Discovering Instance-Spanning Constraints from Process Execution Logs based on Classification Techniques

Karolin Winter
University of Vienna
Faculty of Computer Science
Vienna, Austria
Email: karolin.winter@univie.ac.at

Stefanie Rinderle-Ma
University of Vienna
Faculty of Computer Science
Vienna, Austria
Email: stefanie.rinderle-ma@univie.ac.at

Abstract—**Process-aware Information Systems (PAIS) have become ubiquitous in companies. Thus the amount of data that can be used to analyze and monitor process executions is vast. The event logs generated by PAIS might contain information about decision making processes and can support the understanding and improving of procedures in companies. Mining decisions and constraints from logs has already been investigated, but so far only for each instance in a separate manner. However, in many practical settings instances are connected to each other if they share, for example, the same resources. Therefore, we present an approach for discovering Instance-Spanning Constraints (ISC) from event logs. The main idea is to identify instance-spanning attributes in the logs and to separate the logs accordingly. Based on these projections, classification algorithms are applied in order to obtain ISC candidates. The feasibility and applicability of the approach is evaluated based on artificial as well as real-life logs. The discovered ISC candidates are then assessed by domain experts.**

Index Terms—**Instance-Spanning Constraints; Constraint Mining; Decision Mining; Classification Techniques**

I. Introduction

Process-Aware Information Systems pervade almost any enterprise and produce a plethora of data on process execution [1]. Typically, this data is collected in process execution logs which contain crucial information on the executed process instances, relevant constraints, and decisions made during process execution. Process mining offers promising means to analyze logs accordingly [2]. Constraints have been mined using declarative mining [3]. Decision mining "aims at the detection of data dependencies that affect the routing of a case" [4].

So far, existing approaches have focused on decisions and constraints that influence and restrict single instances [3] (referred to as so-called intra-instance constraints [5]). Constraints that "refer to more than one instance of one or several process types" [6] are called **Instance-Spanning Constraints (ISC)**. Examples of ISC are *A user is not allowed to execute more than 100 tasks (of any workflow) in a day.* [7] or *The number of connections to UDS should not exceed 10* [8]. As stated in [6] ISC have not been considered comprehensively though they play a decisive role in business process compliance. Report [9] has shown that there are plenty of real-world ISC examples and they encounter a variety of domains, like logistics, health care, security or manufacturing. Clearly, there is a gap between the relevance of ISC and their coverage in existing approaches. In particular, the (automatic) discovery of ISC candidates from process execution logs has been neglected so far. As ISC elicitation is a crucial topic – especially beyond manual elicitation from documents – this paper investigates ISC discovery based on the following research questions.

RQ1 *Which information contained in the event log are indicating ISC?*
RQ2 *How can spanning relations in the logs be detected?*
RQ3 *How to design an algorithm to discover ISC candidates from execution logs?*
RQ4 *How to evaluate the ISC discovery algorithm? What are application scenarios for ISC mining?*

For answering RQ1 and RQ2, real-world ISC and the underlying process instances [9] are investigated. Note that this paper considers ISC that span multiple instances of the same process type, but not across several process types. It will be shown that ISC impose decision points, but not within instances as considered by decision mining, but "between" instances. This finding implies that a) the log information has been pre-processed by a projection on the parts with spanning information and b) classification techniques can be applied on these projections that allow for discovering ISC candidates ($\mapsto$ RQ3). The proposed technique is tested in two application scenarios ($\mapsto$ RQ4), i.e., based on (a) the simulated log files where the ISC are known beforehand and (b) the real-world HEP data set [10] where the possible ISC are not known in advance. For (b) the derived ISC candidates are assessed by experts. Scenario (a) represents conformance checking aspects, i.e., it is possible to investigate whether the recorded log sticks to the imposed ISC while scenario (b) relates to the actual discovery of ISC.

The paper is structured as follows. In Sect. II an overview of terms and notions as well as a definition for ISC decision

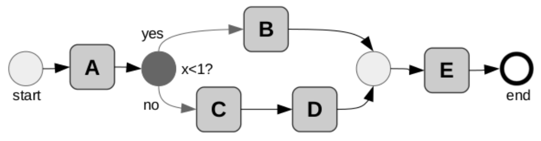


Fig. 1. Petri Net with XOR-Split marked dark grey

points is given. Based on this definition, requirements for log preparations are set out in Sect. III-A. An Algorithm fulfilling these requirements was developed and implemented in Sect. III-B. This algorithm as well as the ISC discovery algorithm presented in Sect. IV, are part of the actual evaluation which is carried out in Sect. V. The evaluation reflects the two application scenarios mentioned before. Sect. VI discusses corner cases and limitations. Sect. VII presents related work. The paper concludes in Sect. VIII.

II. ISC Decision Points

In this section, the basic idea behind discovering ISC from execution logs is explained, but first some terminology is introduced. Based on a **process P** several **cases,** i.e., process instances, can be started and executed. Each instance execution can be represented by a **trace** t_i of unique **events.** The traces are stored in an **event log L**. Each event can be equipped with **(event) attributes** which may contain information about timestamps, organizational resources, and costs [2, p.9].

Formal definitions of these terms can be found in e.g., [2, Sect. 4].

As discussed in the introduction, decision mining [4] has been proposed as technique for discovering one kind of intra-instance constraints, i.e., decision rules. More precisely, relying on event data, decision mining discovers the rules that impose the decisions at alternative splits in process models (holding for each of the related instances in a separate way). The associated split points are referred to as **decision points** and can be identified via XOR gateways in BPMN or several outgoing arcs of a place in a Petri net (cf. Fig. 1).

For the abstract process in Fig. 1 the decision rule states to perform task `B` if $x < 1$ holds and tasks `C` and `D` otherwise. Roughly, decision mining classifies the log traces into the execution paths determined by the decision point and applies decision tree algorithms in order to derive the decision for different traces [4].

In contrast to decision points depending on intra-instance constraints an **ISC decision point** cannot directly be recognized by looking at the process model. The reason is that the ISC spans multiple process instances. The ISC decision point hence lies "in between" the affected instances.

Nevertheless, a visualization for ISC decision points by introducing virtual decision points as depicted in Fig. 2 in order to explain the concept is used.

Definition 1. *An* ISC decision point *is a decision point that is not mirrored by the event log and is linked to a decision rule referring to an instance-spanning attribute, i.e., time, resource or data.*

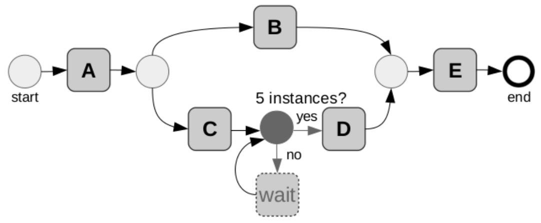


Fig. 2. ISC Decision Point marked dark grey

The example in Fig. 2 extends the model in Fig. 1 by the ISC that *`D` has to be fired in parallel for five instances*, i.e., reflects a synchronization at `D` for a batch of five instances. For the process model this ISC can be mirrored by introducing a virtual decision point right before task `D` that represents the ISC to proceed with `D` in case five instances have arrived or to wait within a loop otherwise. It can be seen that an ISC decision point mimics a deviation from the behavior described by the model by executing tasks that are not explicitly reflected in the execution logs, for example, the implicit task `wait` in Fig. 2. Conversely, traces in the event log are not influenced by that change of the process model. Overall, it can be concluded that decision mining on execution logs as presented in [4] cannot be directly applied to discover ISC.

So, an algorithm for pre-processing the event log in order to be able to utilize classification and clustering techniques is suggested. The next section provides requirements that need to be fulfilled when pre-processing the event log.

