.

Basically, each CRG is built by an antecedent and a consequence pattern where the antecedent pattern might also be empty. Antecedent patterns can be composed from occurrence nodes defining the occurrences of activity executions that activate the compliance rule. Compliance rule R1 (cf. Fig. 2), e.g., is activated by the occurrence of an activity execution associated to the activity type `PC Power up`. The antecedent pattern may also consist of absence nodes defining the absence of particular activity executions. This allows for refining the occurrence pattern by putting additional conditions on the absence of activity executions (e.g., to express patterns such as "if no malware scan is conducted between data receipt and data access"). According to Def. 1 a compliance rule is activated if either the antecedent is empty or if the antecedent of a compliance rule applies. In both cases, one of the rule's *consequence patterns* must also apply in order to satisfy the rule. Each consequence pattern, in turn, may consists of occurrence as well as absence nodes and corresponding relations. Compliance rule R2 (cf. Fig. 2), e.g., has a consequence absence node in its consequence part demanding for the absence of activity `Grant access`. The pattern-based design of a compliance rule is visualized as the Fig. 2 shows. Though the compliance rules of our example are quite simple, it has been shown in [5] that more complex compliance rule patterns can be composed easily using the CRG formalism.


COMPLIANCE RULE REPOSITORY

R1: The proof of authority must follow immediately after the power up of the computer.
R2: Access must not be granted if authorization is denied.
R3: The screen lock must automatically activate after 15 minutes of missing user input.
R4: The malware scan of the complete data base is to be conducted, if the IT resources are not highly loaded.
R5: Data must be malware scanned immediately before it is sent.
R6: Data must be malware scanned immediately after its receipt.
R7: The configuration of malware protection programs installed on clients and devices must not be changed by the user.
R8: The malware protection programs must not be deactivated.

Fig. 2. Use Case - compliance rule repository (left)/compliance rule graphs (right).

The formal semantics of a structural compliance rule is based on the corresponding First Order Logic (FOL) formula. The connection between compliance rule (graphs) and process models is accomplished by interpreting the rules over the execution traces that can be produced on a process model. Execution traces are a well-known concept of capturing process instances created, started, and executed over a process model. The benefit of exploiting execution traces is that this information is completely independent of any process meta model.

Definition 2 (Interpretation of compliance rules). *Let Σ_P be the set of all execution traces of process model P (i.e., all traces P is able to produce). Then, the satisfaction of a compliance rule c over P is defined as:*

$P \models c \leftrightarrow \forall \sigma \in \Sigma_P$ holds $\sigma \models c$ based on the interpretation of the FOL formula of c.

For process P1, e.g., $\Sigma_{P1} = \{$`<PC Power up, Authentication, Authorization proof, Grant access>, <PC Power up, Authentication, Authorization denial>`$\}$. Obviously, for all $\sigma \in \Sigma_{P1}$: $\sigma \models R1$ holds, i.e., `PC Power up` is followed by `Authentication` $\forall\ \sigma\ in\ \Sigma_{P1}$.

3 Activity-Oriented Clustering Techniques

In this section we will present activity-oriented clustering techniques for process model and compliance rule repositories. The techniques will be discussed along the effort of creating clusters, the cost reduction for compliance checks and their impact on process model as well as compliance rule maintenance. As a base line for comparison, the effort for compliance checking without applying any clustering and indexing techniques (cf. Fig. 3a) turns out as

$$O(|\mathcal{C}| * |\mathcal{P}| * CE_{max})$$

for set of process models $\mathcal{P}$, set of compliance rules $\mathcal{C}$, maximum compliance checking effort CE_{max} $\forall\ P\ \in\ \mathcal{P}$, $\forall\ C\ \in\ \mathcal{C}$

