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

Francesco Chiariello

University of Naples Federico II
francesco.chiariello@unina.it

Contribution

- Show how to perform temporal reasoning in ASP using automata;
- Apply the method to Declarative Process Mining;
- Problems considered: Log Generation, Conformance Checking, and Query Checking.

Process Mining

- Process Mining (PM) is at the intersection of Business Process Management and Data Mining;
- PM analyzes event logs to extract information about the underneath process.
- Process models are typically Petri nets[1] or Business Process Modeling Notation[2].

[1] Wil M. P. van der Aalst. "The Application of Petri Nets to Workflow Management". In: *J. Circuits Syst. Comput.* 8.1 (1998), pp. 21–66

[2] Stephen A. White and Conrad Bock. *BPMN 2.0 Handbook Second Edition*. Future Strategies Inc., 2011

Declarative Process Mining

- Declarative PM specifies processes in a constraint-based fashion
- Formalisms used are DECLARE[3], LTL_f[4], and LTL_p[5];
- In DPM models specify the properties of the (traces of the) process
  - it specify *what* property a trace should have, rather then *how* to construct them
  - reduces false negative (i.e., traces erroneously excluded by the model)

[3] Wil M. P. van der Aalst, Maja Pesic, and Helen Schonenberg. "Declarative workflows: Balancing between flexibility and support". In: *Comput. Sci. Res. Dev.* 23.2 (2009), pp. 99–113

[4] Giuseppe De Giacomo and Moshe Y. Vardi. "Linear Temporal Logic and Linear Dynamic Logic on Finite Traces". In: *Proc. of the 23rd Int. Joint Conf. on Artificial Intelligence*. IJCAI/AAAI, 2013

[5] Valeria Fionda and Gianluigi Greco. "LTL on Finite and Process Traces: Complexity Results and a Practical Reasoner". In: *J. Artif. Intell. Res.* 63 (2018), pp. 557–623

DPM Problems

- Log generation: generate a log compliant with a process model.
- Conformance checking: check whether the traces are compliant with a process model.
- Query checking: finding properties of a process by checking possible templates against the event log of the process.

LTL$_f$: Syntax

- Given a set $\mathcal{P}$ of propositional symbols, the syntax is defined by the following grammar:

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

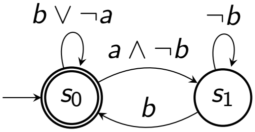
  with $A \in \mathcal{P}$.
- Common abbreviations used are:
  - *true*, $\rightarrow$, $\vee$
  - $\mathbf{F}\varphi \equiv \mathit{true}\mathbf{U}\varphi$
  - $\mathbf{G}\varphi \equiv \neg\mathbf{F}\neg\varphi$
  - $\varphi_1 \mathbf{W} \varphi_2 \equiv \varphi_1 \mathbf{U} \varphi_2 \vee \mathbf{G}\varphi_1$

LTL_f: Semantics

- 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 $1 \leq i \leq len(\pi)$, the semantics is defined as follows:
  - $\pi, i \models A$ iff $A \in \pi_i$,
  - $\pi, i \models \neg\varphi$ iff $\pi, i \not\models \varphi$,
  - $\pi, i \models \varphi_1 \wedge \varphi_2$ 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 $i \leq j \leq len(\pi)$, and $\pi, k \models \varphi_1$ for all $k = i, \ldots, j-1$.
- A formula φ is true in π, and we write $\pi \models \varphi$, if $\pi, 1 \models \varphi$.

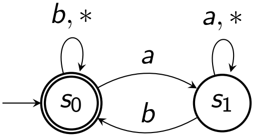
LTL$_f$ and Automata

- For each LTL$_f$ formula φ there exists a NFA A_φ that accepts exactly the traces that satisfy φ.
- For example to $\varphi = \mathbf{G}(a \rightarrow \mathbf{F} b)$ is associated



LTL on Process Traces

- LTL_p restrict the semantics to consider only process traces (or simple finite traces)
- This, in turn, result in simpler automata where arcs are labeled directly by activities[6]



[6] Francesco Chiariello, Fabrizio Maria Maggi, and Fabio Patrizi. "From LTL on Process Traces to Finite-state Automata". In: *BPM (Demos / Resources Forum)*. Vol. 3469. CEUR Workshop Proceedings. CEUR-WS.org, 2023, pp. 127–131

