# Temporal Reasoning in ASP and its Application to Declarative Process Mining

**Francesco Chiariello**[1], **Fabrizio Maria Maggi**[2], **Fabio Patrizi**[1]

[1] DIAG - Sapienza University of Rome, Italy
[2] KRDB - Free University of Bozen-Bolzano, Italy
chiariello@diag.uniroma1.it, maggi@inf.unibz.it, patrizi@diag.uniroma1.it

**Abstract**

We propose a new approach for solving temporal problems with Answer Set Programming that exploit the automata representation of $\text{LTL}_f$ formulae and apply it to a selection of Declarative Process Mining problems.

## Introduction

*Answer Set Programming* (ASP) (Brewka, Eiter, and Truszczynski 2011) is a declarative problem solving approach that has become very popular in the last years also thanks to the development of efficient ASP systems such as DLV (Alviano et al. 2017) and *clingo* (Gebser et al. 2019). Given a problem, this is modeled as a logic program and is fed into an ASP system. The system then computes the answer sets of the program (which can be roughly thought of as models of the program), each corresponding to a different solution to the problem.

To take advantage of the capabilities of ASP systems to solve problems involving temporal specifications, the idea we propose (Chiariello, Maggi, and Patrizi 2022a) is to use the well-known relationship between finite-state automata and $\text{LTL}_f$ formulae (De Giacomo and Vardi 2013). In fact, one can represent $\text{LTL}_f$ formulae in an ASP program by simply 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. Note also that ASP has different advantages. First, it provides clear and concise syntax inspired by Logic Programming and Prolog, to model problems. Second, the minimality of its semantics makes it very efficient at solving problems over graph-like structures (compared e.g. to SAT), thus making ASP a natural choice for the problems of interest.

In (Chiariello, Maggi, and Patrizi 2022b) we have considered various temporal problems from Declarative Process Mining and have shown how to apply the approach described above. Process Mining (PM) (van der Aalst and Carmona 2022) is a research area that deals with analyzing event logs to extract information related to the process that generated the log. Logs are collections of sequences of activities. Processes are modeled using different formalisms. In the case of Declarative Process Mining (DPM) (Di Ciccio and Montali 2022), process models are specified using logical formalisms such as DECLARE (van der Aalst, Pesic, and Schonenberg 2009) or $\text{LTL}_f$, which constitutes a generalization of the former.

*Log Generation* is the problem of generating a set of sequences satisfying a given input model. It is worth noting that in PM at each sequence's position there is only one activity. Therefore, such sequences differ from the traces usually seen in $\text{LTL}_f$. However, we can think of them as particular traces where the propositional interpretations are singleton. For that reason, we shall call them just 'traces', or 'process traces' when we want to emphasize their peculiarity. *Conformance Checking* is the problem of checking whether the traces of a model satisfy a given input model. Finally, *Query checking* is the problem of checking constraint templates, i.e. formulae with variables, against a log in order to find the instantiations compliant with the log.

These problems can be easily solved with ASP once the automata corresponding to the input model/formulae are available. The automata can be obtained using available tools such as Lydia[1] or LTLf2DFA[2][3]. 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 $\text{LTL}_f$ formula $\mathbf{G}(a \rightarrow \mathbf{F}b)$. Note that simplifications can be made (although they are not necessary) in the ASP encoding by exploiting the fact that we are working with process traces, i.e. only one activity is performed at a time. The corresponding ASP encoding is instead shown in Listings 1. Here, $a$ and $b$ represent placeholders for the activation and target activity of the Response template, while the new symbol $c$ stands for any other activity. It is then sufficient to add rules for updating the current automata state while reading the traces and we have all the ingredients we need for solving the problems.

For log generation, we add generation rules for guessing

[1] https://github.com/whitemech/lydia
[2] https://github.com/whitemech/LTLf2DFA
[3] http://ltlf2dfa.diag.uniroma1.it/

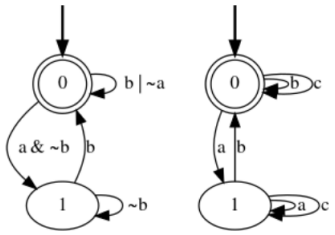


Figure 1: Automaton of Response template: (left) as obtained by LTLf2DFA (right) after simplification

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 rules are required: the traces are already given. We just need to test whether they are accepted. For query checking, we use generation rules to guess the instantiation and then check whether the input log satisfies the formula so obtained.