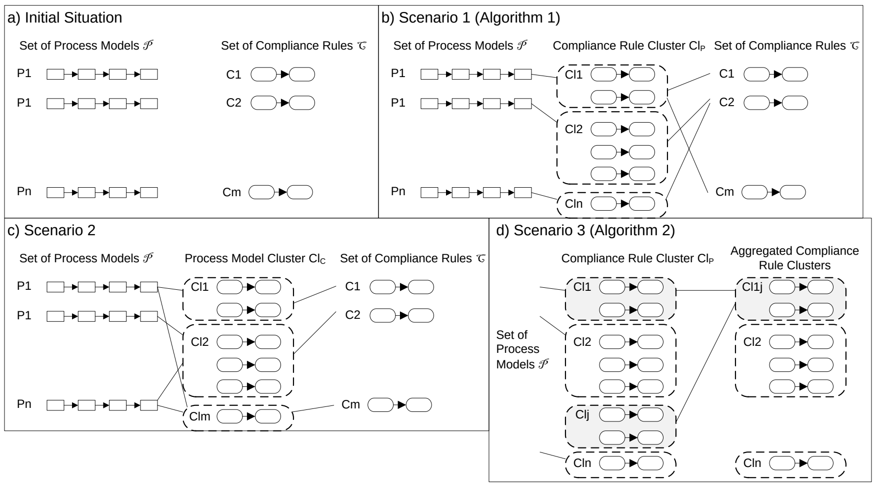

a) Initial Situation
Set of Process Models 𝒫
Set of Compliance Rules 𝒞
P1
P1
Pn
C1
C2
Cm
b) Scenario 1 (Algorithm 1)
Compliance Rule Cluster Cl_P
Cl1
Cl2
Cln
c) Scenario 2
Process Model Cluster Cl_C
Clm
d) Scenario 3 (Algorithm 2)
Aggregated Compliance Rule Clusters
Set of Process Models 𝒫
Cl1j
Clj

Fig. 3. Basic Clustering Scenarios

Without any further knowledge provided by clustering or indexing techniques (semantic or activity-oriented ones), every compliance rule has to be verified for every process model. For the structural compliance rules considered in this paper, all compliance checks can be decided at design time. However, for data-aware [6] or time-aware compliance rules certain compliance checks are to be postponed to runtime [7]. Then clustering techniques become even more favorable, including the information on design and runtime verification.

Depending on the cardinalities of $\mathcal{C}$ and $\mathcal{P}$, the effort of $O(|\mathcal{C}| * |\mathcal{P}| * CE_{max})$ might be not that dramatic. The potential performance bottlenecks more likely

arise from the effort of compliance checking CE_{max}. For checking compliance verification, most approaches adopt model checking techniques, e.g. based on LTL. These techniques require the transformation of process model and compliance rule into a state-transition system that has to be verified (state explosion problem). Minimizing the number of compliance checks to the absolutely necessary ones is a promising way to keep compliance checking effort under control.
Scenario 1: Activity-oriented Compliance Rule Clustering determines all compliance rules that are to be checked for each process model. This clustering could be already given by a semantic clustering based on a policy-oriented modeling of the compliance rules as proposed in [3]. If no semantic clustering is provided, the connection between compliance rule and process model can be determined in an activity-oriented way (at the moment abstracting from other process aspects such as data) as follows: According to Def. 1 a compliance rule is triggered over a process model, if the antecedent pattern of the compliance rule is potentially activated. This holds true if all activities associated with antecedent occurrence nodes of a compliance rule are contained in a process model. In general, this criterion can be used for optimization of compliance checks, e.g., as pre-selection before applying model-checking based techniques. Note that compliance rules that are not associated with any process model are "collected" in complementary cluster Cl_{comp}. Based on the set-oriented definition of compliance rules and process models we can define the following function IsTriggered (P,C) for a process model P = (N,E) and compliance rule C=(N$_A$,...) as follows:

$IsTriggered : \mathcal{P} \times \mathcal{C} \rightarrow \{0, 1\}$

$$\text{IsTriggered(P,C)} := \begin{cases} 1 & \text{if}(\{n \in N_A \mid nt(n) = ANTEOCC\} = \emptyset) \vee \\ & (\{n \in N_A \mid nt(n) = ANTEOCC\} \subset N) \\ 0 & \text{otherwise} \end{cases}$$

Using function $IsTriggered$ clustering of process models and compliance rules can be easily determined based on Algorithm 1.