III. Pre-processing the Log Files: Multiple Instance and Event Attribute View

A. Requirements for pre-processing

The idea of (virtual) ISC decision points as described in the previous section together with a technique similar to decision mining seems promising for discovering ISC. One difference is that ISC span multiple instances and this suggests that a multiple instance view on the execution logs is required. This means that in contrast to most process mining approaches one has to consider all traces simultaneously not separately.

$\Longrightarrow$ Reqt. 1: Multiple instance view on execution logs

Modeling requirements time, data, resources (on top of structural aspects) for process constraints in general are stated by [3]. In [6], it was analyzed that ISC are subject to these requirements as well. In addition, ISC might contain trigger actions, i.e., actions that are triggered by the conditions set out in the ISC. For the above synchronization example, the triggered actions are `wait` for a batch size lower than 5 right before activity `D` and proceed with `D` for a batch size of 5.

It can be concluded that the ISC-related information in the execution logs is encoded in the event attributes task, organizational resources, timestamps, and data where tasks often represent delays of event executions or concurrency. Looking at the synchronization example, the ISC-related information is encoded in the attribute `timestamp` of event `D`, which would appear to be similar for packages of five instances.

In connection with that, attributes related to task `D` and the implicit events for invisible task `wait` have to be considered.

In summary, an event attribute view on execution logs is required where the attributes are tasks, time, data, and resources. These event attributes can then be utilized as classifier in the proposed algorithm (cf. Sect. IV).

$\Longrightarrow$ Reqt. 2: Event attribute view on execution logs

The implementation of Reqt. 1 and 2 (cf. III-B) results in a log consisting of one trace encountering all events. This results in a complex log, L_{vm} (virtual log that contains all merged traces), and a grouping of L_{vm} should be issued in order to reduce the dimension of the problem. Since the attributes timestamps, data and resources represent spanning-relations, the grouping of the log depends on these attributes. This consideration is added as third requirement:

$\Longrightarrow$ Reqt. 3: Dimension reduction by projection on partial logs based on instance-spanning attributes

In Sect. III-B, an algorithm is proposed that illustrates how the logs can be pre-processed in order to meet the set out requirements. Sect. IV provides the ISC discovery algorithm, i.e., classification based on the pre-processed event logs.

B. Implementation of Requirements

This section yields an algorithm to pre-process execution logs for ISC discovery based on the requirements set out in Sect. III-A as well as an exemplary techniqual implementation relying on the Attribute-Relation File Format (arff)[1] which serves as input for Weka[2].

Before introducing this preparation algorithm, let us take a closer look at the event attributes. According to [6] ISC refer to event attributes such as tasks, resources, timestamps, and data. Specifically, ISC always refer to tasks (structural information). Moreover, in real-world scenarios, ISC often impose a time threshold for different settings such as the (non) parallel execution of instances or a threshold for process executions per day [9]. In these cases, the ISC depends on the time attribute and one has to choose a certain level of time granularity meaning that it might be necessary to coarsen the timestamps to minutes, hours, days, etc.. This is of course depending on the particular situation and could either be done by domain experts or an algorithm depending on stochastic methods. The latter is subject to future work.

In order to meet Reqt. 1, the trace structure of a log L is removed, i.e., the emerging log only contains events which are not related to a specific trace resulting in a virtual intermediate log L_{vm}. After merging all events separately into this virtual log one needs to find a method for reducing the dimensions again in order to impose a new structure that allows for discovering ISC candidates other than synchronization. Every present instance-spanning attribute could be used for this dimension reduction. Mostly, instance-spanning parts relate to organizational resources. This is also confirmed by the evaluation in Sect. V. Nevertheless, one could try each attribute. We want to mention that timestamps could be too selective, i.e., too many single arff files could be produced in the end because it is likely that a lot of different timestamps are present in the log. So in this case it might be useful to adjust the granularity first as described before.

For meeting Reqt. 2, we opt to transform L_{vm} into an existing format that reflects an event attribute view and can directly be used for applying classification techniques. A good candidate here is the Attribute-Relation File Format (arff) which is further combined with the Weka data mining framework. XES[3] is used as input format. [4] As a result, for each of the partial logs a decision tree or rule set is discovered (cf. Sect. IV) which can be aggregated to improve the results.

```
Data: Process execution log $L$ with traces $t_i$; selected
      attribute $A_s$
Result: one or more arff file(s)
$L_{vm} = []$;
for $i <$ number of traces in $L$ do
    $L_{vm}$.appendAll(events in traces[i]);
end
attributeSet := unique $A_s$ in $L_{vm}$;
if attributeSet.size $< 2$ then
    $L_{vm}$.to_arff;
else
    attributeEventMap := [$L_{vm}$[events] grouped by $A_s$];
    for attribute, events in attributeEventMap do
        attributeEventMap[attribute].to_arff;
    end
end
```

Algorithm 1: Projection and transformation algorithm

Alg. 1 describes the projection and transformation algorithm. First L_{vm} is generated from the original log L by merging all traces together. Under the assumption that the indermediate log should be grouped by organizational resources the algorithm tests first if there is more than one resource present in the log. If so, the log is projected onto partial logs having the same resource. These logs are then saved as arff files. In particular, the transformed files do now only contain individual events which are no longer embedded into traces. Each event attribute appearing at least once in the log is taken as (class) attribute and the response variable is chosen from that set. One data instance in the arff file consists of the attributes appearing in the corresponding event with missing values being marked as "?".

Consider the following XES snippet (Listing 1) for the centrifuge example introduced in Sect. V where we can see the attributes and attribute values for two events, i.e., `examine mixtures` (Listing 1, lines 3-9) and `put in centrifuge` (Listing 1, lines 11-16).

[1]https://weka.wikispaces.com/ARFF

[2]http://www.cs.waikato.ac.nz/ml/weka/

[3]www.xes-standard.org

[4]It should be emphasized that other frameworks providing implementations of classification algorithms could be used as well. Nevertheless, we chose ARFF since its structure illustrates Reqt. 2.

Listing 1. Centrifuge example snippet in XES format

```
<trace>
  <string key="concept:name" value="case_1"/>
  <event>
    <string key="concept:name" value="examine_mixtures"/>
    <string key="org:resource" value="UNDEFINED"/>
    <date key="time:timestamp"
          value="2016-07-13T08:59:24.000+02:00"/>
    <string key="lifecycle:transition" value="complete"/>
  </event>
  <event>
    <string key="concept:name" value="put_in_centrifuge"/>
    <string key="org:resource" value="UNDEFINED"/>
    <date key="time:timestamp"
          value="2016-07-13T08:59:52.000+02:00"/>
    <string key="lifecycle:transition" value="complete"/>
  </event>
      ...
  </trace>
```

Applying Alg. 1 ($A_s = organizational \quad resource$) to this log results in the arff file depicted by Listing 2 (no partial logs). While the XES log contained the trace information (`value="case 1"`) the arff file contains the event attributes for all traces (instance-spanning).

Listing 2. Centrifuge example snippet in arff format – no resource-based split

```
@RELATION Example_2
@ATTRIBUTE "concept:name" {"examine mixtures","put in
centrifuge", "centrifugation", "put out of centrifuge",
"check result", "documentation","clean bin"}
@ATTRIBUTE "org:resource" STRING
@ATTRIBUTE "timestamp" {"2016-07-13T08:59:24.000+02:00",...}
@ATTRIBUTE "lifecycle:transition" STRING
@DATA
"examine mixtures", "UNDEFINED",
"2016-07-13T08:59:24.000+02:00", "complete"
"put in centrifuge", "UNDEFINED",
"2016-07-13T08:59:52.000+02:00", "complete"
...
```

Consider another ISC stating that *A user is not allowed to do event* `approve loan` *if the total loan amount per day and clerk exceeds $1M.* [7]. Assume a corresponding execution log containing events for 10 clerks ($\widehat{=}$ resources). Alg. 1 ($A_s = organizational \quad resource$) would merge all traces in one log and then project the merged log on 10 projections along the 10 resources. Finally, for each of these projections the corresponding arff file is generated.