- Answer Set Programming (ASP): declarative approach for search and optimization problems [7][8].
- Provide a modeling language for writing logic programs.
- Programs' models are computed with ASP systems such as
  - *clingo*[9]
  - DLV [10]

[7] Ilkka Niemelä. "Logic Programs with Stable Model Semantics as a Constraint Programming Paradigm". In: *Ann. Math. Artif. Intell.* 25.3-4 (1999), pp. 241–273

[8] Victor W. Marek and Miroslaw Truszczynski. "Stable Models and an Alternative Logic Programming Paradigm". In: *The Logic Programming Paradigm*. Artificial Intelligence. Springer, 1999, pp. 375–398

[9] Martin Gebser et al. "Multi-shot ASP solving with clingo". In: *Theory Pract. Log. Program.* 19.1 (2019), pp. 27–82

[10] Mario Alviano et al. "The ASP System DLV2". In: *LPNMR*. vol. 10377. Lecture Notes in Computer Science. Springer, 2017, pp. 215–221

ASP: Syntax

- A **normal rule** is of the form

$$h \leftarrow b_1, \ldots, b_m, \text{not } b_{m+1}, \ldots, \text{not } b_n$$

  where $h, b_1, \ldots, b_n$ are atoms.
- An **integrity constraint** is of the form

$$\leftarrow b_1, \ldots, b_m, \text{not } b_{m+1}, \ldots, \text{not } b_n$$

- A **choice rule** with cardinality constraints is of the form

$$l\{h_1, \ldots h_n\}u$$

  with $l, u \in \mathbb{N}, l \leq u \leq n$.

ASP: Semantics

Given a logic program Π and a set X of atoms we define the reduct Π^X of Π w.r.t. X as the program obtained from Π as follows:

- if a rule contains in the negative body an atom that is in X we remove the rule,
- of the remaining rules, we remove the negative body.

In this way the resulting program Π^X doesn't contain default negation. X is then an *answer set*, or *stable model*, of Π if it coincides with the (unique) minimal model of Π^X.

Generate-and-test Methodology

- Generate and test (also called Guess and Check) methodology:
  1. Generate: guess a candidate solution
  2. Test: check if the candidate is a proper solution
- Differences from brute force:
  - candidate's selection
  - evaluation of partial candidates

Automata-based Temporal Reasoning in ASP

The proposed approach[11] consistis of the following steps:

- Convert temporal specifications to automata.
- Represent automata in ASP.
- Represent traces in ASP.
- Modeling how automata read trace.
- Add generation and test rules.

[11] Francesco Chiariello, Fabrizio Maria Maggi, and Fabio Patrizi. "ASP-Based Declarative Process Mining". In: *AAAI*. AAAI Press, 2022, pp. 5539–5547

ASP for DPM: traces

Predicates:

- $trace(A, T)$: activity A happens at time T.

Example

Trace $\pi = a_2, a_1, a_2$ becomes:

- $trace(a_2, 1)$.
- $trace(a_1, 2)$.
- $trace(a_2, 3)$.

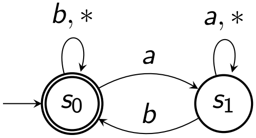
ASP for DPM: automata

- $init(S)$: S is the initial state.
- $acc(S)$: S is an accepting state.
- $trans(S, F, S')$: there exists a transition from state S to state S' labeled with event formula F.

- $holds(F, T)$: event formula F holds at time T.

Example

The ASP encoding of the LTL_p formula $\varphi = \mathbf{G}(a \rightarrow \mathbf{F} b)$ is given by:



- $init(s_0)$.
- $acc(s_0)$.
- $trans(s_0, 1, s_1)$.
- $holds(1, T) \leftarrow trace(a, T)$.
- $trans(s_1, 2, s_0)$.
- $holds(2, T) \leftarrow trace(b, T)$.
- $trans(s_0, 3, s_0)$.
- $holds(3, T) \leftarrow trace(b, T)$.
- $holds(3, T) \leftarrow trace(C, T), C \neq a, C \neq b$.
- $trans(s_1, 4, s_1)$.
- $holds(4, T) \leftarrow trace(A, T), A \neq b$.

ASP for DPM: reading trace

Predicate *state* models execution of automaton on trace

- $state(S, T)$: S is current state at time T.

and updated as

- $state(S, 0) \leftarrow init(S)$.
- $state(S', T) \leftarrow state(S, T - 1), trans(S, F, S'), holds(F, T)$.

