



## Highlights

- A new approach is proposed for Temporal Reasoning in ASP;
- The approach takes advantage of the automata representation of  $LTL_f$  formulae;
- It is shown how to apply it for solving three DPM problems: Log Generation, Conformance Checking, and Query Checking;
- Poster based on work that appeared in [1, 2, 3]

## Declarative Process Mining

Declarative Process Mining [4] is a subfield of Process Mining where processes are modeled using constraint-based languages, such as DECLARE [5] or  $LTL_f$  [6].

## $LTL_f$

- Linear-Time Temporal logic on finite traces ( $LTL_f$ ) is a logic that allows expressing properties of finite sequences, called traces.
- Given a set  $P$  of propositional symbols, the syntax is defined by the following grammar:

$$\varphi ::= A \mid \neg\varphi \mid \varphi_1 \quad \varphi_2 \mid \mathbf{X}\varphi \mid \varphi_1 \mathbf{U}\varphi_2$$

with  $A \in P$ .

- Given a formula  $\varphi$ , a trace  $\pi = \pi_1, \pi_2, \dots, \pi_{len(\pi)} \in (2^P)^+$ , and a time instant  $i$ , with  $1 \leq i \leq len(\pi)$ , the semantics is defined as follows:
  - $\pi, i \models A$  iff  $A \in \pi_i$ ,
  - $\pi, i \models \neg\varphi$  iff  $\pi, i \not\models \varphi$ ,
  - $\pi, i \models \varphi_1 \quad \varphi_2$  iff  $\pi, i \models \varphi_1$  and  $\pi, i \models \varphi_2$ ,
  - $\pi, i \models \mathbf{X}\varphi$  if  $i < len(\pi)$  and  $\pi, i+1 \models \varphi$ ,
  - $\pi, i \models \varphi_1 \mathbf{U}\varphi_2$  iff  $\pi, j \models \varphi_2$  for some  $j$ , with  $i \leq j \leq len(\pi)$ , and  $\pi, k \models \varphi_1$  for all  $k = i, \dots, j-1$ .
- Common abbreviations used are:
  - $true$ ,
  - $\mathbf{F}\varphi \quad true \mathbf{U}\varphi$
  - $\mathbf{G}\varphi \quad \neg\mathbf{F}\neg\varphi$
  - $\varphi_1 \mathbf{W}\varphi_2 \quad \varphi_1 \mathbf{U}\varphi_2 \quad \mathbf{G}\varphi_1$

## DECLARE as $LTL_f$

| Template                             | Formula  |
|--------------------------------------|--|
| <i>Absence</i> ( $a$ )               | $\neg\mathbf{F}a$                                    |
| <i>Existence</i> ( $a$ )             | $\mathbf{F}a$  |
| <i>Response</i> ( $a, b$ )           | $\mathbf{G}(a \quad \mathbf{F}b)$                    |
| <i>NotResponse</i> ( $a, b$ )        | $\mathbf{G}(a \quad \neg\mathbf{F}b)$                |
| <i>RespondedExistence</i> ( $a, b$ ) | $\mathbf{F}a \quad \mathbf{F}b$                      |
| <i>AlternateResponse</i> ( $a, b$ )  | $\mathbf{G}(a \quad \mathbf{X}(\neg a \mathbf{U}b))$ |
| <i>Precedence</i> ( $a, b$ )         | $\neg b \mathbf{W}a$                                 |

## $LTL_f$ 2DFA

For each  $LTL_f$  formula, there exists a finite-state automaton that accepts exactly the traces satisfying the formula.

