Automata-Based Temporal Reasoning in Answer Set Programming with Application to Process Mining

Francesco Chiariello

University of Naples Federico II, Italy

Abstract

We present a new approach for solving temporal problems using Answer Set Programming (ASP), which exploits the automata representation of temporal specifications. This approach is then used to solve key problems from Process Mining. The contributions of the paper are manifold. Firstly, for the Temporal Logics community, it provides a tool to perform temporal reasoning. Secondly, for the ASP community, it offers a method to intuitively handle time. Finally, for the Process Mining community, it provides both tools and methods for analyzing event logs.

Keywords

Answer Set Programming, Automata Theory, Process Mining, Temporal Logics

1. Introduction

Answer Set Programming (ASP) [1] is a declarative problem solving approach that has become very popular in recent years. This is partly due to the development of efficient ASP systems such as DLV [2] and *clingo* [3]. Here, a problem is modeled as a logic program that is then fed into an ASP system. The system computes the *answer sets* of the program, each corresponding to a different solution to the problem.

In order to use ASP systems to solve problems involving temporal specifications, the idea we propose [4] is to use the well-known relationship between finite-state automata and $\text{LTL}_f/\text{LDL}_f$ formulae [5], stating that it is possible to construct an automaton that accepts exactly the traces satysfing the formula. In fact, one can represent $\text{LTL}_f/\text{LDL}_f$ formulae in an ASP program simply by encoding the corresponding automata. In this way, checking whether a trace satisfies the specifications reduces to checking whether the automata accept such a trace, which is easily done in ASP. There are many advantages to considering ASP. First, it provides a clear and concise syntax, inspired by Logic Programming and Prolog, to model problems. Second, the minimality of its semantics makes it very efficient (compared to SAT) in solving reachability problems, making ASP a natural choice for our techniques.

Process Mining (PM) [6**?**] is the research area at the intersection of Business Process Management [7] and Data Mining [8]. It studies methods and techniques for analyzing event logs to extract information related to the processes that generate such logs. An *event log* is

OVERLAY 2023: 5th Workshop on Artificial Intelligence and Formal Verification, Logic, Automata, and Synthesis, November 7, 2023, Rome, Italy

francesco.chiariello@unina.it (F. Chiariello)

https://www.francescochiariello.me/ (F. Chiariello)

a collection of *process traces*, i.e. sequences of events, where an event contains information about the business activity being performed. A *process* is a collection of traces achieving a desired business goal, of which event logs constitute the observed traces. Processes are modeled using different formalisms. Standard imperative process models are Petri nets [9, 10] and BPMN [11, 12]. These models tend to over-constrain the process. Indeed, all models whose traces satisfy some properties of interest are considered acceptable. In the case of declarative specifications [13], it is assumed that the properties directly represent the model. In this way, all the traces satisfying the properties are assumed to be part of the model (and nothing more). Declarative specifications are typically expressed in DECLARE [14], LTL_f [15], or LTL_p [16].

Automata are a possible choice for process models that have gained increasing attention in recent years. The main reason for this is their relation to temporal logics, which makes automata easy to define and understand while preserving all the advantages of a procedural representation. Indeed, if we think of a process as a set of process traces, that is, as event sequences constituting a formal language, finite-state automata are a natural choice for modeling processes. In this paper, we show how to encode automata in ASP together with various Declarative Process Mining problems (that is, PM problems where processes are represented using declarative specifications). The problems are then solved using ASP to simulate the run of the traces over the automata.

The contributions of the presented approach are manifold and benefit different communities:

- For the Temporal Logics community, it provides a tool (an ASP system) to perform temporal reasoning;
- For the ASP community, it offers a method (based on automata) to intuitively handle time;
- For the Process Mining community, it provides both tools and methods for analyzing event logs.

2. Approach

Let's consider a problem involving temporal specifications that admit a finite-state automaton representation, i.e. for which there exists an automaton that accepts all and only the traces satisfying the specification. Our approach, introduced in [17], consists of the following steps:

1. Convert the temporal specifications into automata;
2. Represent the automata into ASP;
3. Represent the traces into ASP;
4. Model the problem in ASP by adding generation and test rules;
5. Check the acceptance of traces by simulating the automata run over them.

Regarding the first point, this can be done in a pre-processing step, with available tools whose choice depends on the logical formalism used. For example, for LTL_f/LDL_f specifications one can use the state-of-the-art tool Lydia[1] [18]. In this paper, we are interested in the application of such an approach to problems in Declarative Process Mining. Since, in this context, the traces of interest are process traces, the tool employed is LTLp2DFA [19].

[1] https://github.com/whitemech/lydia

It should be noted that the automata-based approach was independently proposed in [20, 21] for Temporal ASP [22, 23]. While they compare different automata representations and conversion algorithms on toy examples [24] with the aim of determining the best representation, their work lacks an experimental evaluation of real-world problems to show the feasibility and scope of the approach.

