SVIPEX: A Web Service for Discovering and Visualizing Instance Spanning Constraints based on Process Execution Logs

Florian Stertz[1], Karolin Winter[1], Stefanie Rinderle-Ma[1,2]

[1]Research Group Workflow Systems and Technology, Faculty of Computer Science,
[2]Data Science@Uni Vienna, University of Vienna, Vienna, Austria
{firstname.lastname}@univie.ac.at

Abstract. Instance spanning constraints (ISC) realize controls on the execution of multiple process instances of one or several process types. Examples include synchronization at critical resources and authorization across several instances. ISC leave marks in process execution logs, e.g., process tasks having the same time stamp. As ISC are not always known and might change over time, process execution logs provide a valuable resource for mining ISC. Algorithms for ISC discovery have been proposed in previous work. SVIPEX implements these algorithms and provides a novel visualization for discovered ISC through a lightweight web service. This way ISC mining becomes accessible. SVIPEX maturity is demonstrated based on case studies from manufacturing and higher education.

Keywords: Instance Spanning Constraints, Process Discovery, Business Process Compliance

1 Introduction and Significance for the BPM Field

Digitalized compliance management includes the (semi-)automatic extraction of constraints from regulatory documents that can be automatically verified over business process models (design time) and process instances (run time). Constraints can refer to single process models and single process instances, but can be also used to establish controls across multiple process instances of one or multiple process types. The latter are called *Instance Spanning Constraints* (ISC) [1]. They are crucial in various domains, including logistics, production, and health care, and realize various applications such as batching [3] and queuing [4] of process instances. An example for ISC in the health care domain is: *"A patient is involved simultaneously in two different treatments, i.e., dermatology and diabetes. A list of drugs should not be taken during the dermatology treatment. The diabetes treatment should consider this list."* [6].

During run time, ISC constrain and control the execution of process instances by, for example, halting the execution of instances until a condition is met (cf. batching). As a consequence, ISC leave (implicit) marks in process execution logs that store the behavior of process instances. In previous work [6], several algorithms for discovering ISC from process execution logs have been proposed. A summary of these algorithms and further background is provided in Sect. 2. Providing (automatic) support for ISC discovery from process execution logs is crucial as ISC might not be explicitly known and – if they are known – they might change due to changes in the associated regulations. This demonstration paper describes the SVIPEX implementation of the ISC discovery algorithms proposed in [6] and provides a novel visualization for the discovered ISC. SVIPEX is realized as lightweight web service in order to increase its accessibility (cf. Sect. 2). SVIPEX maturity is demonstrated based on case studies from the manufacturing and higher education domains in Sect. 3. Potential users of SVIPEX are compliance officers, process analysts, auditors, and researchers. Further application scenarios and future perspectives are discussed in Sect. 4.

2 SVIPEX Innovation and Architecture

SVPIEX can be accessed via

`http://svipex.wst.univie.ac.at`

A tutorial, the source code as well as example data sets are available at

`http://gruppe.wst.univie.ac.at/projects/crisp/index.php?t=discovery`

The screencast is accessible via

`http://gruppe.wst.univie.ac.at/projects/crisp/screencasts/svipex.mp4`

SVIPEX targets the novel research topic of ISC discovery by providing the ISC discovery algorithms presented in [6] as a REST web service. Figure 1 depicts the SVIPEX architecture consisting of a frontend, backend and a Riddl application, based on the Ruby Gem Riddl[1] which serves for processing information between the frontend and the backend.

2.1 Frontend

The SVIPEX frontend consists of two components, based on HTML markup, JavaScript components, CSS and the jQuery Smart Wizard plugin[2]. The main page serves as start page and for displaying the results. The second page for uploading the dataset (one or several XES log files) and passing further information required for the process model and ISC mining processes.

