A Tool for Generating Event Logs from Multi-Perspective Declare Models

Vasyl Skydanienko[1], Chiara Di Francescomarino[2], Chiara Ghidini[2], and Fabrizio Maria Maggi[1]

[1] University of Tartu, Tartu, Estonia
{skydanienko,f.m.maggi}@ut.ee
[2] FBK-IRST, Trento, Italy
{dfmchiara,ghidini}@fbk.eu

Abstract. The availability of event logs with specific characteristics is nowadays one of the challenges of the process mining field in order to be able to validate and evaluate the proposed techniques. Although some effort has been done towards the generation of synthetic logs from imperative models, the generation of event logs starting from declarative models is still a very young research stream. Few works have focused on the generation of event logs taking into account the control flow perspective or focusing on a particular data attribute as the resources carrying out the activities. However, none of them, so far, has dealt with the generation of event logs from declarative models providing full support for data attributes. `MP-Declare Log Generator` is a tool for the generation of event logs starting from Multi-Perspective Declare models, a language for expressing declarative constraints on both control flow and data. The implementation of the tool is based on the Alloy model checker and is provided as a standalone desktop application.

Keywords: Multi-perspective declarative models, Log generation, Model checker

1 Introduction

The growth of the techniques in the process mining field and, in particular, of a number of more and more accurate and efficient discovery algorithms has led to an increasing need to evaluate and compare them, thus demanding for event logs (and corresponding gold-standard models) able to demonstrate their correctness and scalability. Although some real-life logs are publicly available, their number is still quite limited. Most companies, indeed, are not willing to share their own event logs, e.g., due to privacy issues or because they depend on third parties. Moreover, real-life logs are not always the best choice for evaluating process mining approaches, as they do not offer control over different parameters such as the log size, the length of the traces or the amount of noise. Having control over these characteristics is of the utmost importance to be able to evaluate a process mining algorithm in a complete and predictable way, e.g., in order to be

F. Casati et al. (Eds.): Proceedings of the Dissertation Award and Demonstration, Industrial Track at BPM 2018, CEUR-WS.org, 2018.

able to understand how much robust is the mining algorithm with respect to the noise or scalable with respect to the log size.

Most of the modeling languages and tools for process mining mainly focus on the control flow perspective of a process. However, in real life, activities (and execution traces) may contain data, as for example who is executing the activity, what type of resources was used or what amount was paid (e.g., in a selling process). This makes evaluating process mining algorithms using real-life logs even harder, because the amount and type of data cannot be changed, and hence different aspects of the algorithm cannot be evaluated separately.

While some work on the generation of event logs starting from imperative models has been recently carried out [4], very few tools are available for generating artificial logs from declarative process models. Though some of them support resources, none of them allows for using arbitrary constrained data and no tools support the generation of logs from the full set of constraints provided by Multi-Perspective Declare (MP-Declare), a language presented in [5] for expressing declarative constraints on both control flow and data. In this paper, we aim at filling this gap by introducing `MP-Declare Log Generator`, a tool that generates event logs from MP-Declare models by leveraging the Alloy model checker.

The generation of logs from declarative process models is not an easy task, and it becomes even harder when providing data support. It indeed poses two interesting challenges. First, since there are no well-defined execution paths like in imperative models, different constraints might influence and contradict each other, and some models do not even have valid finite traces. In these scenarios, it might be really hard to find traces for some models due to the combinatorics of the problem, so it is important for the log generator to work efficiently. Second, when generating a log, it is preferable to get diverse traces rather than similar ones so that the log contains more useful information about possible process execution patterns. `MP-Declare Log Generator` aims at facing both these challenges by efficiently generating event logs in a reasonable amount of time and guaranteeing a high heterogeneity of the generated traces.

2 Multi-Perspective Declare

In this section, we illustrate MP-Declare, a Multi-Perspective version of Declare introduced in [5]. This semantics is expressed in Metric First-Order Linear Temporal Logic (MFOTL). We describe here the semantics informally and we refer the interested reader to [5] for more details. To explain the semantics, we have to introduce some preliminary notions.

The first concept we use here is the one of *payload* of an event. Consider, for example, that the execution of an activity BookAccomodation (B) is recorded in an event log and, after the execution of B at timestamp τ_B, the attribute $HotelName$ and $Price$ have values $Paradise$ and 90. In this case, we say that, when B occurs, two special relations are valid $event$(B) and $p_B(Paradise, 90)$. In the following, we identify $event$(B) with the event itself B and we call ($Paradise$, 90) the *payload* of B.