Algorithm 1 Activity-oriented Compliance Rule Clustering

```
Require: $\mathcal{P}, \mathcal{C}$
Ensure: $Cl_P := \emptyset \; \forall \; P \in \mathcal{P}$, Cl$_{comp} := \emptyset$
  for all $P = (N, E) \in \mathcal{P}$ do
    for all $C = (N_A, N_C, E_A, E_C, E_{AC}, nt, p) \in \mathcal{C}$ do
      if $IsTriggered(P, C) = 1$ then
        $Cl_P := Cl_P \cup \{C\}$
      end if
    end for
  end for
  for all $C \in (\mathcal{C} \setminus \bigcup_P Cl_P)$ do
    $Cl_{comp} := Cl_{comp} \cup \{C\}$
  end for
  return Clustering $Cl_P$, $Cl_{comp}$
```

Applying Algorithm 1 to our use case results in the clusters depicted in Fig. 4. Note that compliance rules R7 and R8 are contained within every cluster since their antecedent pattern is empty and thus they are activated for every process model. The number of necessary compliance checks is reduced from 48 to 19.

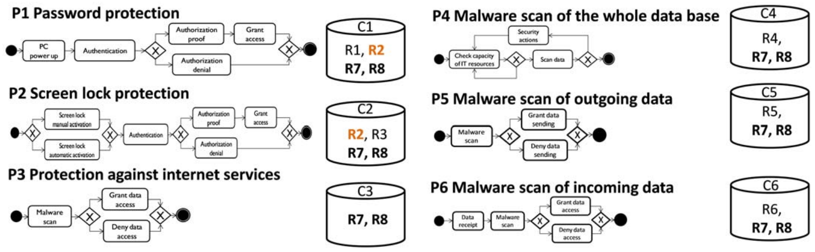


Fig. 4. Use Case IT baseline security - activity-oriented compliance rule clustering.

The complexity of Algorithm 1 is $O(|\mathcal{P}| * |\mathcal{C}|)$ which has to be considered as initial effort for clustering, i.e., the effort typically occurs once. The effort for compliance checking can be determined as

$$O(\Sigma_P |Cl_P| * CE_{max}) \ \leq \ O(|\mathcal{P}| * |\mathcal{C}| * CE_{max})$$

This means that each process model has to be checked for the compliance rules contained within the associated cluster. Based on the "clustering degree" of the clustering the reduction in effort might be significant. In the worst case, no clustering is achieved, i.e., all compliance rules refer to all process models. In this case the effort for compliance checking remains the same as the effort without applying clustering techniques. When comparing effort for compliance checking and effort for building up the clustering we obtain the following conclusion:

$$O(|\mathcal{C}| * |\mathcal{P}|) + O(\Sigma_P |Cl_P| * CE_{max}) \leq O(|\mathcal{C}| * |\mathcal{P}| * CE_{max})$$

The effect of clustering on maintaining compliance rule and process model repositories will be discussed in Section 5.

Scenario 2: Compliance Checking With Process Model Clustering can be conducted inversely to Algorithm 1: process models could be clustered for each compliance rule in $\mathcal{C}$ resulting in clusters $Cl_C \ \forall \ C \ \in \ \mathcal{C}$. Again the membership within a cluster can be determined by evaluating **Cond** set out in Algorithm 1. The complexity results again in $O(|\mathcal{P}| * |\mathcal{C}|)$. Effort for compliance checking can be determined as $\Sigma_C |Cl_C| \ \leq \ |\mathcal{P}| * |\mathcal{C}|$. Due to space limitations we omit further discussion of Scenario 2.