IV. Discovering ISC Candidates

The prepared arff files (cf. Alg. 1) form the input for discovering the ISC decision points that are controlled based on ISC candidates (cf. Sect. II). The metaphor of ISC decision points suggests the application of classification techniques as for discovering decision rules for decision points (cf. [4]). More precisely, **decision trees** as well as **decision rules** (cf. [11]) will be used as they can directly be interpreted by users. Both are supervised classifying learning techniques and display the most likely decision paths. Computing decision rules and trees requires a set of **attributes**, also called **predictor variables** and one **response variable** or **classifier.** The candidates for both are the attributes within the arff files and one data instance of this file corresponds to one event and its associated attributes in the primary log.

Alg. 2 sets out a template on how to proceed with the generated arff files and encounters interactive parts (a)-c)) depending on user choices. It is recommended to start with **JRip** because it can handle outliers better than e.g. **C4.5** [12] and `event name` as classifier (as mentioned before event names represent spanning relations). This is reasonable since the arff file is likely "noisy" due to the load of events in the log and only a few of them might be related to ISC. As it will be shown in the evaluation (Example 1) an ISC imposing a synchronization can be detected without further steps. The number of runs *r* is chosen by the user and its maximum is the sum of possible adaptions in combination with the number of decision tree and rule algorithms currently implemented in Weka. If an ISC candidate is found it is appended to a list. At the end the list is returned to the user.

```
Data: One arff file
Result: List of ISC candidate(s)
apply JRip on arff file;
determine number of runs r;
i = 0;
candidates = [];
while $i < r$ do
    a) remove events that are not relevant/do not appear;
    b) adapt granularity of timestamps;
    c) apply classification technique;
    if ISC candidate found is true then
        candidates.append(candidate);
    end
    i++;
end
return candidates;
```

Algorithm 2: ISC candidate discovery algorithm per arff file

After the first application of JRip, it might become necessary to adapt the log. This means that the timestamps could be coarsened to a suitable time level, e.g., minutes, hours, or days and certain events should be neglected due to unimportance for ISC. When the ISC is known beforehand and should just be checked like in the first part of the evaluation these transformations are obvious. If the existence of ISC is unknown domain knowledge can be useful but is not required as shown in the second part of the evaluation. It is always possible to try different adaptions and check whether the results have improved, i.e., if an ISC candidate was found after the adaptions or not. Overall, two options for improving the results in finding ISC candidates are conceivable: a) coarsening the timestamp granularity and b) erasing events from the log. Mining the process model can support the choice. As mentioned in Sec. III, the automatic suggestion of the time granularity is future work and making recommendations on the relevance of events for the ISC candidates can be added to this.

V. Applying the proposed algorithms

Supplementary material of the evaluation such as code, logs, and analysis results is provided at http://bit.ly/2lztLv6. The source code of the log preparation algorithm is written in Ruby using the rarff[5], xml-smart[6] and xes[7] gems.

The ISC discovery method (cf. Sect. III and IV) is evaluated in two ways. In Sect. V-A, its feasibility is shown based on two artificial logs created by respecting two different real-world ISC, i.e., the ISC are known beforehand. Secondly, the method is applied to a real-world log in Sect. V-B for which potential ISC are not known beforehand. The validity of discovered ISC is evaluated based on domain expert interviews afterwards. Both scenarios reflect use cases for the proposed methods.

A. Artificial Logs

First, two artificial logs reflecting commonly used ISC (simultaneous execution of events or delays) are investigated. The log files, generated by Ruby scripts, respect the ISC, but also include random aspects. In this case an event log conformance check with respect to the ISC is performed.

Example 1: The ISC reads as follows *Wait until centrifuge is filled.* [13]. In Fig. 3 the process models based on which the first artificial log is generated is depicted using BPMN.

Only one organizational resource (undefined) was included and consequently Alg. 1 with $A_s = organizational \quad resource$ transferred the whole log (consisting of 500 traces) into a single arff file for classification. Applying Alg. 1 with $A_s = timestamp$ results in 2956 separate arff files. Since in this case the ISC should be checked, i.e., it is known beforehand, it is reasonable to start with the result of Alg. 1 for $A_s = organizational \quad resource$ to keep the computational effort as low as possible. In addition we know that the centrifuge (shared resource) is responsible for the instance-spanning part.

Following Alg. 2, JRip with pruning was used as first classification technique. The result of this analysis is shown in Fig. 4. It can be seen that the timestamps and the desired event `centrifugation` with no misclassified instances is the activity showing up most often in the decision rules. The last line containing `check results` means that all other cases are captured by this rule.

It can also be recognised that `centrifugation` is always issued for a batch of 25 instances simultaneously. The numbers in brackets (25/0) indicate this (25 instances where captured by this rule and zero of them were misclassified). Therefore, an ISC candidate could be *The event* `centrifugation` *has to be issued in parallel for every 25 instances.* Since the ISC should be checked, r was set to 0, because a suitable ISC candidate was found.

If the process becomes more complex the ISC candidate could also be found without any difficulties. Having more than one centrifuge in use results in several log files (Alg. 1 with $A_s = organizational \quad resource$). This would obviously not affect the results except that one decision rule set for each centrifuge would be produced. The number of slots in the centrifuge can be determined by considering the number of correctly classified instances when `centrifugation` took place. So, checking the original ISC *wait until centrifuge is filled* can be issued if the user is familiar to the quantity of slots. Furthermore it can be concluded whether the centrifuges' capacity was fully exploited.

If the ISC is not known beforehand, i.e., it should be discovered, then the ISC candidate could be assessed by conducting an expert interview.

Example 2: Another ISC, taken from the financial sector, is *A user is not allowed to do event* `approve loan` *if the total loan amount per day and clerk exceeds $1M.* [7]. The associated process is depicted in Fig. 5. It starts with a customer request. First, a solvency check is performed. If this check is negative the customer is informed about the rejection and his request is deleted. Otherwise a payment schedule is developed and sent to the customer. If the customer does not agree on the schedule his request is deleted, else the events `arrange loan`, `approve loan` and `inform customer again`, `inform head office` are performed in parallel where event `approve loan` may only be issued if the ISC is not violated. Each trace finishes when the request is completed.

In consequence one instance (trace) in the event log represents a customer loan request which is executed by one clerk and the event attributes are `event name`, `organizational resource (clerk)` and a `timestamp`. The attributes `requested amount` and already approved `amount per clerk` and day only show up in the log when the events `arrange loan` and `approve loan` are triggered.

The event log is generated as follows: At the beginning of each instance one out of 10 clerks is randomly chosen as well as a requested amount between \$5.000 and \$500.000. With a probability of 10% the solvency check is negative and with a probability of 5% the customer rejects the payment schedule. If the ISC is triggered, i.e., the requested amount together with the already approved amount would exceed \$1M, the timestamp of event `approve loan` is generated for the following work day. The resulting event log contains 500 traces.

Based on this log, Alg. 1 with $A_s = organizational \quad resource$ produces 10 arff files, one for each clerk. These files only contain single events representing the data instances and are not related to a specific trace anymore. Using $A_s = timestamp$ in Alg. 1 results in 3689 arff files. Checking the ISC is the purpose of this case. This helps in directly choosing the appropriate time granularity (day level). Reapplying Alg. 1 with timestamps coarsened to day level generated 21 arff files. This is still more than twice as much as using $A_s = organizational \quad resource$. For the additional data attributes (requested amount and already approved amount per clerk) one receives 11 arff files. So the analysis was conducted on the 10 arff files related to the resource.