Main Page. The main page is displayed when a user accesses the website (cf. Fig. 1). Clicking on “New Collection” leads to the upload page (see next paragraph) where process execution logs can be uploaded and parameters can be set. The main page also offers results for three data sets for ISC discovery, i.e., *Manufacturing, Printer* and *HigherEducationDomain* that were previously analyzed in [6]. Newly uploaded data sets will be displayed below these prepared examples. The main page enables to choose combinations of data sets on the left hand side and one of four implemented ISC discovery algorithms on the top

[1] `https://rubygems.org/gems/riddl`
[2] `http://techlaboratory.net/jquery-smartwizard`

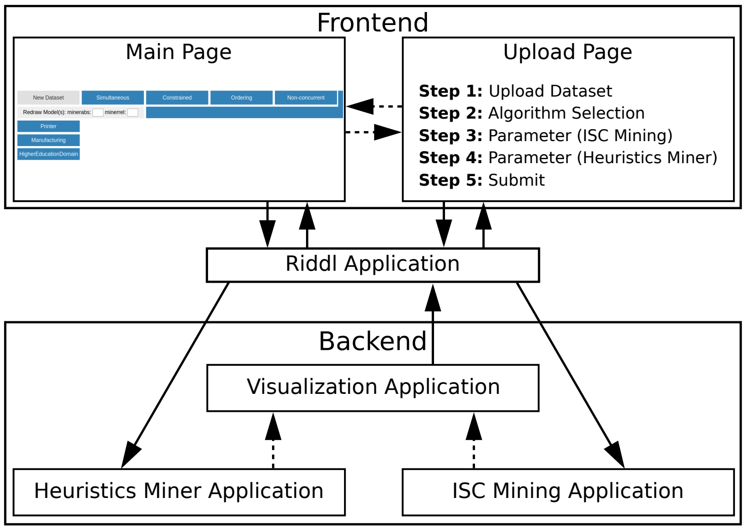
Frontend
Main Page
New Dataset
Simultaneous
Constrained
Ordering
Non-concurrent
Redraw Model(s): minerabs: minerrel:
Printer
Manufacturing
HigherEducationDomain
Upload Page
Step 1: Upload Dataset
Step 2: Algorithm Selection
Step 3: Parameter (ISC Mining)
Step 4: Parameter (Heuristics Miner)
Step 5: Submit
Riddl Application
Backend
Visualization Application
Heuristics Miner Application
ISC Mining Application

Fig. 1: SVIPEX Architecture

of the page. Section 2.2 provides background information on the ISC discovery algorithms. The combination of data set and algorithm that is currently selected is highlighted in bright blue on the main page. The main page displaying the results of the third ISC discovery algorithm for the manufacturing example is depicted in Fig. 2.

Upload Page. Uploading is partitioned into five steps to ease the process for the user. In the first step, the user needs to provide a name for the dataset and upload at least one process execution log in XES format[3]. The user can provide an attribute based on which traces are identified and can be automatically detected by SVIPEX under the assumption that there exists an event attribute that is unique in terms of values. The process execution log files need to fulfill the following requirements:

– *Concept names* for events, i.e., activity labels must be unique
– *Time stamps*
– *Unique identifiers* of traces or events if ISC span multiple process types
– *Life cycle transitions start*, *end* of an activity execution for IV.

These requirements are typically met by Process Aware Information System and represent standard extensions XES. In a second step, the user needs to select which ISC mining algorithms the user wants to execute. In the third step the values for the ISC mining parameters need to be selected by the user whereas we provide default values based on experience from previous papers. For details on the parameters see [6]. In the fourth step the absolute and relative threshold of edges for the Heuristics Miner need to be provided and the fifth step submits the provided information. Afterwards, the user is redirected to the main page.

[3] `http://www.xes-standard.org`

2.2 Backend

The backend consists of three applications implemented in Python 3. One application serves for mining the process models based on the Heuristics Miner [5]. The ISC mining application implements the ISC algorithms which are developed based on the following ISC categorization derived from a collection of 114 ISC examples (cf. [1]). ISC trace back to synchronizations which can be caused by

I simultaneous execution of events $\mapsto$ Algorithm 1
II constrained event execution based on data elements, time constraints or regularities $\mapsto$ Algorithm 2
III order of events $\mapsto$ Algorithm 3
IV non concurrent execution of events. $\mapsto$ Algorithm 4

The third application uses GraphViz[4] to combine the process models and the results of the ISC mining algorithms into an overall visualization. In particular one or several process models are aligned with ISC candidates. An ISC candidate is depicted as grey filled node and in the case of multiple process types a red dashed line indicates the connection between multiple ISC candidates. For Algorithm 2 the ISC candidate nodes are enriched with decision rules which were detected by the decision tree classifier of the Python package scikit-learn [2].