Scenario 3: Aggregated Rule Clustering addresses the question whether the results of Algorithm 1 could be still optimized by aggregating clusters. Cl_{P1} and Cl_{P2}, e.g., both contain rule R2 (cf. Fig. 4). Hence it could be considered to aggregate those clusters as well as the associated process models. The decision to aggregate can only be answered by evaluating the trade-off between the benefit of reducing the number of clusters and the potential performance penalty by increasing the number of unnecessary compliance checks. Figure 5 depicts different relations between two clusters Cl_{P1} and Cl_{P2}.

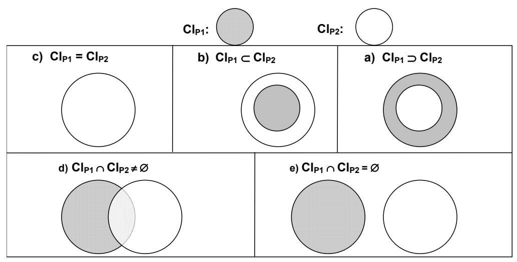

Cl_{P1}:
Cl_{P2}:
c) $Cl_{P1} = Cl_{P2}$
b) $Cl_{P1} \subset Cl_{P2}$
a) $Cl_{P1} \supset Cl_{P2}$
d) $Cl_{P1} \cap Cl_{P2} \neq \emptyset$
e) $Cl_{P1} \cap Cl_{P2} = \emptyset$

Fig. 5. Possible Relations between Compliance Rule Clusters

In case a) both clusters are equal, meaning that all of the compliance rules contained within the clusters refer to process models P_1 and P_2. By merging compliance rule clusters Cl_{P1} and Cl_{P2} into one cluster, the number of clusters is reduced by one and there is no additional effort for any of both process models P_1 and P_2. Thus in case a) cluster aggregation is advisable. In all other cases, the number of clusters will be also reduced by one, but at the expense of additional (unnecessary) compliance checks: either for P_1 against Cl_{P2} (b) or P_2 against Cl_{P1} (c) or both (d+e). The maximum number of unnecessary checks will arise in case e. However, to decide on the question whether cluster aggregation is beneficial or not, additional information is needed, e.g., on the similarity of process models. However, we leave these considerations to future work and present Algorithm 2 that aggregates two clusters only if they are equal.

Algorithm 2 Aggregated Rule Clustering

```
Require: $\mathcal{P}$, $\mathcal{C}$, $Cl_P$ (cf. Algorithm 1)
Ensure: $\mathcal{P}' = \mathcal{P}$, $Cl_{P_{i,j}} = \emptyset$
  for all $Cl_{P_i}$, $Cl_{P_j}$ with $Cl_{P_i} = Cl_{P_j}$ do
    $Cl_{P_{i,j}} := Cl_{P_i} \cup Cl_{P_j}$
    remove $Cl_{P_i}$, $Cl_{P_j}$
    $Cl_{P_{i,j}} := \{P_i\} \cup \{P_j\}$
    $\mathcal{P}' := \mathcal{P}' \setminus (\{P_i\} \cup \{P_j\})$
  end for
  return Clustering $Cl_i$, $Cl_{P_{i,j}}$, $\mathcal{P}'$
```

4 Clustering Effects on Maintenance Issues

There are several reasons for providing a cluster structure on compliance rules and process models. One reason is maintenance of rule and models. Every time a new compliance rule is added to the rule base, or fragments of compliance rule bases are merged, consistency checks of the resulting base becomes inevitable. Different approaches for checking knowledge base consistency exist, e.g., [8, 9].