The standard semantics of Declare is extended by requiring two additional conditions on data, i.e., the *activation condition* φ_a and the *correlation condition* φ_c. As an example, we consider the response constraint "activity BookAccomodation is always eventually followed by activity CollectTickets" having BookAccomodation as activation and CollectTickets as target. The activation condition is a relation over the attributes of the payload of the activation that must be valid when the activation occurs. If the activation condition does not hold, the constraint is not activated. The activation condition has the form $p_A(x) \wedge r_a(x)$, meaning that when A occurs with payload x, the relation r_a over x must hold. For example, we can say that whenever BookAccomodation occurs, and the price is lower than 50 euros, eventually the tickets are collected. In case BookAccomodation occurs but the amount is higher than 50 euros, the constraint is not activated. The correlation condition is a relation that must be valid when the target occurs. It has the form $p_B(y) \wedge r_c(x, y)$, where r_c is a relation over the payloads of both A and B. For example, we can say that whenever BookAccomodation occurs for a certain application ID, eventually BookTrasport must follow for the same application ID.

3 Multi-Perspective Declare Log Generator

Synthetic log generators are essential for different types of tasks. They allow the evaluation of the correctness and scalability of process mining algorithms [3,6,10], the translation between process modeling languages - Model to Model Translation (M2MT) [2,1], the visualization of declarative processes [9]. Though log generation tools for declarative process models are needed in all the above scenarios, data support in the currently existing tools is not present or it does not cover the entire semantics of MP-Declare.

`MP-Declare Log Generator` aims at filling this gap by leveraging the Alloy model checker [8]. `MP-Declare Log Generator` takes as input an MP-Declare model, defined using an ad-hoc user-friendly syntax and follows the steps of the procedure reported in Fig. 1.

In the first step, the MP-Declare model is parsed and separate statements for activities, data, bindings of data to activities and constraints are collected. In the second step - the preprocessing step - since Alloy does not support numeric variables, intervals for numeric values are generated according to the constraints. In the third step, the Alloy code corresponding to the MP-Declare model is generated. After this, the Alloy analyzer is run. The analyzer generates the CNF (conjunctive normal form) for the model, which is then solved using a SAT (boolean satisfiability problem) solver [7]. Once the Alloy solution is computed, in the fifth step, the trace can be extracted. Finally, in the post-processing stage, the categorical intervals returned as solutions by Alloy for the numeric values are replaced with actual numbers randomly selected within the intervals. We then move to the next solution and repeat the last two steps until the required amount of traces is collected. When all traces are generated, they are saved in a XES (`http://www.xes-standard.org/`) file.

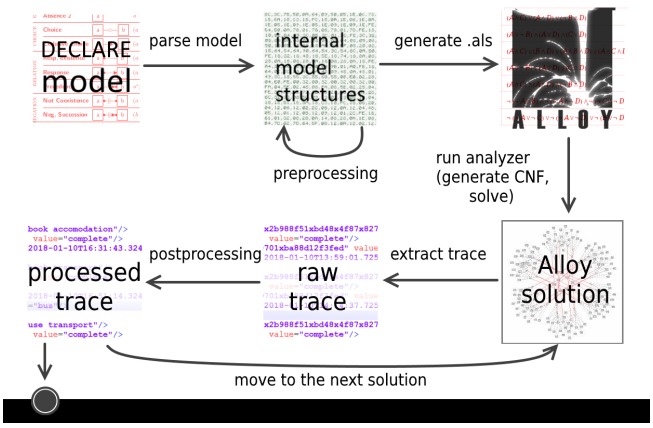

Fig. 1. The logical steps followed by `MP-Declare Log Generator` for the generation of event logs