It should be mentioned that the ISC of Example 2 is more complex than the one for Example 1 as it consists of two parts, i.e., the threshold of \$1M per day and clerk as well as

[5]https://rubygems.org/gems/rarff/versions/0.2.2
[6]https://rubygems.org/gems/xml-smart
[7]https://rubygems.org/gems/xes

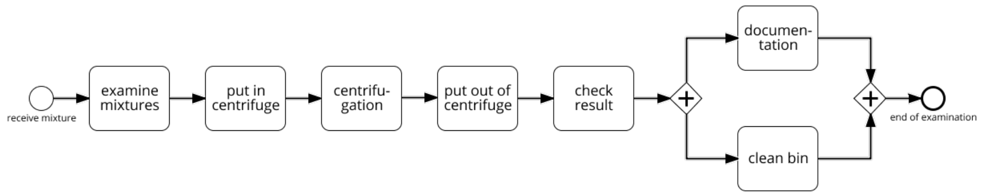
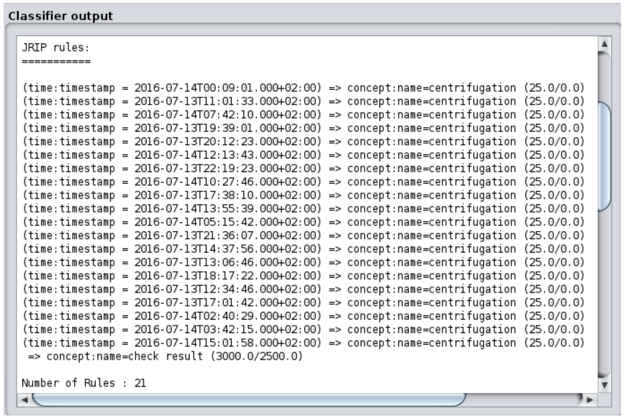


Fig. 3. Process model for Example 1, using Signavio



```
JRIP rules:
===========

(time:timestamp = 2016-07-14T00:09:01.000+02:00) => concept:name=centrifugation (25.0/0.0)
(time:timestamp = 2016-07-13T11:01:33.000+02:00) => concept:name=centrifugation (25.0/0.0)
(time:timestamp = 2016-07-14T07:42:10.000+02:00) => concept:name=centrifugation (25.0/0.0)
(time:timestamp = 2016-07-13T19:39:01.000+02:00) => concept:name=centrifugation (25.0/0.0)
(time:timestamp = 2016-07-13T20:12:23.000+02:00) => concept:name=centrifugation (25.0/0.0)
(time:timestamp = 2016-07-14T12:13:43.000+02:00) => concept:name=centrifugation (25.0/0.0)
(time:timestamp = 2016-07-13T22:19:23.000+02:00) => concept:name=centrifugation (25.0/0.0)
(time:timestamp = 2016-07-14T10:27:46.000+02:00) => concept:name=centrifugation (25.0/0.0)
(time:timestamp = 2016-07-13T17:38:10.000+02:00) => concept:name=centrifugation (25.0/0.0)
(time:timestamp = 2016-07-14T13:55:39.000+02:00) => concept:name=centrifugation (25.0/0.0)
(time:timestamp = 2016-07-14T05:15:42.000+02:00) => concept:name=centrifugation (25.0/0.0)
(time:timestamp = 2016-07-13T21:36:07.000+02:00) => concept:name=centrifugation (25.0/0.0)
(time:timestamp = 2016-07-13T14:37:56.000+02:00) => concept:name=centrifugation (25.0/0.0)
(time:timestamp = 2016-07-13T13:06:46.000+02:00) => concept:name=centrifugation (25.0/0.0)
(time:timestamp = 2016-07-13T18:17:22.000+02:00) => concept:name=centrifugation (25.0/0.0)
(time:timestamp = 2016-07-13T12:34:46.000+02:00) => concept:name=centrifugation (25.0/0.0)
(time:timestamp = 2016-07-13T17:01:42.000+02:00) => concept:name=centrifugation (25.0/0.0)
(time:timestamp = 2016-07-14T02:40:29.000+02:00) => concept:name=centrifugation (25.0/0.0)
(time:timestamp = 2016-07-14T03:42:15.000+02:00) => concept:name=centrifugation (25.0/0.0)
(time:timestamp = 2016-07-14T15:01:58.000+02:00) => concept:name=centrifugation (25.0/0.0)
 => concept:name=check result (3000.0/2500.0)

Number of Rules : 21
```

Fig. 4. Result for the first example applying JRip

the delay of the approval to the following work day.

Following Alg. 2, JRip with default values, no pruning and `event name` as classifier was utilized as initial step. The results for clerk 1 are shown in Fig. 6.

It can be seen that `approve loan` was e.g., issued for an `amount` between \$501227 and \$862607 and a `requested amount` between \$164442 and \$173196. Consequently, there has to be a threshold for approving a loan request and even a range for the upper limit can be deduced.

The events `approve loan` and `arrange loan` seem to be more important than others but their dependency cannot be resolved at this stage of the analysis. So further steps are needed.

According to Alg. 2 every event except `arrange loan` and `approve loan` is removed from the log, so only the data instances containing these events are left. Here it does not make any difference if the ISC is known beforehand or not because this step is based on the previous results and quite intuitive.

Knowing the ISC beforehand, i.e., using this approach for conformance checking, reduces the effort to find the right time granularity, e.g., day level for the timestamps is immediately chosen if the user is familiar with the ISC. Consequently, *r* is chosen as the number of implemented decision tree and rule algorithms in Weka.

If the ISC is unknown, the granularity of the timestamps has to be coarsened incrementally, e.g., first the timestamps are coarsened to minutes, then hours and at least days. For each level classification techniques, in this case the **RandomTree** algorithm, have to be applied using `event name` as response variable since the relation between `arrange loan` and `approve loan` has to be resolved. The final result is not influenced because the dependency between `arrange loan` and `approve loan` cannot be resolved unless the right granularity is chosen. In this application scenario the ISC should only be checked, so the timestamps are coarsened to day level and the corresponding decision tree is illustrated in Fig. 7. The events `arrange loan` and `approve loan` marked green (dark grey) are the ones representing the ISC. For `requested amount` smaller than \$190862 only `arrange loan` was issued that day. It can be concluded that if the event `approve loan` can be found the next day for an amount that is smaller than \$190862 the ISC was triggered on that day causing a delay of the loan approval to the next day. Similar results are received for the remaining nine clerks. The discovered ISC candidate is: *Each clerk is allowed to issue* `approve loan` *as long as a threshold (around \$1M) is not reached. Otherwise he has to delay this event to the following day.*

Using $A_s = timestamp$ with timestamps coarsened to day level results in the rules depicted in Fig. 8. JRip without pruning was applied on one timestamp. The third rule indicates a threshold for clerk 5 (`requested amount` between \$54902 and \$286465, `amount of clerk 5` bigger or equal to \$649282 then `approve loan`) but the rules are not as significant as for the splitting by resources. The delay cannot be seen. This is not surprising since the timestamps are considered separately in this case.

So far our approach has been applied to artificial logs for which the ISC was known beforehand. ISC candidates could be derived that reflect the original ISC. In addition, the approach could also be used for deriving additional information or reveal ISC changes, e.g., in the first example it could be investigated if the maximum number of slots was used or in the second example a change of the threshold for the loan amount could be noticed. However the goal is also to find ISC candidates when no precedent information about the log is available. This is evaluated for a real-life log in the subsequent section.