Common consistency problems are caused by redundant, conflicting, subsumed, and circular rules. Further there might be knowledge gaps resulting from missing, unreachable or dead-end rules [9]. In this paper, we want to investigate the question: how do the proposed clustering techniques influence the complexity of such consistency checks. As a first step, we claim that compliance rule sets must be *conflict-free.* Formally:

Definition 3 (Conflict-free Compliance Rule Set). *Let $\mathcal{C}$ be a set of compliance rules that are imposed on a set of process models $\mathcal{P}$. Then we denote $\mathcal{C}$ as conflict-free, i.e.,*
$cf(\mathcal{C})=TRUE \iff \bigwedge\ FOL_C$ *is satisfiable* $\forall\ C$ *in* $\mathcal{C}$

Assume now that for $\mathcal{C}$ with $cf(\mathcal{C})$=TRUE, compliance rule C_{new} is added. Modifying an existing rule C to C' can be treated analogously. Without applying clustering techniques, the effort for checking conflict-freeness of $\mathcal{C}\ \cup\ \{C_{new}\}$ turns out as $O(|\mathcal{C}| * maxSat)$ where $maxSat$ denotes the maximum effort for checking satisfiability of C_{new} and $C\ \in\ \mathcal{C}$. In addition C_{new} has to be checked for compliance $\forall\ P\ \in\ \mathcal{P}$. Again clustering supports reduction of effort. In case a compliance rule is added, we do not have to check all other compliance rule whether they conflict with the new rule or not, but restrict consistency checks to the clusters C_{new} will be added to:

Proposition 1 (Conflict Checking for Compliance Clusters). *Let $\mathcal{C}$ be a set of compliance rules that are imposed on a set of process models $\mathcal{P}$ and let $cf(\mathcal{C})$=TRUE. Let further Cl_P be a clustering of $\mathcal{C}$ over $\mathcal{P}$. Assume that a new rule C_n is added to $\mathcal{C}$ and consequently added to clusters $Cl_{P_1}, \ \ldots\ , Cl_{P_k}$, $Cl_{P_i}\ \in\ Cl_P$, $i\ =\ 1,\ \ldots,\ k$.*[2] *Then:*
$cf(\mathcal{C}\ \cup\ \{C\})=TRUE \iff \forall\ i : cf(Cl_{P_i}\ \cup\ \{C\})=TRUE$

Proof. "$\Longrightarrow$": $cf(\mathcal{C}\ \cup\ \{C\})$=TRUE $\Longrightarrow \forall\ i : cf(Cl_{P_i}\ \cup\ \{C\})$=TRUE
Follows directly from $\mathcal{C} = \bigcup_P\ Cl_P$.
"$\Longleftarrow$": $\forall\ i : cf(Cl_{P_i}\ \cup\ \{C\})$=TRUE $\Longrightarrow cf(\mathcal{C}\ \cup\ \{C\})$=TRUE
Proof by contradiction:
Contradictory assumption: $\exists\ i$ with $cf(Cl_{P_i}\ \cup\ \{C_n\})$ = FALSE
$\overset{Cl_{P_i}\subseteq\mathcal{C}}{\Longrightarrow}\ cf(\mathcal{C}\ \cup\ \{C_n\})$ = FALSE
$\Longrightarrow$ contradiction $\square$

5 Discussion

In this section we sketch a simulation approach to quantitatively assess the application of clustering techniques for process models and compliance rule repositories. Further we discuss qualitative aspects in this context.

[2] Adding C_n to corresponding clusters results in $O(|\mathcal{P}|)$.

5.1 Quantitative Discussion

The quantitative evaluation of applying clustering techniques can be simulated based on the following parameters:

- sizes of compliance rule and process model sets $\mathcal{C}$ and $\mathcal{P}$
- $\Longrightarrow |\mathcal{P}|$ clusters exist after applying clustering
- clustering degree cd with $cd \in [0..1]$