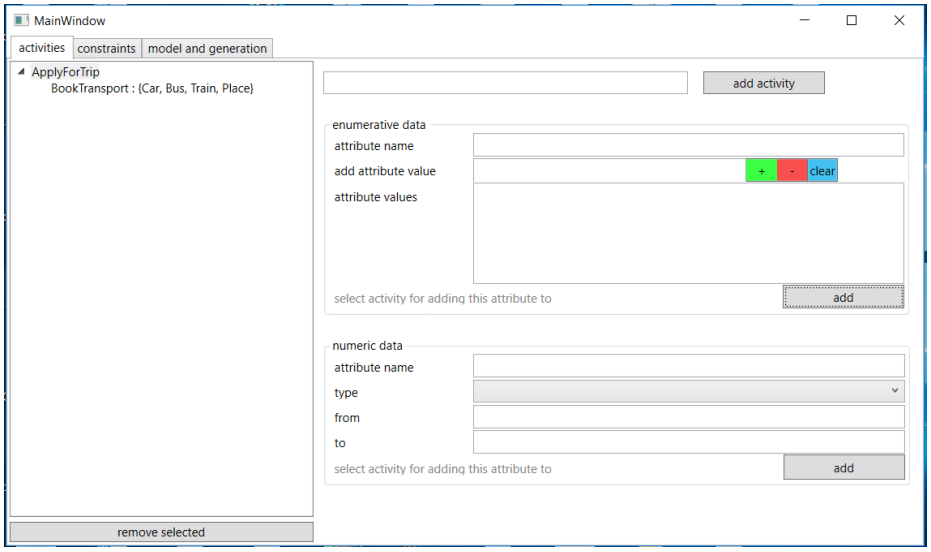

Fig. 2. A screenshot of the `MP-Declare Log Generator` standalone tool

3.1 Implementation

`MP-Declare Log Generator` has been implemented as a standalone desktop application.[1] The tool allows the users to define an MP-Declare model by specifying through a simple interface, activities, attributes (both categorical and numerical), the range of values that the attributes can take and the constraints. Once

[1] The tool can be downloaded from `https://tinyurl.com/y9uqnfok`. A video of this demo is available at `https://youtu.be/e4q_ThtY70E`.

the model has been defined, it can be used for the generation of the log. Several options are available for the log generation. For example, it is possible to specify minimum and maximum trace length, number of traces, to force the occurrence of at least an activation of each constraint in each trace, to generate negative traces. To this purpose, `MP-Declare Log Generator` provides three tabs: one for the definition of the activities, the definition of the attributes and their values and the association between activities and attributes; the second one for the definition of constraints; and the third one for the actual log generation. Fig. 2 shows a screenshot of the the first tab of the tool in which it is possible to specify activities and attributes.

4 Tool Maturity and Potential

Although the tool is still at an initial stage of development and can be improved with further features, it already offers to users (and to the BPM community) a very useful instrument for the generation of synthetic logs. In particular, `MP-Declare Log Generator` has revealed to be mature enough to face the efficiency and the heterogeneity challenges characterizing this type of tools (see Section 1). Indeed, empirical results have shown that `MP-Declare Log Generator` is able to deal with logs of 1000 traces with an execution time of the order of seconds and that the diversity of the generated logs is comparable with the diversity of real-life logs.

References

1. Ackermann, L., Schönig, S., Jablonski, S.: Towards simulation- and mining-based translation of process models. In: EOMAS. vol. 272, pp. 3–21 (2016)
2. Ackermann, L., Schönig, S., Jablonski, S.: Towards simulation- and mining-based translation of resource-aware process models. In: BPM Workshops. vol. 281, pp. 359–371 (2016)
3. Ackermann, L., Schönig, S., Jablonski, S.: Simulation of multi-perspective declarative process models. In: Dumas, M., Fantinato, M. (eds.) BPM Workshops. pp. 61–73. Cham (2017)
4. Burattin, A.: PLG2: multiperspective process randomization with online and offline simulations. In: BPM Demo. pp. 1–6 (2016)
5. Burattin, A., Maggi, F.M., Sperduti, A.: Conformance checking based on multi-perspective declarative process models. Expert Syst. Appl. 65, 194–211 (2016)
6. Di Ciccio, C., Bernardi, M.L., Cimitile, M., Maggi, F.M.: Generating event logs through the simulation of declare models. In: EOMAS. pp. 20–36 (2015)
7. Eén, N., Sörensson, N.: An extensible SAT-solver. In: Theory and Applications of Satisfiability Testing. pp. 502–518 (2004)
8. Jackson, D.: Software Abstractions: Logic, Language, and Analysis. The MIT Press (2006)
9. Laurent, Y., Bendraou, R., Baarir, S., Gervais, M.P.: Planning for declarative processes. pp. 1126–1133. SAC (2014)
10. Schönig, S., Cabanillas, C., Jablonski, S., Mendling, J.: Mining the organisational perspective in agile business processes. In: Enterprise, Business-Process and Information Systems Modeling. pp. 37–52 (2015)