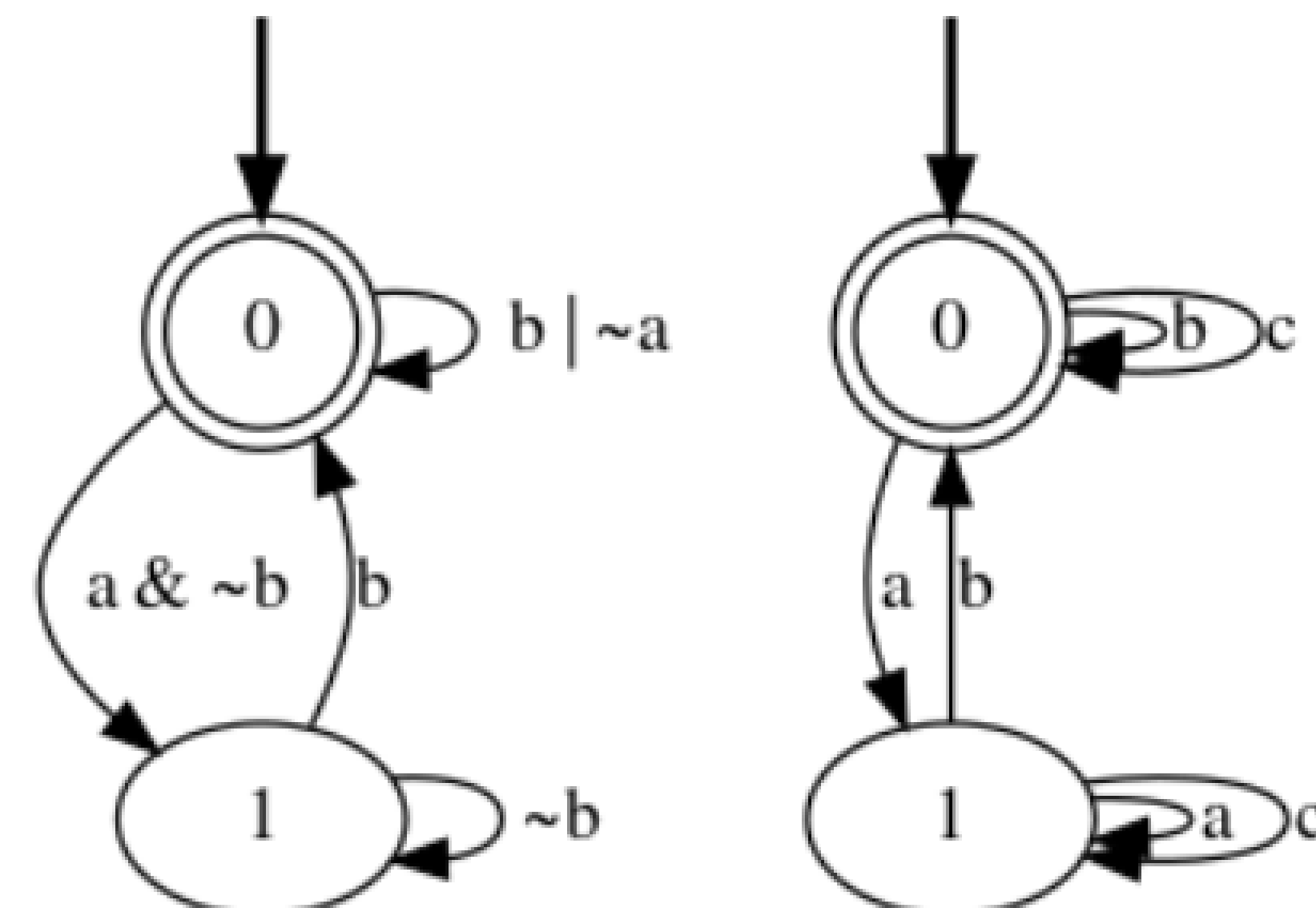


Figure: Automaton of *Response*( $a, b$ ) template: (left) as obtained by available  $LTL_f$  tools for conversion (right) simplified by exploiting that we work with process traces

## ASP

- Answer Set Programming [7] is a declarative problem solving approach inspired by Logic Programming and SAT.
- Given a problem, this is modeled as a logic program and is fed into an ASP system, such as *clingo* [8]. The system then computes the stable models of the program, each corresponding to a different solution to the problem.

## Encoding Temporal Problems in ASP

Given a problem involving temporal specifications one can represent the corresponding automata in ASP and simulate their running over a trace. The problems then reduce to checking whether the automata accept the trace.

```

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).

```

ASP encoding of *Response*( $a, b$ ).

## Problems

- Log generation: use generation rules for guessing a trace and a test rule for checking whether the trace is accepted.
- Conformance Checking: just check whether the traces are accepted.
- Query Checking: guess a template instantiation and check if the automata obtained accepts the log.

## Conclusions and Future Work

- We have seen how to solve DPM problems using ASP;
- The solution is based on exploiting the automata representation of the process models;
- The approach is applicable to many other DPM problems, e.g., Process Discovery and Trace Alignment;
- One could also consider more in general PM problems by using ASP for modeling Petri nets.

## References

- [1] Francesco Chiariello, Fabrizio Maria Maggi, and Fabio Patrizi. ASP-Based Declarative Process Mining. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(5):5539–5547, June 2022. Number: 5.
- [2] Francesco Chiariello, Fabrizio Maria Maggi, and Fabio Patrizi. 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), 2022.
- [3] Francesco Chiariello, Fabrizio Maria Maggi, and Fabio Patrizi. A tool for compiling Declarative Process Mining problems in ASP. *Software Impacts*, page 100435, October 2022.
- [4] Claudio Di Ciccio and Marco Montali. Declarative process specifications: Reasoning, discovery, monitoring. In Wil M. P. van der Aalst and Josep Carmona, editors, *Process Mining Handbook*, volume 448 of *Lecture Notes in Business Information Processing*, pages 108–152. Springer, 2022.
- [5] Wil M. P. van der Aalst, Maja Pesic, and Helen Schonenberg. Declarative workflows: Balancing between flexibility and support. *Comput. Sci. Res. Dev.*, 23(2):99–113, 2009.
- [6] Giuseppe De Giacomo and Moshe Y. Vardi. Linear temporal logic and linear dynamic logic on finite traces. In Francesca Rossi, editor, *IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013*, pages 854–860. IJCAI/AAAI, 2013.
- [7] Gerhard Brewka, Thomas Eiter, and Miroslaw Truszczynski. Answer set programming at a glance. *Commun. ACM*, 54(12):92–103, 2011.
- [8] Martin Gebser, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub. Multi-shot ASP solving with clingo. *Theory Pract. Log. Program.*, 19(1):27–82, 2019.

## Acknowledgements

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 AI-Plan4EU (No. 101016442), and the PRIN project RIPER (No. 20203FFYLK).