3 Maturity

SVIPEX was tested on real-life execution logs from the *Manufacturing* and *Higher Education Domain* domain an on an artificial *Printer* log in [6]. In the following and in the associated tutorial and screencast, SVIPEX is demonstrated based on the real-life *Manufacturing* log. It consists of nine different process models and several events are enriched with extensive data elements prescribing, e.g., positions of machines during the execution of the processes. The process execution logs consist altogether of 2546 events (11.58 MB) and SVIPEX took on average 46.707 seconds to complete the mining process. Figure 2 depicts a screenshot of SVIPEX containing the results for the order of events (Algorithm 3) with $\gamma_3 = 1, \kappa = 0$. The red dashed lines connecting events describe the order relations between these events, i.e, the spawning of entire processes by certain tasks. The results of this case study were evaluated by domain experts resulting in a precision of $\frac{3}{4}$ and recall of $\frac{3}{7}$ whereas the retrieved false positive results emerge due to physical conditions which are not representing ISC but still pose restrictions on the execution order of processes or tasks, i.e., SVIPEX could provide further insights into the log files.

4 Conclusion

Due to the tremendous amount of data that is stored in process execution logs providing automated tool support for analyzing this data becomes mandatory.

[4] `https://www.graphviz.org/`

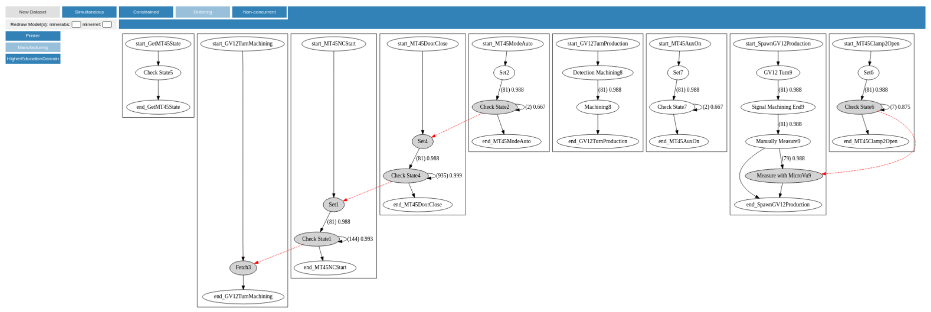

Fig. 2: Screenshot of SVIPEX Depicting the Results for the Manufacturing Case Study for Algorithm 3

Moreover, processes are highly connected and need to obey to constraints that span one or several process types. Since current process mining algorithms and tools are not capable of detecting such types of constraints, we have presented SVIPEX, a graphical user interface provided as lightweight web service. The case studies have proven SVIPEX feasibility and usefulness. As future work we plan to integrate additional process mining algorithms and further enhance the visualization of ISC candidates.

Acknowledgment

This work has been partly funded by the Austrian Research Promotion Agency (FFG) via the "Austrian Competence Center for Digital Production" (CDP) under the contract number 854187. This work has been supported by the Pilot Factory Industry 4.0, Seestadtstrasse 27, Vienna, Austria.

References

1. Fdhila, W., Gall, M., Rinderle-Ma, S., Mangler, J., Indiono, C.: Classification and formalization of instance-spanning constraints in process-driven applications. In: Business Process Management. pp. 348–364 (2016)
2. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12**, 2825–2830 (2011)
3. Pufahl, L., Meyer, A., Weske, M.: Batch regions: Process instance synchronization based on data. In: Enterprise Distrib. Object Comp. pp. 150–159 (2014)
4. Senderovich, A., Weidlich, M., Gal, A., Mandelbaum, A.: Queue mining for delay prediction in multi-class service processes. Inf. Syst. **53**, 278–295 (2015)
5. Weijters, A., van Der Aalst, W.M., De Medeiros, A.A.: Process mining with the heuristics miner-algorithm. Technische Universiteit Eindhoven, Tech. Rep. WP **166**, 1–34 (2006)
6. Winter, K., Stertz, F., Rinderle-Ma, S.: Discovering instance and process spanning constraints from process execution logs. Inf. Syst. **89**, 101484 (2020). https://doi.org/10.1016/j.is.2019.101484