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 [2]KRDB - Free University of Bozen-Bolzano

chiariello@diag.uniroma1.it

Highlights

- We propose a new approach [1] for Temporal Reasoning in ASP;
- The approach takes advantage of the automata representation of $\mathbf{LTL}_f$ formulae;
- It is shown how to apply it for solving three DPM problems: Log Generation, Conformance Checking, and Query Checking;

Declarative Process Mining

Declarative Process Mining [2] is a subfield of Process Mining where processes are modeled using constraint-based languages, such as **DECLARE** [3] or $\mathbf{LTL}_f$ [4].

$\mathbf{LTL}_f$

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

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

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

DECLARE as $\mathbf{LTL}_f$

Template	Formula
$Absence(a)$	$\neg\mathbf{F}a$
$Existence(a)$	$\mathbf{F}a$
$Response(a,b)$	$\mathbf{G}(a \rightarrow \mathbf{F}b)$
$NotResponse(a,b)$	$\mathbf{G}(a \rightarrow \neg\mathbf{F}b)$
$RespondedExistence(a,b)$	$\mathbf{F}a \rightarrow \mathbf{F}b$
$AlternateResponse(a,b)$	$\mathbf{G}(a \rightarrow \mathbf{X}(\neg a \mathbf{U} b))$
$Precedence(a,b)$	$\neg b \mathbf{W} a$

$\mathbf{LTL}_f$2DFA

For each $\mathbf{LTL}_f$ formula, there exists a finite-state automaton that accepts exactly the traces satisfying the formula.

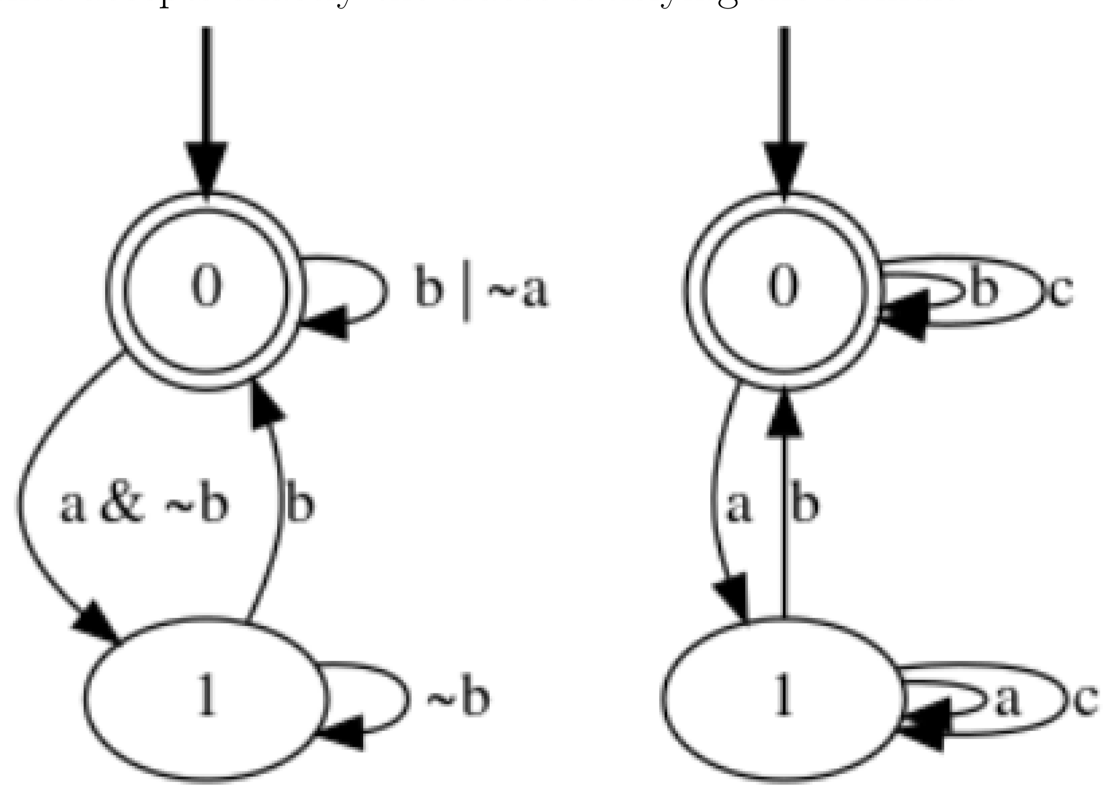

Figure: Automaton of $Response(a,b)$ template: (left) as obtained by available $\mathbf{LTL}_f$ tools for conversion (right) simplified by exploiting that we work with process traces

ASP

- Answer Set Programming [5] 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* [6]. 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 $Response(a,b)$.

Application to DPM 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;
- Many other DPM problems can be considered, e.g., Process Discovery and Trace Alignment;
- The approach is general enough to be applied Temporal Problems from different areas.

References

[1] Francesco Chiariello, Fabrizio Maria Maggi, and Fabio Patrizi. ASP-Based Declarative Process Mining. *Proc: of the AAAI Conference on Artificial Intelligence*, 36(5):5539–5547, 2022.

[2] 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.

[3] 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.

[4] Giuseppe De Giacomo and Moshe Y. Vardi. Linear temporal logic and linear dynamic logic on finite traces. In Francesca Rossi, editor, *IJCAI 2013, Proc. of the 23rd International Joint Conference on Artificial Intelligence*, pages 854–860. IJCAI/AAAI, 2013.

[5] Gerhard Brewka, Thomas Eiter, and Miroslaw Truszczynski. Answer set programming at a glance. *Commun. ACM*, 54(12):92–103, 2011.

[6] 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 AIPlan4EU (No. 101016442), and the PRIN project RIPER (No. 20203FFYLK).

TAILOR
Foundation of Trustworthy AI:
Integrating Learning, Optimisation and Reasoning