The clustering degree reflects the percentage of compliance rules that are contained within exactly one cluster. Hence, $(1-c) * |\mathcal{C}|$ compliance rules are contained in several clusters. In worst case, all $(1-c) * |\mathcal{C}|$ compliance rules are contained within all $|\mathcal{P}|$ clusters, resulting in $(1-c) * |\mathcal{C}| * |\mathcal{P}|$ (compliance/consistency) checks. Consequently, the overall number of checks results in

$$(1-c) * |\mathcal{C}| * |\mathcal{P}| + c * |\mathcal{C}| \ = \ |\mathcal{C}| * |\mathcal{P}| \ - \ c * (|\mathcal{C}| * |\mathcal{P}| \ - \ |\mathcal{C}|) \ := \ f(c)$$

For $c \in [0..1]$, function $f(c)$ is falling in a linear way between maximum value of $f(0) = |\mathcal{C}| * |\mathcal{P}|$ and a minimum value of $f(1) = |\mathcal{C}|$. For $c = 0$ all compliance rules are contained within all clusters (in fact resulting in no clustering at all) with maximum number of compliance checks $|\mathcal{C}| * |\mathcal{P}|$. If all compliance rules are completely clustered in the sense that every compliance rule is only contained within exactly one cluster, only $|\mathcal{C}|$ compliance checks become necessary. The reduction in this case is $|\mathcal{C}| * |\mathcal{P}| \ - \ |\mathcal{C}|$.

In this paper, only a first simple simulation scenario is presented. However, from this starting point, different extensions are possible, e.g., by incorporating probability distributions over the number of compliance rules contained within the different clusters. Further, $f(c)$ only reflects the potential decrease in the number of required checks. A more detailed discussion on decrease efforts will be provided in future work.

5.2 Qualitative Discussion

On top of the effort considerations, clustering can be of help for maintaining compliance rule sets. By applying Algorithm 1 (or 2 respectively), the set of compliance rules that do not refer to any process model are filtered out. Reason for such "orphaned" compliance rules might be the continuous evolution of the compliance rule set. The other way round, we can also detect which process models are not subject to any compliance rule. Finally, by aggregating compliance rule clusters as done in Algorithm 2 might yield interesting results, depending on the aggregation strategy. Recall that the presented algorithm only aggregates equal clusters. However, depending on the cluster relation (cf. Fig. 5) other strategies might be pursued. In any case, if clusters can be aggregated for several process models, this might also point to the existences of similar processes or process families. Summarizing, clustering contributes to the quality of compliance rule and process model sets (repositories) in the following ways:

- decreased effort for compliance checks and maintenance
- filtering out orphaned or outdated rules (cf. Cl_{comp} in Alg. 1)

– filtering out process models that are not subject to any compliance rules
– finding process similarities with respect to the imposed compliance rules

6 Related Work

For querying large process repositories, query languages on process models have been developed [10, 11, 12]. BPMN-Q [11], e.g., is a graph-based language for querying process models. A process model will be contained in the result set of a BPMN-Q query if the query graph matches the process graph. In the context of compliance checking, BPMN-Q can be used to query process model repositories for those process models containing activities or structures that are relevant to a compliance rule [13]. Hence, finding associated process models for compliance rules as necessary for clustering can be supported by such query languages, particularly in combination with sophisticated platforms for large process repositories such as APROMORE [14]. Another current stream of research deals with the efficient evaluation of queries on process model repositories. For this, indexing techniques on process models have been developed [15]. As stated above, these indexing techniques can be applied to support the efficient finding of associations between process models and compliance rules. However, approaches for clustering and indexing process models for compliance checking as well as for the compliance rules themselves have not been addressed so far. Our approach can further be combined with approaches to manage compliance rules and their relations to process models such as [3, 16]. Further, as the clustering approach does not necessitate a particular compliance checking approach, it can be combined with existing process model verification approaches such as [6, 17].

