ASP-based 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

Presented at AAAI-22

Contribution

- Show how ASP can solve Declarative Process Mining problems;
- Problems considered: Log Generation, Conformance Checking, and Query Checking;
- Control-flow and Data-aware perspective considered;
- Show how to handle temporal specifications (exploiting automata representation).

Declarative Process Mining

- Process Mining (PM) is at the intersection of Business Process Management and Data Mining.
- PM analyses event logs for extracting information about the underneath process.
- Declarative PM specifies process models in a constraint-based fashion using formalisms like DECLARE [APS09] and LTL$_f$ [DV13]

Process Mining Terminology

- Event: activity + timestamp + (possibly) attributes.
- Trace: finite sequence of events.
- Event log: collection of traces.
- Process model: specification of properties of traces.

DPM Problems

- Log generation [Sky+18]: generating a log compliant with a process model.
- Conformance checking [BMS16]: checking whether traces are compliant with a process model.
- Query checking [Räi+14]: finding properties of a process by checking possible templates against the event log of the process.

Process Perspectives

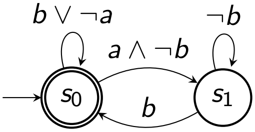
- Control-flow perspective focuses on activities.
- Data-aware perspective focuses on attributes.
- Time perspective focuses on timestamps.

Template	Meaning
Absence(a)	Activity a never happens
Existence(a)	Activity a happens at least 1 time
Response(a, b)	If a happens, b happens afterwards
NotResponse(a, b)	If a happens, b doesn't happen afterwards
RespondedExistence(a, b)	If a happens, b happens
AlternateResponse(a, b)	If a happens then b happens without any a inbetween
Precedence(a, b)	If b happens, then a happened before it

DECLARE and LTL_f

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

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



Approach

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

ASP for DPM: data

Predicates:

- $act(A)$: A is an activity.
- $has\_attr(A, N)$: activity A has attribute N.
- $val(N, V)$: a possible value of attribute N is V.

Example

Activities $a_1(int, cat)$ and $a_2()$, with $int \in \{1, \ldots, 10\}$ and $cat \in \{c_1, c_2, c_3\}$ becomes:

- $act(a_1)$. $has\_attr(a_1, int)$. $has\_attr(a_1, cat)$.
- $act(a_2)$.
- $value(int, 1..10)$.
- $value(cat, c1)$. $value(cat, c2)$. $value(cat, c3)$.

ASP for DPM: traces

Predicates:

- $trace(A, T)$: activity A happens at time T.
- $has\_value(N, V, T)$: attribute N has value V at time T.

Example

Trace $a_2(), a_1(2, c_3), a_2()$ becomes:

- $trace(a_2, 0)$.
- $trace(a_1, 1)$. $has\_value(int, 2, 1)$. $has\_value(cat, c_3, 1)$.
- $trace(a_2, 2)$.

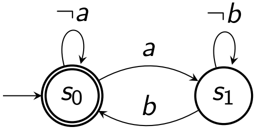
ASP for DPM: automata

- *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 formula $\varphi = \mathbf{G}(a \to \mathbf{F}b)$ is given by:



- $init(s_0)$.
- $acc(s_0)$.
- $trans(s_0, 1, s_1)$.
- $holds(1, T) \leftarrow trace(a, T)$.
- $trans(s_1, 2, s_0)$.
- $holds(2, T) \leftarrow trace(b, T)$.
- $trans(s_0, 3, s_0)$.
- $holds(3, T) \leftarrow \texttt{not}\ holds(1, T), time(T)$.
- $trans(s_1, 4, s_1)$.
- $holds(4, T) \leftarrow trace(A, T), A \neq b$.

Example (cont'd)

For the data-aware formula $\varphi' = \mathbf{G}((a \wedge n < 5) \rightarrow \mathbf{F} b)$ it is sufficient to modify the rule for $holds(1, T)$ as follows:

- $holds(1, T) \leftarrow trace(a, T), has\_value(n, V, T), V < 5.$

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

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

Log Generation

Given an LTL_f formula and trace length t,

Generate traces as follows

- $\{trace(A, T) : activity(A)\} = 1 \leftarrow time(T).$
- $\{has\_value(N, V, T) : value(N, V)\} = 1 \leftarrow trace(A, T), has\_attribute(A, N).$

Test traces as follows

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

Conformance Checking

It is given a set of traces.

- Add the trace index i to predicate sat.
- Check whether $sat(i)$ holds.

Query Checking

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.

Example

Consider formula $\varphi = \mathbf{G}((?A \wedge \mathit{number} < 5) \rightarrow \mathbf{F}b)$.

Rule for $\mathit{holds}(1, T)$ is:

$$\mathit{holds}(1, T) \leftarrow \mathit{trace}(A, T), \mathit{assgnmt}(\mathit{varA}, A), \mathit{has\_value}(n, V, T), V < 5.$$

Results

Our approach

- outperforms the SoA tool MP-Declare Log Generator [Sky+18]
- shows results comparable wrt SoA tool Declare Analyzer [BMS16]
- show the feasibility of data-aware Query checking

Conclusion

Provided

- ASP encoding of data-aware Log Generation, Conformance Checking, and Query Checking
- Performance evaluation wrt state-of-the-art

Future work

- add time-perspective (i.e. timestamp)
- correlation condition
- other problems and domains

Bibliography I

[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.

[BMS16] Andrea Burattin, Fabrizio Maria Maggi, and Alessandro Sperduti. "Conformance checking based on multi-perspective declarative process models". In: *Expert Syst. Appl.* 65 (2016).

[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.

Bibliography II

[Räi+14] Margus Räim et al. "Log-Based Understanding of Business Processes through Temporal Logic Query Checking". In: *On the Move to Meaningful Internet Systems: OTM 2014 Conferences*. 2014, pp. 75–92.

[Sky+18] Vasyl Skydanienko et al. "A Tool for Generating Event Logs from Multi-Perspective Declare Models". In: *Proceedings of the Dissertation Award, Demonstration, and Industrial Track at BPM 2018*. 2018.