Listing 1: ASP encoding of Response template

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

The problems we considered are relatively simple and they are intended just to demonstrate the potential of the approach. Nevertheless, the results were so satisfying that made the authors of the DPM toolkit RuM (Alman et al. 2020) integrates it into their application for log generation. Following our approach, (Ielo, Ricca, and Pontieri 2022) proposes to use it for Process Discovery (i.e. finding a model of the log) while using the ASP optimization capabilities to take into account also user preferences. The optimization capabilities can also be used for solving 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 (De Giacomo et al. 2017). Finally, we stress that, while we have considered here DPM problems, there is no reason to limit ourselves to this particular domain since the approach can be virtually applied to any problem involving $\text{LTL}_f$ specifications.

## Acknowledgments

Work partly supported by the ERC Advanced Grant WhiteMech (No. 834228), the EU ICT-48 2020 project TAILOR (No. 952215), the EU ICT-49 2021 project AIPlan4EU (No. 101016442), and the PRIN project RIPER (No. 20203FFYLK).

## References

Alman, A.; Di Ciccio, C.; Haas, D.; Maggi, F. M.; and Mendling, J. 2020. Rule Mining in Action: The RuM Toolkit. In Di Ciccio, C.; Depaire, B.; Weerdt, J. D.; Francescomarino, C. D.; and Munoz-Gama, J., eds., *Proceedings of the ICPM Doctoral Consortium and Tool Demonstration Track 2020*, volume 2703 of *CEUR Workshop Proceedings*, 51–54. CEUR-WS.org.

Alviano, M.; Calimeri, F.; Dodaro, C.; Fuscà, D.; Leone, N.; Perri, S.; Ricca, F.; Veltri, P.; and Zangari, J. 2017. The ASP System DLV2. In Balduccini, M.; and Janhunen, T., 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*, 215–221. Springer.

Brewka, G.; Eiter, T.; and Truszczynski, M. 2011. Answer set programming at a glance. *Commun. ACM*, 54(12): 92–103.

Chiariello, F.; Maggi, F.; and Patrizi, F. 2022a. ASP-Based Declarative Process Mining (Extended Abstract). In *Proceedings of the 38th International Conference on Logic Programming (Technical Communications) (ICLP)*. Electronic Proceedings in Theoretical Computer Science (EPTCS).

Chiariello, F.; Maggi, F. M.; and Patrizi, F. 2022b. ASP-Based Declarative Process Mining. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022*, 5539–5547. AAAI Press.

De Giacomo, G.; Maggi, F. M.; Marrella, A.; and Patrizi, F. 2017. On the Disruptive Effectiveness of Automated Planning for LTL*f*-Based Trace Alignment. In Singh, S.; and Markovitch, S., eds., *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, 3555–3561. AAAI Press.

De Giacomo, G.; and Vardi, M. Y. 2013. Linear Temporal Logic and Linear Dynamic Logic on Finite Traces. In Rossi, F., ed., *IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013*, 854–860. IJCAI/AAAI.

Di Ciccio, C.; and Montali, M. 2022. Declarative Process Specifications: Reasoning, Discovery, Monitoring. In van der Aalst, W. M. P.; and Carmona, J., eds., *Process Mining Handbook*, volume 448 of *Lecture Notes in Business Information Processing*, 108–152. Springer.

Gebser, M.; Kaminski, R.; Kaufmann, B.; and Schaub, T. 2019. Multi-shot ASP solving with clingo. *Theory Pract. Log. Program.*, 19(1): 27–82.

Ielo, A.; Ricca, F.; and Pontieri, L. 2022. Declarative Mining of Business Processes via ASP. In Giacomo, G. D.; Guzzo, A.; Montali, M.; Limonad, L.; Fournier, F.; and Chakraborti, T., eds., *Proceedings of the Workshop on Process Management in the AI Era (PMAI 2022)*, volume 3310 of *CEUR Workshop Proceedings*, 105–108. CEUR-WS.org.

van der Aalst, W. M. P.; and Carmona, J., eds. 2022. *Process Mining Handbook*, volume 448 of *Lecture Notes in Business Information Processing*. Springer. ISBN 978-3-031-08847-6.

van der Aalst, W. M. P.; Pesic, M.; and Schonenberg, H. 2009. Declarative workflows: Balancing between flexibility and support. *Comput. Sci. Res. Dev.*, 23(2): 99–113.