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

Overview

Objective

- Temporal Reasoning with Answer Set Programming (ASP).
- Application to Declarative Process Mining.

Solution Approach

- Represent the automaton associated to the specifications,
- Simulate the run of trace over the automaton.

Motivation

- Minimality of ASP semantics allows one to easily represent and reason with automata.

ASP

- Answer Set Programming (ASP) is a Declarative Problem Solving paradigm
  1. Problem modeled as logic program,
  2. *Answer sets* computed using ASP system,
  3. Solutions exctrated from answer sets.

Declarative Process Mining

- Process Mining = Business Process Management + Data Mining.
- Extract information from event log.
- In Declarative Process Mining (DPM) processes are modeled with DECLARE or LTL_f.

DPM Problems

- Problems considered are
  - Log Generation: generate a set of traces compliant with the process model,
  - Conformance Checking: check whether a trace is conformant with a process model,
  - Query Checking: check constraint templates, i.e. formulae with variables, against a log to find the instantiations compliant with it.

Approach

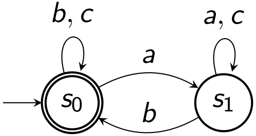
Our approach consists of:

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

Example

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

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



Simulating Trace Execution

Predicate *state* models execution of automaton on trace:

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

Update:

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

Log Generation

It is given an formula and trace length t.

Generate traces:

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

Test traces:

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

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.

Experiments and Results

- Experiments with both syntetic and real-life log show the feasibility of the approach for Conformance Checking and Query Checking (w.r.t. SoA tool)
- Better result than Alloy Log Generator (based on Alloy Analyzer) $\rightarrow$ our method integrated in DPM toolkit RuM

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!!