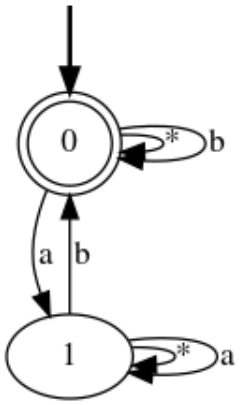

Figure 1: Automaton of Response template

```
automaton(s0,a,s1).
automaton(s1,b,s0).
automaton(s0,b,s0).
automaton(s0,"*",s0).
automaton(s1,a,s1).
automaton(s1,"*",s1).
initial(s0).
accepting(s0).
```

Listing 1: ASP encoding of the Response template

Figure 1 shows the automaton corresponding to the DECLARE template $Response(a, b)$ which is satisfied when 'every time activity a is performed, it is eventually followed by activity b'. This corresponds to the LTL$_p$ formula $\mathbf{G}(a \rightarrow \mathbf{F}b)$. The ASP encoding is shown in Listings 2. Here, a and b represent placeholders for the activation and target activity of the Response template, while the asterisk stands for any other activity.

```
state(S,0) :- initial(S).
state(S2,T) :- state(S1,T-1),automaton(S1,A,S2),trace(A,T).

sat(T) :- state(S,T),accepting(S).
```

Listing 2: ASP rules to update the current automaton state and to track the formula's satisfaction

3. Application

In this section, we describe various PM problems and show how the approach described above can be applied to them. For details on the encodings, the experimental evaluation, and the comparison with the state-of-the-art tools, the reader is referred to [17].

Log Generation is the problem of generating a set of process traces, of some given length, satisfying an input model. *Conformance Checking* is the problem of checking whether the traces

of a log satisfy a given input model. Finally, *Query Checking* is the problem of finding properties of a process, by checking constraint templates, i.e., formulae with variables (the *queries*), against the event log of the process. For a declarative model, these problems can be easily solved with ASP once the automata corresponding to the model are available.

For log generation, we add the ASP generation rule

```
{trace(A,T):activity(A)}=1 :- time(T).
```

for guessing the candidate answer set corresponding to a trace, and a test rule to check whether the trace is accepted by the automata. The case of conformance checking is even simpler since no generation rule is required: the traces are already given. We just need to test whether they are accepted. For query checking, we use the generation rule

```
{assignment(V,A):activity(A)} = 1 :- var(V).
```

to guess the instantiation of variables to activities and then check whether the input log satisfies the formula obtained.

4. Conclusion

The problems we considered are relatively simple and are intended to demonstrate the potential of the approach. However, the results were so satisfactory that the authors of the Declarative PM toolkit RuM [25] integrated it into their application for log generation. Following our approach, [26] proposes to use it for Process Discovery (i.e., finding a model of the log) while using the ASP optimization capabilities to also take into account user preferences. Optimization capabilities can also be used to solve other complex PM problems such as Trace Alignment (i.e. modifying a log to make it compliant with a given model), which can be formulated as cost-optimal planning [27]. Finally, we stress that, while we have considered PM problems, there is no reason to limit ourselves to this particular domain, since the approach can be virtually applied to any problem involving temporal specifications that admit automata representation.

Acknowledgments

The paper is based on my PhD Thesis 'Automata-Theoretic Techniques for Declarative Process Mining' (2023). I thank my advisor Fabio Patrizi and my coauthor Fabrizio Maggi. Work supported by the CeSMA project "Tecnologie abilitanti per il volo in formazione di piccole piattaforme aerospaziali WP2", and the project "Borgo 4.0" POR Campania FESR 2014-2020.

References

[1] G. Brewka, T. Eiter, M. Truszczynski, Answer set programming at a glance, Commun. ACM 54 (2011) 92–103. URL: https://doi.org/10.1145/2043174.2043195. doi:10.1145/2043174.2043195.

[2] M. Alviano, F. Calimeri, C. Dodaro, D. Fuscà, N. Leone, S. Perri, F. Ricca, P. Veltri, J. Zangari, The ASP system DLV2, in: M. Balduccini, T. Janhunen (Eds.), Logic Programming and Nonmonotonic Reasoning - 14th International Conference, LPNMR 2017, Espoo, Finland, July 3-6, 2017, Proceedings, volume 10377 of *Lecture Notes in Computer Science*, Springer, 2017, pp. 215–221. URL: https://doi.org/10.1007/978-3-319-61660-5_19. doi:10.1007/978-3-319-61660-5_19.

[3] M. Gebser, R. Kaminski, B. Kaufmann, T. Schaub, Multi-shot ASP solving with clingo, Theory Pract. Log. Program. 19 (2019) 27–82. URL: https://doi.org/10.1017/S1471068418000054. doi:10.1017/S1471068418000054.