B. Real-life Logs

The real-life log [10] contains information on several semester courses in higher education. For the following analysis one course was selected resulting in 74 instances reflecting

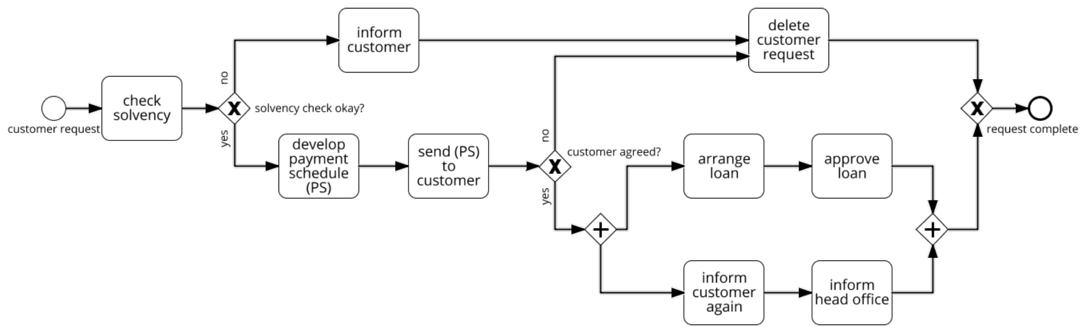
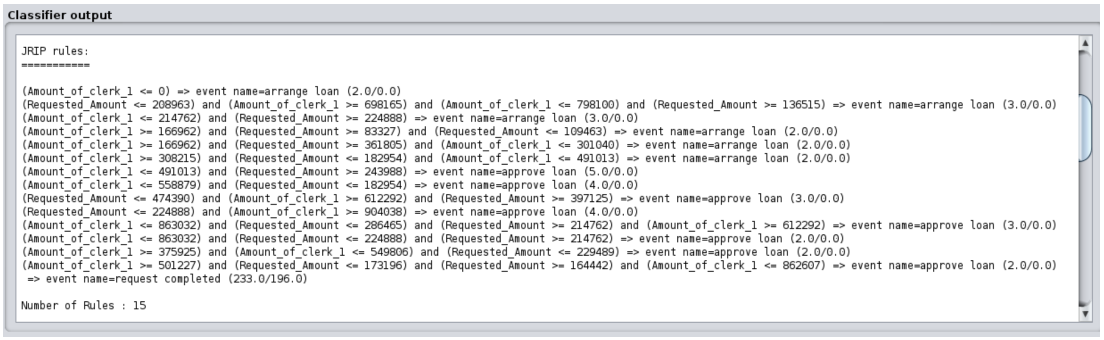


Fig. 5. Process model for Example 2, using Signavio



```
Classifier output

JRIP rules:
===========

(Amount_of_clerk_1 <= 0) => event name=arrange loan (2.0/0.0)
(Requested_Amount <= 208963) and (Amount_of_clerk_1 >= 698165) and (Amount_of_clerk_1 <= 798100) and (Requested_Amount >= 136515) => event name=arrange loan (3.0/0.0)
(Amount_of_clerk_1 <= 214762) and (Requested_Amount >= 224888) => event name=arrange loan (3.0/0.0)
(Amount_of_clerk_1 >= 166962) and (Requested_Amount >= 83327) and (Requested_Amount <= 109463) => event name=arrange loan (2.0/0.0)
(Amount_of_clerk_1 >= 166962) and (Requested_Amount >= 361805) and (Amount_of_clerk_1 <= 301040) => event name=arrange loan (2.0/0.0)
(Amount_of_clerk_1 >= 308215) and (Requested_Amount <= 182954) and (Amount_of_clerk_1 <= 491013) => event name=arrange loan (2.0/0.0)
(Amount_of_clerk_1 <= 491013) and (Requested_Amount >= 243988) => event name=approve loan (5.0/0.0)
(Amount_of_clerk_1 <= 558879) and (Requested_Amount <= 182954) => event name=approve loan (4.0/0.0)
(Requested_Amount <= 474390) and (Amount_of_clerk_1 >= 612292) and (Requested_Amount >= 397125) => event name=approve loan (3.0/0.0)
(Requested_Amount <= 224888) and (Amount_of_clerk_1 >= 904038) => event name=approve loan (4.0/0.0)
(Amount_of_clerk_1 <= 863032) and (Requested_Amount <= 286465) and (Requested_Amount >= 214762) and (Amount_of_clerk_1 >= 612292) => event name=approve loan (3.0/0.0)
(Amount_of_clerk_1 <= 863032) and (Requested_Amount <= 224888) and (Requested_Amount >= 214762) => event name=approve loan (2.0/0.0)
(Amount_of_clerk_1 >= 375925) and (Amount_of_clerk_1 <= 549806) and (Requested_Amount <= 229489) => event name=approve loan (2.0/0.0)
(Amount_of_clerk_1 >= 501227) and (Requested_Amount <= 173196) and (Requested_Amount >= 164442) and (Amount_of_clerk_1 <= 862607) => event name=approve loan (2.0/0.0)
 => event name=request completed (233.0/196.0)

Number of Rules : 15
```

Fig. 6. Result applying JRip for example two with event name as classifier

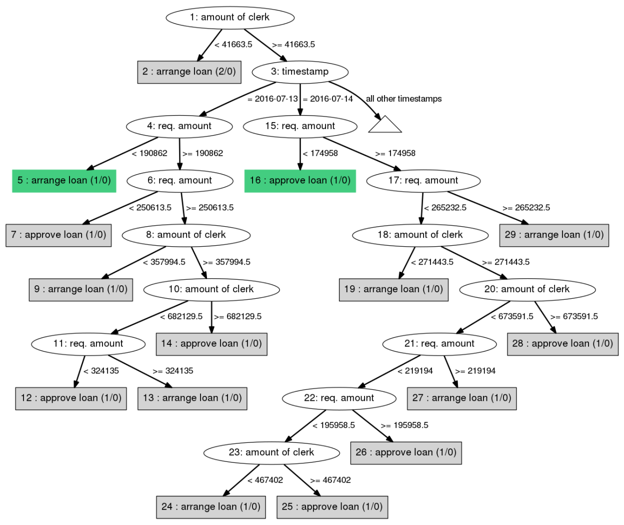


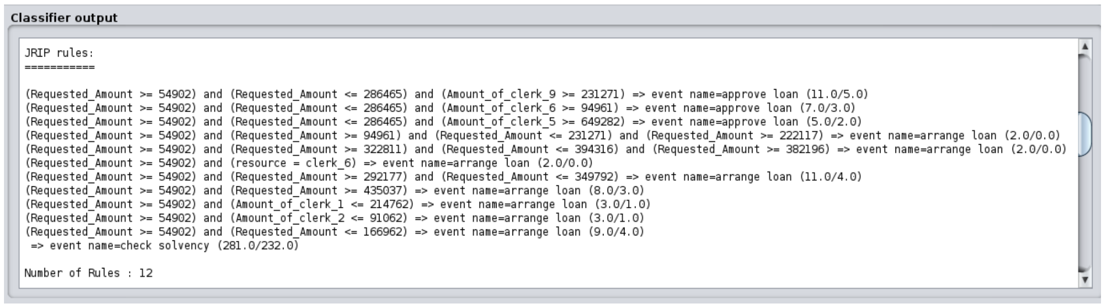
Fig. 7. Decision tree for second example, computed with RandomTree for day level, only including events `arrange loan` and `approve loan`

the students participating in this course. One trace therefore represents the progress and events related to that student during the course. It is not known beforehand whether the log contains data on ISC or not.

Examples of events are deadlines of milestones and exercises which are issued by the organizational resource system. Two other groups of organizational resources are the different tutors and lecturers who trigger events like `Milestone feedback` or `Evaluation of presentation`. The students are also organizational resources because they can upload milestones or can ask questions in different forums. All types of events are equipped with five attributes: `organizational resource`, `data`, `concept name`, `lifecycle transition` and `timestamp`. The `data` attribute contains different information depending on the organizational resource, e.g., for a lecturer, `data` is a list consisting of the course id, points, group id, lecturer id, etc.. To keep things simple and as we were not convinced of its relevance for ISC this attribute was not included.

