Declarative Mining of Business Processes via ASP

(Discussion/Short Paper)

Antonio Ielo[1,*], Luigi Pontieri[2] and Francesco Ricca[1]

[1]*University of Calabria, Italy*
[2]*ICAR-CNR, Italy*

Abstract
Declarative process discovery algorithms aim to identify a subset of relationships between process' activities ("constraints") that implicitly define the acceptable behavior of a process given a bag of its execution traces. Declarative process models have proven effective in dealing with loosely-structured processes and complex domains. This paper proposes a declarative approach to mine constraints over a set of process variants that satisfy user-defined optimization criteria and preferences. We implement the approach using Answer Set Programming, a declarative paradigm developed in logic programming and non-monotonic reasoning.

Keywords
Answer Set Programming, Declarative process discovery, Declare

Process Discovery (PD), one of the main Process Mining tasks, is concerned with algorithmically finding ("mining") a model for an unknown process by using its execution traces as input [1]. There exist multiple process modeling languages to define process models. However, one can identify two main categories of process models: procedural (or imperative) and declarative. Procedural approaches are most suited to stable, well-established processes, while they are lackluster when applied to complex or poorly specified, loosely-structured ones [2]. Procedural models attempt to describe the desired behavior of the process explicitly. For example, the most common formalism to define procedural process models is the Petri net - an automaton-like computational model whose accepted language should be the set of possible process traces. On the other hand, declarative models implicitly describe a process by a set of constraints process' traces must not violate to be considered compliant. That is, while procedural models attempt to describe processes explicitly, declarative models shrink the space of allowed behavior the process can exhibit [2].

One of the most common modeling languages for expressing declarative process models is Declare [2], a collection of constraint templates whose semantics is rooted in linear temporal logic over finite traces (LTL_f; [3, 4]). Declare templates enforce temporal conditions between process' activities; "instantiations" of templates (e.g., the "binding" of specific activities to the template's parameters) are called *constraints*. A Declare model, also called a "map", consists of

PMAI@IJCAI22: International IJCAI Workshop on Process Management in the AI era, July 23, 2022, Vienna, Austria
*Corresponding author.
antonio.ielo@unical.it (A. Ielo); luigi.pontieri@icar.cnr.it (L. Pontieri); ricca@mat.unical.it (F. Ricca)

a set of Declare constraints. Classic process discovery techniques for Declare maps perform two steps [5]: a generation phase that finds candidate constraints to include in the model and a pruning phase that considers quality measures such as support, confidence, and interest factor of the candidate constraints [6]. In general, due to the exponential number of candidate models, it is infeasible to fully explore the generation phase's search space, even limiting its scope to constraints above a fixed support threshold. Hence, currently available tools to mine Declare maps [7, 8] let users choose a subset of templates to be mined or a subset of activities to mine. The output Declare map can be later repaired (e.g., relaxing, removing constraints from the model) or enhanced (e.g., adding constraints to adapt to a new log of traces) as customary in Process Mining. The declarativeness of the Declare modeling language allows analysists to inspect or even modify models manually [9]. Although efficient implementations of the pruning phase partially address issues like removing vacuously satisfied, subsumed, redundant or conflicting constraints [10, 11], Declare maps often require post-processing steps to ensure desired properties.

One might wonder whether a software system could assist the analyst while performing PD of Declare maps by singling out a subset of the mined patterns satisfying some user-specified criteria. To this end, several possible selection criteria and optimality conditions are already available in the literature [6, 10]. However, the last word on the expected results should always be in the hands of the analyst, that ideally would love to be left free of defining, trying, and finally choosing the most appropriate specification for the constraints to be mined. Thus, we expect such a system to benefit the analyst [12]. Having this in mind, we observe that we can model the problem of mining Declare maps as a combinatorial optimization problem, where the optimization criteria should be selected, specified, and combined by the analyst.