[4] F. Chiariello, F. Maggi, F. Patrizi, ASP-based declarative process mining (extended abstract), in: Proceedings of the 38th Fabio Patrizi, Electronic Proceedings in Theoretical Computer Science (EPTCS), 2022.

[5] G. De Giacomo, M. Y. Vardi, Linear temporal logic and linear dynamic logic on finite traces, in: F. Rossi (Ed.), IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013, IJCAI/AAAI, 2013, pp. 854–860. URL: http://www.aaai.org/ocs/index.php/IJCAI/IJCAI13/paper/view/6997.

[6] W. M. P. van der Aalst, Process Mining - Data Science in Action, Second Edition, Springer, 2016.

[7] M. Weske, Business Process Management - Concepts, Languages, Architectures, Third Edition, Springer, 2019.

[8] I. H. Witten, E. Frank, M. A. Hall, Data mining: practical machine learning tools and techniques, 3rd Edition, Morgan Kaufmann, Elsevier, 2011.

[9] W. M. P. van der Aalst, The application of Petri nets to workflow management, J. Circuits Syst. Comput. 8 (1998) 21–66.

[10] W. M. van der Aalst, C. Stahl, Modeling business processes - a Petri net-oriented approach, in: CoopIS series, 2011.

[11] S. A. White, Introduction to BPMN, Ibm Cooperation 2 (2004) 0.

[12] T. Allweyer, BPMN 2.0 : introduction to the standard for business process modeling, 2016.

[13] C. Di Ciccio, M. Montali, Declarative process specifications: Reasoning, discovery, monitoring, in: W. M. P. van der Aalst, J. Carmona (Eds.), Process Mining Handbook, volume 448 of *Lecture Notes in Business Information Processing*, Springer, 2022, pp. 108–152. URL: https://doi.org/10.1007/978-3-031-08848-3_4. doi:10.1007/978-3-031-08848-3_4.

[14] 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. URL: https://doi.org/10.1007/s00450-009-0057-9. doi:10.1007/s00450-009-0057-9.

[15] B. Finkbeiner, H. Sipma, Checking finite traces using alternating automata, Formal Methods Syst. Des. 24 (2004) 101–127.

[16] V. Fionda, G. Greco, LTL on finite and process traces: Complexity results and a practical reasoner, J. Artif. Intell. Res. 63 (2018) 557–623.

[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. URL: https://ojs.aaai.org/index.php/AAAI/article/view/20493.

[18] G. De Giacomo, M. Favorito, Compositional approach to translate ltlf/ldlf into deterministic finite automata, in: ICAPS, AAAI Press, 2021, pp. 122–130.

[19] F. Chiariello, F. M. Maggi, F. Patrizi, From LTL on process traces to finite-state automata, in: BPM (Demos / Resources Forum), volume 3469 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2023, pp. 127–131.

[20] S. Hahn, Automata techniques for temporal answer set programming, in: ICLP Technical Communications, volume 345 of *EPTCS*, 2021, pp. 258–266.

[21] P. Cabalar, M. Diéguez, S. Hahn, T. Schaub, Automata for dynamic answer set solving: Preliminary report, in: ICLP Workshops, volume 2970 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2021.

[22] P. Cabalar, R. Kaminski, P. Morkisch, T. Schaub, telingo = ASP + time, in: LPNMR, volume 11481 of *Lecture Notes in Computer Science*, Springer, 2019, pp. 256–269.

[23] P. Cabalar, Temporal ASP: from logical foundations to practical use with telingo, in: Reasoning Web, volume 13100 of *Lecture Notes in Computer Science*, Springer, 2021, pp. 94–114.

[24] F. Chiariello, F. M. Maggi, F. Patrizi, A tool for compiling declarative process mining problems in ASP, Softw. Impacts 14 (2022) 100435.

[25] A. Alman, C. Di Ciccio, D. Haas, F. M. Maggi, J. Mendling, Rule mining in action: The rum toolkit, in: C. Di Ciccio, B. Depaire, J. D. Weerdt, C. D. Francescomarino, J. Munoz-Gama (Eds.), Proceedings of the ICPM Doctoral Consortium and Tool Demonstration Track 2020, volume 2703 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2020, pp. 51–54.

[26] A. Ielo, F. Ricca, L. Pontieri, Declarative mining of business processes via ASP, in: G. D. Giacomo, A. Guzzo, M. Montali, L. Limonad, F. Fournier, T. Chakraborti (Eds.), Proceedings of the Workshop on Process Management in the AI Era (PMAI 2022), volume 3310 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2022, pp. 105–108. URL: http://ceur-ws.org/Vol-3310/paper14.pdf.

[27] G. De Giacomo, F. M. Maggi, A. Marrella, F. Patrizi, On the disruptive effectiveness of automated planning for ltl$_f$-based trace alignment, in: S. Singh, S. Markovitch (Eds.), Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA, AAAI Press, 2017, pp. 3555–3561. URL: http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14652.