Alg. 1 with $A_s = organizational\ \ resource$ produced one file for each student, tutor, lecturer and the system, altogether 80 files. Grouping them by organizational roles four arff files **studentsJoined, tutorsJoined, lecturersJoined** and **system** result. This is reasonable because students, as well as tutors and lecturers are likely to have the same tasks and trigger the same events. For $A_s = timestamp$ the outcome was 2077 arff files, so the ISC candidate search was again issued for $A_s = organizational\ \ resource$.

The ISC candidate search started by executing Alg. 2 on the **system data set**. JRip without pruning and `event name` as classifier delivered the results shown in Fig. 9. It can be recognized that these rules seem to be incomplete since there are some deadlines missing as far as one assumes the log to be consistent. To figure this out RandomTree with default values was applied delivering the following results (cf. Fig. 10). The number of total instances (148) to incorrectly classified instances (74) at e.g., `Exercise 2 submission`

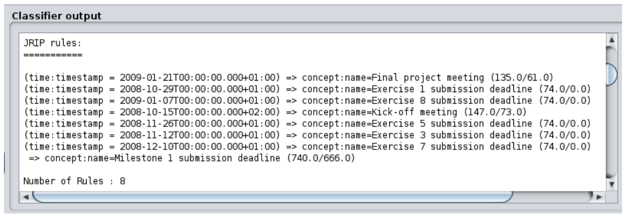


```
JRIP rules:
===========

(Requested_Amount >= 54902) and (Requested_Amount <= 286465) and (Amount_of_clerk_9 >= 231271) => event name=approve loan (11.0/5.0)
(Requested_Amount >= 54902) and (Requested_Amount <= 286465) and (Amount_of_clerk_6 >= 94961) => event name=approve loan (7.0/3.0)
(Requested_Amount >= 54902) and (Requested_Amount <= 286465) and (Amount_of_clerk_5 >= 649282) => event name=approve loan (5.0/2.0)
(Requested_Amount >= 54902) and (Requested_Amount >= 94961) and (Requested_Amount <= 231271) and (Requested_Amount >= 222117) => event name=arrange loan (2.0/0.0)
(Requested_Amount >= 54902) and (Requested_Amount >= 322811) and (Requested_Amount <= 394316) and (Requested_Amount >= 382196) => event name=arrange loan (2.0/0.0)
(Requested_Amount >= 54902) and (resource = clerk_6) => event name=arrange loan (2.0/0.0)
(Requested_Amount >= 54902) and (Requested_Amount >= 292177) and (Requested_Amount <= 349792) => event name=arrange loan (11.0/4.0)
(Requested_Amount >= 54902) and (Requested_Amount >= 435037) => event name=arrange loan (8.0/3.0)
(Requested_Amount >= 54902) and (Amount_of_clerk_1 <= 214762) => event name=arrange loan (3.0/1.0)
(Requested_Amount >= 54902) and (Amount_of_clerk_2 <= 91062) => event name=arrange loan (3.0/1.0)
(Requested_Amount >= 54902) and (Requested_Amount <= 166962) => event name=arrange loan (9.0/4.0)
 => event name=check solvency (281.0/232.0)

Number of Rules : 12
```

Fig. 8. Result for the second example applying JRip on one timestamp coarsened to day level

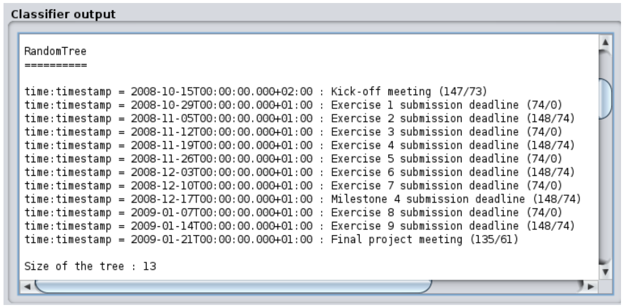


```
JRIP rules:
===========

(time:timestamp = 2009-01-21T00:00:00.000+01:00) => concept:name=Final project meeting (135.0/61.0)
(time:timestamp = 2008-10-29T00:00:00.000+01:00) => concept:name=Exercise 1 submission deadline (74.0/0.0)
(time:timestamp = 2009-01-07T00:00:00.000+01:00) => concept:name=Exercise 8 submission deadline (74.0/0.0)
(time:timestamp = 2008-10-15T00:00:00.000+02:00) => concept:name=Kick-off meeting (147.0/73.0)
(time:timestamp = 2008-11-26T00:00:00.000+01:00) => concept:name=Exercise 5 submission deadline (74.0/0.0)
(time:timestamp = 2008-11-12T00:00:00.000+01:00) => concept:name=Exercise 3 submission deadline (74.0/0.0)
(time:timestamp = 2008-12-10T00:00:00.000+01:00) => concept:name=Exercise 7 submission deadline (74.0/0.0)
 => concept:name=Milestone 1 submission deadline (740.0/666.0)

Number of Rules : 8
```

Fig. 9. Result applying JRip without pruning for the system data set

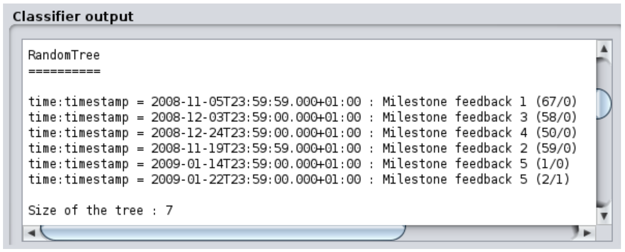


```
RandomTree
==========

time:timestamp = 2008-10-15T00:00:00.000+02:00 : Kick-off meeting (147/73)
time:timestamp = 2008-10-29T00:00:00.000+01:00 : Exercise 1 submission deadline (74/0)
time:timestamp = 2008-11-05T00:00:00.000+01:00 : Exercise 2 submission deadline (148/74)
time:timestamp = 2008-11-12T00:00:00.000+01:00 : Exercise 3 submission deadline (74/0)
time:timestamp = 2008-11-19T00:00:00.000+01:00 : Exercise 4 submission deadline (148/74)
time:timestamp = 2008-11-26T00:00:00.000+01:00 : Exercise 5 submission deadline (74/0)
time:timestamp = 2008-12-03T00:00:00.000+01:00 : Exercise 6 submission deadline (148/74)
time:timestamp = 2008-12-10T00:00:00.000+01:00 : Exercise 7 submission deadline (74/0)
time:timestamp = 2008-12-17T00:00:00.000+01:00 : Milestone 4 submission deadline (148/74)
time:timestamp = 2009-01-07T00:00:00.000+01:00 : Exercise 8 submission deadline (74/0)
time:timestamp = 2009-01-14T00:00:00.000+01:00 : Exercise 9 submission deadline (148/74)
time:timestamp = 2009-01-21T00:00:00.000+01:00 : Final project meeting (135/61)

Size of the tree : 13
```

Fig. 10. Result of RandomTree for the system data set



```
RandomTree
==========

time:timestamp = 2008-11-05T23:59:59.000+01:00 : Milestone feedback 1 (67/0)
time:timestamp = 2008-12-03T23:59:00.000+01:00 : Milestone feedback 3 (58/0)
time:timestamp = 2008-12-24T23:59:00.000+01:00 : Milestone feedback 4 (50/0)
time:timestamp = 2008-11-19T23:59:59.000+01:00 : Milestone feedback 2 (59/0)
time:timestamp = 2009-01-14T23:59:00.000+01:00 : Milestone feedback 5 (1/0)
time:timestamp = 2009-01-22T23:59:00.000+01:00 : Milestone feedback 5 (2/1)

Size of the tree : 7
```

Fig. 11. Result of RandomTree for the tutor data set

`deadline` and the lack of milestone deadlines suggest that some deadlines might have taken place on the same day. This could be verified by inspecting the log.