These kind of tasks are suitable for being solved in a declarative way by resorting to Answer Set Programming (ASP; [13]). ASP is a fully-declarative problem-solving paradigm developed in logic programming and non-monotonic reasoning. Indeed, ASP proved to be a viable solution for representing and solving many classes of combinatorial optimization problems since ASP features both a standardized first-order language with declarative semantics [14] and efficient implementations [15]. The practical applicability of ASP is evident in the effective development of several academic and industrial applications [16]. The idea of resorting to ASP for solving problems related to process mining is not entirely new. Recently, Chiarello and co-authors [17] proposed tackling the tasks of conformance checking of Declare maps and generating logs compliant to Declare maps using Answer Set Programming. Their approach models the LTL$_f$ formulae definitions of Declare constraints via deterministic finite automa [3]. Hence, conformance checking amounts to simulating the automata runs over the log's traces. Similarly, the generation of compliant traces amounts to searching for a string accepted by all (the automata corresponding to) Declare constraints in the model - or rejected by at least one of the constraints in the case of non-compliant traces.

Going forward in this direction, we aim at implementing a versatile system for mining Declare maps via ASP. Users can express their optimization criteria, preferences, and domain knowledge through logic programs, and the system will compute - for a given input log - the optimal constraints to include in the model to meet users' specifications. The main language feature for expressing optimization problems via ASP is the *weak constraint*. Weak constraints assign a cost to each answer set, on multiple priority levels. Stacking weak constraints on different priority levels allows users to express complex preference criteria. ASP systems optimize weak constraints in a multi-level fashion, in decreasing order of priority level. This approach is meant to prune the space of possible Declare candidate maps by exploiting the user's preferences as an objective function. The approach is suited to integrate well-known vacuity-detection techniques [10] and constraints subsumption hierarchies [11] to prune the search space and will allow analysts to include domain knowledge to guide the mining while reducing the need for post-processing of mined maps.

Prototype development status and future work. We report that we have developed a prototype system implementing the proposed approach. In particular, the systems encodes an input XES log into a set of ASP facts and uses a logic program to compute which Declare constraints hold on the log's simple traces. This information is exploited to extract the *safe model*, the set of Declare constraints that are satisfied by every trace in the log. User-defined criteria are then used to *extend* the safe model with Declare constraints that are not safe. As outlined previously, users can write logic programs - featuring weak constraints - to express their preferences. Examples of easily-expressible criteria include stating that certain traces in the log should be rejected (or accepted) and that certain constraint types should be preferred (or completely avoided) between some activities. As previously stated, the user can compose these criteria and assign them different priorities. These preferences are taken into account when extending the safe model with unsafe constraints. Thus user's specifications become the reason to include unsafe behavior in the model. The optimal answer sets obtained as a result are interpreted as Declare maps, for the sake of integration with existing Declare toolsets. Furthermore, since the input to the optimization logic program is a set of facts, any existing mining tool can substitute the mining module in the proposed framework.

Preliminary results on standard logs seem encouraging in terms of performance and flexibility. The time to obtain optimal pattern sets is in the order of minutes on a common laptop. We plan to perform thorough experiments on benchmark logs to outline the strengths and weaknesses of our proposal. Ongoing and future development efforts are devoted to supporting different optimization criteria and to including redundancy detection in the mining phase. We will also investigate the opportunity of supporting "heterogeneous models" combining constraints expressed via multiple formalisms (i.e., not only expressed in pure ASP or Declare, for example we could target DCR Graphs [18]), subject to shared, user-defined optimization criteria.

Acknowledgments. This work is partially supported by the Italian Ministry of Research under PRIN project "exPlaInable kNowledge-aware PrOcess INTelligence" (PINPOINT), CUP H23C22000280006.

References

[1] W. M. P. van der Aalst, M. Pesic, H. Schonenberg, Declarative workflows: Balancing between flexibility and support, Comput. Sci. Res. Dev. 23 (2009) 99–113.