Log Generation

Given an formula and trace length t,

Generate traces as follows

- $\{trace(A, T) : activity(A)\} = 1 \leftarrow time(T).$

Test traces as follows

- $sat \leftarrow state(S, t), accepting(S).$
- $\leftarrow \texttt{not}\ sat.$

Conformance Checking

- Traces are given as input
- Just check whether they are accepted

Query Checking Example

Query checking: finding properties of a process by checking possible templates against its event log.

- Input
  - Log: (a,b,c,c,b); (c,b,c,c,c)
  - Formula: $\mathbf{G}(?a \rightarrow \mathbf{F}?b)$
  - (optional) Constraints number: 1
- Output: $\mathbf{G}(a \rightarrow \mathbf{F}b)$

Query Checking Modeling

The following predicates are introduced

- $var(V)$: V is a variable.
- $assgnmt(V, A)$: activity A is assigned to variable V.

The body of the rule for *holds* is modified by replacing $trace(act, T)$ with $trace(A, T), assgnmt(v, A)$, with v being the variable in place of activity act.

Then for generating

- $\{assgnmt(V, A) : activity(A)\} = 1 \leftarrow var(V)$.

and for testing we check that the formula is satisfied by the trace.

Conclusion

- We have seen how the automa representation of temporal specifications can be used in ASP to perform temporal reasoning.
- We have considered Declarative Process Mining as an application domain to illustrate the approach.
- The contributions are manifold and benifit different communities:
  - To the Temporal Logics community, it provides a tool to perform temporal reasoning;
  - To the ASP community, it provides a method to intuitively handle time;
  - to the Process Mining community, it provides both tools and methods for analzing event logs.

Future Work

- Application to other DPM problems, e.g.,
  - Process Discovery,
  - Process Model Repair,
  - Trace Alignment.
- Application to other areas, e.g.
  - Discrete Event Systems,
  - Planning,
  - **Put your field here.**

Thank you!!

Bibliography I

[Aal98] Wil M. P. van der Aalst. "The Application of Petri Nets to Workflow Management". In: *J. Circuits Syst. Comput.* 8.1 (1998), pp. 21–66.

[Alv+17] Mario Alviano et al. "The ASP System DLV2". In: *LPNMR*. Vol. 10377. Lecture Notes in Computer Science. Springer, 2017, pp. 215–221.

[APS09] Wil M. P. van der Aalst, Maja Pesic, and Helen Schonenberg. "Declarative workflows: Balancing between flexibility and support". In: *Comput. Sci. Res. Dev.* 23.2 (2009), pp. 99–113.

[CMP22] Francesco Chiariello, Fabrizio Maria Maggi, and Fabio Patrizi. "ASP-Based Declarative Process Mining". In: *AAAI*. AAAI Press, 2022, pp. 5539–5547.

Bibliography II

[CMP23] Francesco Chiariello, Fabrizio Maria Maggi, and Fabio Patrizi. "From LTL on Process Traces to Finite-state Automata". In: *BPM (Demos / Resources Forum)*. Vol. 3469. CEUR Workshop Proceedings. CEUR-WS.org, 2023, pp. 127–131.

[DV13] Giuseppe De Giacomo and Moshe Y. Vardi. "Linear Temporal Logic and Linear Dynamic Logic on Finite Traces". In: *Proc. of the 23rd Int. Joint Conf. on Artificial Intelligence*. IJCAI/AAAI, 2013.

[FG18] Valeria Fionda and Gianluigi Greco. "LTL on Finite and Process Traces: Complexity Results and a Practical Reasoner". In: *J. Artif. Intell. Res.* 63 (2018), pp. 557–623.

[Geb+19] Martin Gebser et al. "Multi-shot ASP solving with clingo". In: *Theory Pract. Log. Program.* 19.1 (2019), pp. 27–82.

Bibliography III

[MT99] Victor W. Marek and Miroslaw Truszczynski. "Stable Models and an Alternative Logic Programming Paradigm". In: *The Logic Programming Paradigm*. Artificial Intelligence. Springer, 1999, pp. 375–398.

[Nie99] Ilkka Niemelä. "Logic Programs with Stable Model Semantics as a Constraint Programming Paradigm". In: *Ann. Math. Artif. Intell.* 25.3-4 (1999), pp. 241–273.

[WB11] Stephen A. White and Conrad Bock. *BPMN 2.0 Handbook Second Edition*. Future Strategies Inc., 2011.