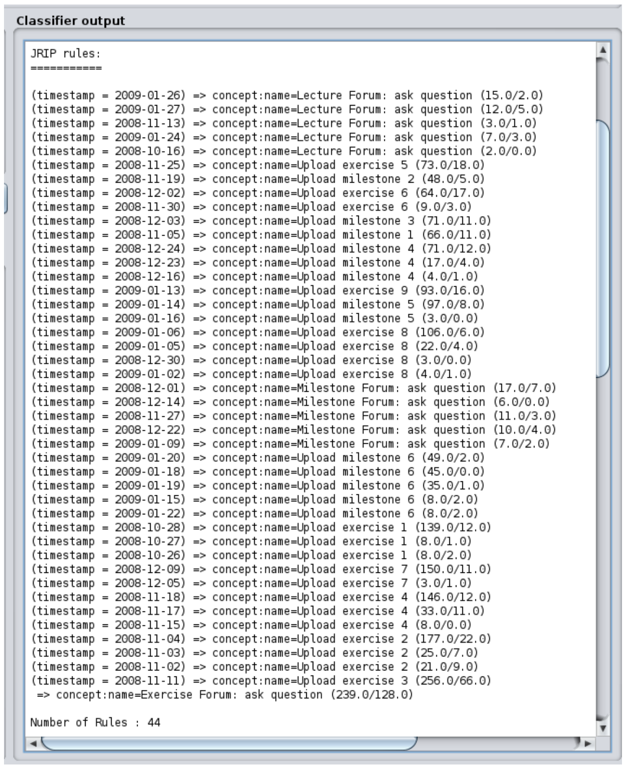
The results for the **tutor data set** are shown in Fig. 11. It seems as if there was a deadline for the evaluation of milestones because all events `Milestone feedback` take place simultaneously for all students (no misclassified instances). The timestamps raise the question whether there could be an internal system that is uploading all the feedbacks at the same time. Consequently, one could think of an ISC candidate like *The tutors should give feedback for every student and milestone within a certain time range/until a certain date (e.g., within 24 hours after deadline or before the deadline of the following milestone). The feedbacks for every milestone are uploaded simultaneously.* ($\mathcal{C}$ I)

Similar deadlines can be seen for the **lecturer data set**. Here the events are also milestone and exercise feedbacks as well as evaluations of student presentations. An analogous ISC could be stated *All exercise and milestone feedbacks as well as evaluations of presentations are uploaded simultaneously.* ($\mathcal{C}$ II)

The **student data set** was the largest one. Starting again with JRip with pruning 76 rules were received. As for the second example it can be recognized that exact timestamps are not important here and so they were coarsened to day level resulting in the rules displayed in Fig. 12.

It can be seen that most students seem to upload their exercises and milestones very closely to the deadlines and after the deadlines expired no uploads seem to be possible anymore. Consequently, another ISC candidate is *All exercises and milestones have to be uploaded before the deadline.* ($\mathcal{C}$ III)

Expert Interviews: The feasibility of the discovered ISC candidates is checked based on interviews with domain experts, i.e., at first with two senior lecturers. The goal was to learn whether there is some demand for imposing ISC in the teaching domain in general. The definition of ISC was explained to the senior lecturers and they were asked if they can think of ISC in one of their courses. Both mentioned that group work might lead to ISC since group members would have to upload their exercises in sync before the deadline. If there is no groupwork required the deadlines would rather affect each instance separately. It can be concluded that $\mathcal{C}$ III can be considered as ISC in case of group work. When being asked about $\mathcal{C}$ I and $\mathcal{C}$ II the senior lecturers stated that there are in general no official deadlines for lecturers or tutors during the semester, but it is common that most teachers upload their feedbacks at the same time. On top of these rather general statements another interview with a lecturer who was responsible for the evaluated course was conducted. He mentioned that in this course no group work was required, therefore $\mathcal{C}$ III is not instance-spanning. $\mathcal{C}$ I and $\mathcal{C}$ II do



```
JRIP rules:
===========

(timestamp = 2009-01-26) => concept:name=Lecture Forum: ask question (15.0/2.0)
(timestamp = 2009-01-27) => concept:name=Lecture Forum: ask question (12.0/5.0)
(timestamp = 2008-11-13) => concept:name=Lecture Forum: ask question (3.0/1.0)
(timestamp = 2009-01-24) => concept:name=Lecture Forum: ask question (7.0/3.0)
(timestamp = 2008-10-16) => concept:name=Lecture Forum: ask question (2.0/0.0)
(timestamp = 2008-11-25) => concept:name=Upload exercise 5 (73.0/18.0)
(timestamp = 2008-11-19) => concept:name=Upload milestone 2 (48.0/5.0)
(timestamp = 2008-12-02) => concept:name=Upload exercise 6 (64.0/17.0)
(timestamp = 2008-11-30) => concept:name=Upload exercise 6 (9.0/3.0)
(timestamp = 2008-12-03) => concept:name=Upload milestone 3 (71.0/11.0)
(timestamp = 2008-11-05) => concept:name=Upload milestone 1 (66.0/11.0)
(timestamp = 2008-12-24) => concept:name=Upload milestone 4 (71.0/12.0)
(timestamp = 2008-12-23) => concept:name=Upload milestone 4 (17.0/4.0)
(timestamp = 2008-12-16) => concept:name=Upload milestone 4 (4.0/1.0)
(timestamp = 2009-01-13) => concept:name=Upload exercise 9 (93.0/16.0)
(timestamp = 2009-01-14) => concept:name=Upload milestone 5 (97.0/8.0)
(timestamp = 2009-01-16) => concept:name=Upload milestone 5 (3.0/0.0)
(timestamp = 2009-01-06) => concept:name=Upload exercise 8 (106.0/6.0)
(timestamp = 2009-01-05) => concept:name=Upload exercise 8 (22.0/4.0)
(timestamp = 2008-12-30) => concept:name=Upload exercise 8 (3.0/0.0)
(timestamp = 2009-01-02) => concept:name=Upload exercise 8 (4.0/1.0)
(timestamp = 2008-12-01) => concept:name=Milestone Forum: ask question (17.0/7.0)
(timestamp = 2008-12-14) => concept:name=Milestone Forum: ask question (6.0/0.0)
(timestamp = 2008-11-27) => concept:name=Milestone Forum: ask question (11.0/3.0)
(timestamp = 2008-12-22) => concept:name=Milestone Forum: ask question (10.0/4.0)
(timestamp = 2009-01-09) => concept:name=Milestone Forum: ask question (7.0/2.0)
(timestamp = 2009-01-20) => concept:name=Upload milestone 6 (49.0/2.0)
(timestamp = 2009-01-18) => concept:name=Upload milestone 6 (45.0/0.0)
(timestamp = 2009-01-19) => concept:name=Upload milestone 6 (35.0/1.0)
(timestamp = 2009-01-15) => concept:name=Upload milestone 6 (8.0/2.0)
(timestamp = 2009-01-22) => concept:name=Upload milestone 6 (8.0/2.0)
(timestamp = 2008-10-28) => concept:name=Upload exercise 1 (139.0/12.0)
(timestamp = 2008-10-27) => concept:name=Upload exercise 1 (8.0/1.0)
(timestamp = 2008-10-26) => concept:name=Upload exercise 1 (8.0/2.0)
(timestamp = 2008-12-09) => concept:name=Upload exercise 7 (150.0/11.0)
(timestamp = 2008-12-05) => concept:name=Upload exercise 7 (3.0/1.0)
(timestamp = 2008-11-18) => concept:name=Upload exercise 4 (146.0/12.0)
(timestamp = 2008-11-17) => concept:name=Upload exercise 4 (33.0/11.0)
(timestamp = 2008-11-15) => concept:name=Upload exercise 4 (8.0/0.0)
(timestamp = 2008-11-04) => concept:name=Upload exercise 2 (177.0/22.0)
(timestamp = 2008-11-03) => concept:name=Upload exercise 2 (25.0/7.0)
(timestamp = 2008-11-02) => concept:name=Upload exercise 2 (21.0/9.0)
(timestamp = 2008-11-11) => concept:name=Upload exercise 3 (256.0/66.0)
 => concept:name=Exercise Forum: ask question (239.0/128.0)

Number of Rules : 44
```

Fig. 12. Result of JRip with pruning for the student data set at day level

not impose official deadlines, but could be best practices as tutors prefer to upload feedback simultaneously for a set of students. In this sense, $\mathcal{C}$ I and $\mathcal{C}$ II can be interpreted as "soft" constraints. In fact, there exists a regulation to grade within one month after the course, imposing a "hard" ISC then.