[2] M. Pesic, H. Schonenberg, W. M. P. van der Aalst, DECLARE: full support for loosely-

structured processes, in: 11th IEEE Intl. Enterprise Distributed Object Computing Conference (EDOC 2007), IEEE Computer Society, 2007, pp. 287–300.

[3] G. De Giacomo, R. De Masellis, M. Montali, Reasoning on LTL on finite traces: Insensitivity to infiniteness, in: Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, AAAI Press, 2014, pp. 1027–1033.

[4] V. Fionda, A. Guzzo, Control-flow modeling with declare: Behavioral properties, computational complexity, and tools, IEEE Trans. Knowl. Data Eng. 32 (2020) 898–911.

[5] C. Di Ciccio, M. Mecella, A two-step fast algorithm for the automated discovery of declarative workflows, in: IEEE Symposium on Computational Intelligence and Data Mining, CIDM 2013, Singapore, 16-19 April, 2013, IEEE, 2013, pp. 135–142.

[6] A. Cecconi, G. De Giacomo, C. Di Ciccio, F. M. Maggi, J. Mendling, Measuring the interestingness of temporal logic behavioral specifications in process mining, Inf. Syst. 107 (2022) 101920.

[7] M. Westergaard, F. M. Maggi, Declare: A tool suite for declarative workflow modeling and enactment, in: Proceedings of the Demo Track of the Nineth Conference on Business Process Management 2011, volume 820 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2011, pp. 1–5.

[8] C. Di Ciccio, M. H. M. Schouten, M. de Leoni, J. Mendling, Declarative process discovery with minerful in prom, in: Proceedings of the BPM Demo Session 2015, (BPM 2015), volume 1418 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2015, pp. 60–64.

[9] C. Corea, S. Nagel, J. Mendling, P. Delfmann, Interactive and minimal repair of declarative process models, in: Business Process Management Forum - BPM Forum 2021, Proceedings, volume 427 of *LNBIP*, Springer, 2021, pp. 3–19.

[10] C. Di Ciccio, F. M. Maggi, M. Montali, J. Mendling, On the relevance of a business constraint to an event log, Inf. Syst. 78 (2018) 144–161.

[11] C. Di Ciccio, F. M. Maggi, M. Montali, J. Mendling, Resolving inconsistencies and redundancies in declarative process models, Inf. Syst. 64 (2017) 425–446.

[12] C. Haisjackl, I. Barba, S. Zugal, P. Soffer, I. Hadar, M. Reichert, J. Pinggera, B. Weber, Understanding declare models: strategies, pitfalls, empirical results, Softw. Syst. Model. 15 (2016) 325–352.

[13] G. Brewka, T. Eiter, M. Truszczynski, Answer set programming at a glance, Commun. ACM 54 (2011) 92–103.

[14] M. Gelfond, V. Lifschitz, Classical negation in logic programs and disjunctive databases, New Gener. Comput. 9 (1991) 365–386.

[15] M. Gebser, N. Leone, M. Maratea, S. Perri, F. Ricca, T. Schaub, Evaluation techniques and systems for answer set programming: a survey, in: J. Lang (Ed.), Proceedings of the 26th Intl. Joint Conference on Artificial Intelligence, IJCAI 2018, ijcai.org, 2018, pp. 5450–5456.

[16] E. Erdem, M. Gelfond, N. Leone, Applications of answer set programming, AI Mag. 37 (2016) 53–68.

[17] F. Chiariello, F. M. Maggi, F. Patrizi, Asp-based declarative process mining, in: Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, AAAI Press, 2022, pp. 5539–5547.

[18] S. Debois, T. T. Hildebrandt, P. H. Laursen, K. R. Ulrik, Declarative process mining for DCR graphs, in: A. Seffah, B. Penzenstadler, C. Alves, X. Peng (Eds.), Proceedings of the Symposium on Applied Computing, SAC 2017, ACM, 2017, pp. 759–764.