7 Summary and Outlook

In this work we presented activity-oriented clustering techniques that particularly support the management of large business process and compliance rule repositories independent of any a-priory knowledge (like policies or process meta models). Summarized in a simplified way, the activity-oriented compliance rule clustering bundles compliance rules for each process model and the aggregated rule clustering technique considers the relations between clusters in order to decide if merging clusters reduces the number and thus the efficiency of compliance checks. Furthermore it was shown how the clustering techniques can accelerate consistency checks by introducing a theorem that reduces checks for conflict-freeness of the overall compliance rule sets to respective checks on the clusters. Finally, aspects of quantitative and qualitative evaluations of applying the clustering techniques were discussed. The techniques were explained by means of the use case IT baseline security. In future work we want to define further techniques for managing large collections of business processes and compliance rules, particularly focusing on e.g. indexing techniques, or clustering according to data flows in business processes. Further, the effects of process model evolution on clustering and indexing will be investigated.

References

1. Ly, L.T., Rinderle, S., Dadam, P.: Integration and verification of semantic constraints in adaptive process management systems. Data & Knowledge Engineering **64**(1) (January 2008) 3–23
2. Valkenburg, M.: Van ameyde international case study. Technical report, www.bptrends.com (2010)
3. Namiri, K.: Model-Driven Management of Internal Controls for Business Process Compliance. PhD thesis, University of Karlsruhe (2008)
4. Federal Agency for Security in IT, G.: It baseline security - catalogues. www.bsi.bund.de (2006) in German Language.
5. Ly, L., Rinderle-Ma, S., Dadam, P.: Design and verification of instantiable compliance rule graphs in Process-Aware information systems. In: Advanced Information Systems Engineering. (2010) 9–23 10.1007/978-3-642-13094-6_3.
6. Knuplesch, D., Ly, L., Rinderle-Ma, S., Pfeifer, H., Dadam, P.: On enabling Data-Aware compliance checking of business process models. In: Conceptual Modeling – ER 2010. Volume 6412. (2010) 332–346
7. Tran, H., Zdun, U., Dustdar, S.: VbTrace: using view-based and model-driven development to support traceability in process-driven SOAs. Software and Systems Modeling **10** (2011) 5–29 10.1007/s10270-009-0137-0.
8. Suwa, M., Scott, A.C., Shortcliffe, E.H.: An approach to verifying completeness and consistency in a Rule-Based expert system. AI Magazine **3**(4) (1982)
9. Nguyen, T.A., Perkins, W.A., Laffey, T.J., Pecora, D.: Checking an expert systems knowledge base for consistency and completeness. In: Proc. Int'l Conf. on Artificial intelligence - Vol. 1. (1985) 375–378 ACM ID: 1625205.
10. Hornung, T., Koschmider, A., Oberweis, A.: A recommender system for business process models. SSRN eLibrary (2007)
11. Awad, A., Sakr, S.: Querying Graph-Based repositories of business process models. In: Database Systems for Advanced Applications. Volume 6193. (2010) 33–44
12. Francescomarino, C., Tonella, P.: Crosscutting concern documentation by visual query of business processes. In: Business Process Management Workshops. Volume 17 of LNBIP. (2009) 18–31
13. Awad, A., Decker, G., Weske, M.: Efficient compliance checking using BPMN-Q and temporal logic. In: Int'l Confe. Business Process Management. (2008) 326–341
14. Rosa, M.L., Reijers, H.A., van der Aalst, W.M., Dijkman, R.M., Mendling, J., Dumas, M., García-Bañuelos, L.: APROMORE: an advanced process model repository. Expert Systems with Applications **38**(6) (June 2011) 7029–7040
15. Jin, T., Wang, J., Wu, N., Rosa, M.L., ter Hofstede, A.: Efficient and accurate retrieval of business process models through indexing. In: On the Move to Meaningful Internet Systems: OTM 2010. Volume 6426. (2010) 402–409
16. Namiri, K., Stojanovic, N.: Towards a formal framework for business process compliance. In: Multikonferenz Wirtschaftsinformatik (MKWI 2008). (2008)
17. Liu, Y., Müller, S., Xu, K.: A static compliance-checking framework for business process models. IBM Systems Journal **46**(2) (2007) 335–261