VI. Discussion

The evaluation has shown that the presented approach is reasonable and can be used to discover ISC candidates from a process execution log. Nevertheless, selected corner cases and limitations of the approach are discussed in the following.

Time granularity: The ISC examples discussed before refer to points in time. However, there are other examples that refer to time periods, for example, *When starting the read-out of 00:00 values 99% of all meters should be read out within 6 hours.* [9] Being able to detect ISC with time periods depends on the presence of log data with respect to lifecycle transitions of events, i.e., events distinguishing between start and end of a task.

ISC Complexity: What if the ISC consists of several parts like in the second example? This requires several runs of Alg. 2. If the ISC is known beforehand the number of runs r can be adapted to this (cmp. Example 2, Sect. V-A).

Multitude of Files: What should be done if for each instance-spanning attribute Alg. 1 generates a multitude of files? Resources could be grouped together to reduce the number of files that need to be investigated or timestamps could be coarsened. If this is not possible there is currently no other way than investigating each of the files.

Small Event Logs: As stated in [14] the problem with small event logs is "that not enough data is available to make reliable conclusions". What holds true in general for process mining can be also posing a threat to ISC mining. As suggested by [14] one option is to try out whether or not the available data is still sufficient to derive meaningful results. Other options could be to collect more data (if possible) or to derive the ISC manually together with domain experts.

VII. Related Work

The topic studied in this paper is situated at the intersection between process compliance (cf., for example, [3]) and process mining [2], i.e., it tackles the general question on how to discover constraints from a process execution log. Stating formalizations and characterizations for different constraint types is not the aim of the paper.

Constraints can be discovered by, e.g., mining a declarative process model or by aligning procedural models based on decision mining. Discovering declarative process models has been proposed by, e.g., [15] and [1]. Furthermore, ProM[8] plugins like the Declare Component, consisting of several tools for determining declarative process models [16] and the MINERful Plugin [17] were implemented for supporting users in mining declarative models.

Decision mining deals with discovering the decision rules connected with alternative branches. The approach was introduced in [4] and further developed in [18]. Both, mining declarative models and decision mining focus on intra-instance constraints and do currently not address instance-spanning constraints as it is the goal of this paper.

In general, little attention has been paid to constraints that span multiple instances so far. First approaches address instance batching [19], collecting ISC examples [9], as well as modeling [6] and verifying ISC [20]. All these approaches assume the existence of ISC or collect them from documents, but do not aim at discovering ISC from log information.

VIII. Conclusion and Future Work

This paper has provided an approach for checking and discovering ISC candidates from event logs. One assumption was that the ISC refer to the same process type. A definition for ISC decision points as well as an algorithm that can restructure the log such that a discovery of ISC candidates is possible was presented. The evaluation based on two artificial and one real-life log, together with expert interviews has shown the feasibility and applicability of the approach. Two use cases for the approach were described, i.e., conformance checking and discovery of ISC candidates from event logs. Future work will center on discovering ISC candidates spanning multiple processes. Moreover, the proposed algorithms will be optimized by developing stochastic methods for choosing the right time granularity and making recommendations on the relevance of events.

[8]http://www.promtools.org/doku.php?id=start

Acknowledgment

This work has been funded by the Vienna Science and Technology Fund (WWTF) through project ICT15-072.

References

[1] A. Burattin, M. Cimitile, F. M. Maggi, and A. Sperduti, "Online discovery of declarative process models from event streams," *IEEE Trans. Services Computing*, vol. 8, no. 6, pp. 833–846, 2015.

[2] W. M. P. van der Aalst, *Process Mining: Discovery, Conformance and Enhancement of Business Processes*. Springer, 2011.

[3] L. T. Ly, F. M. Maggi, M. Montali, S. Rinderle-Ma, and W. M. P. van der Aalst, "Compliance monitoring in business processes: Functionalities, application, and tool-support," *Inf. Syst.*, vol. 54, pp. 209–234, 2015.

[4] A. Rozinat and W. M. P. van der Aalst, "Decision mining in prom," in *Business Process Management*, 2006, pp. 420–425.

[5] M. Leitner, J. Mangler, and S. Rinderle-Ma, "Definition and enactment of instance-spanning process constraints," in *Web Information System Engineering*, 2012, pp. 652–658.

[6] W. Fdhila, M. Gall, S. Rinderle-Ma, J. Mangler, and C. Indiono, "Classification and formalization of instance-spanning constraints in process-driven applications," in *Business Process Management*, 2016, pp. 348–364.

[7] J. Warner and V. Atluri, "Inter-instance authorization constraints for secure workflow management," in *Access Control Models and Technol.*, 2006, pp. 190–199.

[8] R. Mrasek, J. A. Mülle, K. Böhm, M. Becker, and C. Allmann, "User-friendly property specification and process verification - A case study with vehicle-commissioning processes," in *Business Process Management*, 2014, pp. 301–316.

[9] S. Rinderle-Ma, M. Gall, W. Fdhila, J. Mangler, and C. Indiono, "Collecting examples for instance-spanning constraints," *CoRR*, vol. abs/1603.01523, 2016.

[10] L. T. Ly, C. Indiono, J. Mangler, and S. Rinderle-Ma, "Data transformation and semantic log purging for process mining," in *Advanced Information Systems Engineering*, 2012, pp. 238–253.

[11] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques, Second Edition*. Morgan Kaufmann Publishers Inc., 2005.

[12] W. W. Cohen, "Fast effective rule induction," in *Machine Learning*, 1995, pp. 115–123.

[13] J. Mangler and S. Rinderle-Ma, "Iupc: Identification and unification of process constraints," *CoRR*, vol. abs/1104.3609, 2011.

[14] W. e. a. van der Aalst, *Process Mining Manifesto*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 169–194.

[15] F. M. Maggi, A. J. Mooij, and W. M. P. van der Aalst, "User-guided discovery of declarative process models," in *2011 IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*, April 2011, pp. 192–199.

[16] F. M. Maggi, "Declarative process mining with the declare component of prom." in *BPM (Demos)*, 2013.

[17] C. Di Ciccio, M. H. Schouten, M. de Leoni, and J. Mendling, "Declarative process discovery with minerful in prom." in *BPM (Demos)*, 2015, pp. 60–64.

[18] F. Mannhardt, M. de Leoni, H. A. Reijers, and W. M. P. van der Aalst, "Decision mining revisited - discovering overlapping rules," in *Advanced Information Systems Engineering*, 2016, pp. 377–392.

[19] L. Pufahl, A. Meyer, and M. Weske, "Batch regions: Process instance synchronization based on data," in *Enterprise Distributed Object Computing*, 2014, pp. 150–159.

[20] C. Indiono, J. Mangler, W. Fdhila, and S. Rinderle-Ma, "Rule-based runtime monitoring of instance-spanning constraints in process-aware information systems," in *On the Move to Meaningful Internet Syst.*, 2016, pp. 381–399.