LTL and Past LTL on Finite Traces for Planning and Declarative Process Mining

Francesco Fuggitti

Master of Science in
Engineering in Computer Science
Sapienza, University of Rome

Advisor: Prof. Giuseppe De Giacomo

A.Y. 2017/2018

Outline

Introduction

- *Linear Temporal Logic* (LTL) is a simple formal language for expressing temporal specifications both in Artificial Intelligence (AI) and in Business Process Management (BPM)
  - LTL on *finite* traces (LTL$_f$) (De Giacomo and Vardi, 2013)
  - Past LTL on *finite* traces (PLTL) (Lichtenstein et al. 1985)

- In AI: planning for temporally extended goals from Bacchus and Kabanza, (1998) to Camacho et al. (2017/2018)

- In BPM: temporal specification of the constraint formula in Declarative Process Mining from Pesic and van der Aalst (2006) to Cecconi et al. (2018)

Objectives

- Provide a new efficient technique to transform LTL$_f$/PLTL formulas into *Deterministic Finite-state Automaton* (DFAs)

- Provide an approach to (FOND) Planning for LTL$_f$/PLTL goals:
  - reducing the problem to standard (FOND) planning
  - working with (FOND) domains instead of automata

- Provide a generalization of an approach to declarative process mining:
  - generalization of the constraint formula representation

- Implementation of all above-mentioned topics

PLTL and LTL$_f$ (De Giacomo and Vardi, 2013)

- Linear Temporal Logic on finite traces: LTL$_f$
  - exactly the same syntax of LTL
  - interpreted over *finite* traces
    - next: $\bigcirc$*happy*
    - until: *reply* $\mathcal{U}$ *acknowledge*
    - eventually: $\Diamond$*rich*
    - always: $\Box$*safe*
- Past Linear Temporal Logic: PLTL
  - same syntax of LTL$_f$, but looks into the past
    - yesterday: $\ominus$*happy*
    - since: *reply* $\mathcal{S}$ *acknowledge*
    - once: $\Diamondminus$*rich*
    - hystorically: $\boxminus$*safe*
- Reasoning in LTL$_f$/PLTL:
  - transform formulas φ into DFAs $\mathcal{A}_\varphi$
  - for every trace π, an LTL$_f$/PLTL formula φ is such that:

$$\pi \models \varphi \iff \pi \in \mathcal{L}(\mathcal{A}_\varphi)$$

Translation of LTL_f and PLTL formulas to DFA

- Technique based on a translation procedure:

1. starting from an LTL_f/PLTL formula φ, we translate it to FOL on finite sequences (De Giacomo and Vardi 2013; Zhu et al. 2018) through $fol(\varphi, x)$ and $fol_p(\varphi, x)$ translation functions:
   - LTL_f formulas evaluated in $x = 0$, hence $fol(\varphi, 0)$, since they look at the future
   - PLTL formulas evaluated in $x = last$, hence $fol_p(\varphi, last)$ with $last = |\pi| - 1$, since they look at the past

2. apply the highly optimized tool MONA able to transform Monadic Second Order Logic (and hence FOL as well) on finite strings to minimum DFA automata

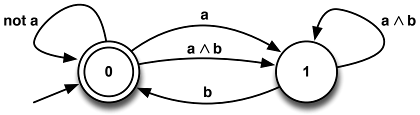
LTL$_f$2DFA: the implementation of the translation procedure

Python package supporting:

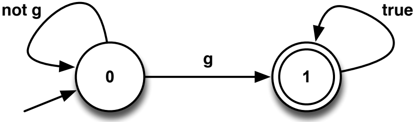
- parsing of LTL$_f$/PLTL formulas
- translation to FOL, DFA
- option for DECLARE assumption (De Giacomo et al. 2014)
- available online at http://ltlf2dfa.diag.uniroma1.it

Output examples:

$\varphi_1 = \Box(a \Rightarrow \bigcirc b)$



$\varphi_2 = \Diamond g$



FOND Planning for Extended Temporal Goals

- A *fully observable non-deterministic* (FOND) domain with initial state is a tuple $\mathcal{D} = \langle 2^{\mathcal{F}}, A, s_0, \varrho, \alpha \rangle$ where:
  - $\mathcal{F}$ is a set of *fluents* (atomic propositions);
  - A is a set of *actions* (atomic symbols);
  - $2^{\mathcal{F}}$ is the set of states;
  - s_0 is the initial state (initial assignment to fluents);
  - $\alpha(s) \subseteq A$ represents *action preconditions*;
  - $(s, a, s') \in \varrho$ with $a \in \alpha(s)$ represents *action effects* (including frame)
- Specified in PDDL as domain $\mathcal{D}$ and problem $\mathcal{P}$
- Actions effects are *non-deterministic*: who chooses what?
  - the Agent chooses the action to execute
  - the Environment chooses the successor state
- Goals, planning and plans
  - Goal: an LTL$_f$/PLTL formula φ
  - Planning: a *game* between the two players
  - Plan: *strategy* to *win* the game

The FOND4LTL$_f$/PLTL approach:

- Idea: reduce the problem to standard FOND planning
- How to deal with LTL$_f$/PLTL goal:
  1. transform the goal φ into the DFA $\mathcal{A}_\varphi = \langle \Sigma, Q, q_0, \delta, F \rangle$, through LTL$_f$2DFA
  2. to capture the general representation of $\mathcal{A}_\varphi$ in the $\mathcal{D}$, we modify $\mathcal{A}_\varphi$ to $\mathcal{A}'_\varphi$

The parametrization of $\mathcal{A}_\varphi$: $\mathcal{A}'_\varphi$

- from $\Sigma = \{a_0(\vec{o}), \ldots, a_n(\vec{o})\}$ to $\Sigma' = \{a'_0, \ldots, a'_n\} = \{a_0(\vec{x}), \ldots, a_n(\vec{x})\}$
- from $Q = \{q_0, \ldots, q_n\}$ to $Q' = \{q'_0(\vec{x}), \ldots, q'_n(\vec{x})\}$

where $\vec{o} = (o_0, \ldots, o_k)$ are objects of interest and $\vec{x} = (x_0, \ldots, x_k)$ are the corresponding variables

3. introduce the *turnDomain* predicate that enables to perform a step on the domain and another step on the DFA alternatively
4. encode δ' of $\mathcal{A}'_\varphi$ in PDDL as a new operator:

Action `trans`: encoding of δ' into domain $\mathcal{D}$

`parameters`: $(x_0, \dots, x_k)$, where $x_i \in \mathcal{V}$
`precondition`: $\neg turnDomain$
`effect`:
$when\ (q_i(x_0, \dots, x_k) \wedge a'_j)\ then\ (\delta'(q'_i, a'_j) = q''_i(x_0, \dots, x_k) \wedge (\neg q, \forall q \in Q \text{ s.t. } q \neq q''_i) \wedge turnDomain), \forall i, j : 0 \leq i \leq m, 0 \leq j \leq n$

5. in problem $\mathcal{P}$, produce a new initial state and new goal state accordingly

New initial state

$$s_0 \wedge turnDomain \wedge q_0(o_0, \dots, o_k)$$

New goal specification

$$turnDomain \wedge (\bigvee\nolimits_{q \in F} q(o_0, \dots, o_k))$$

Implementation and Results

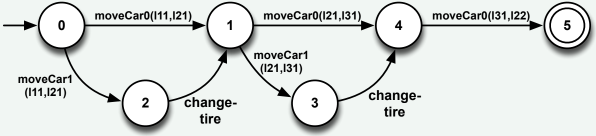
- FOND4LTL$_f$/PLTL Python package using FOND-SAT planner
- soon available online at http://fond4ltlfpltl.diag.uniroma1.it

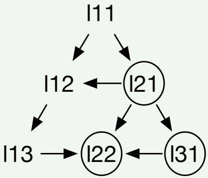
Example: the Triangle Tireworld domain

Objective: Drive from one location to another. A tire may be going flat. If there is a spare tire in the location of the car, then the car can use it to fix the flat tire.

Goal: $\varphi = vehicleAt(l22) \wedge \Diamond\!\!\!\!-(vehicleAt(l31))$

(Strong) Plan: any path from state 0 to state 5 achieves to the goal





The Janus approach for Declarative Process Mining

Declarative Process Mining is the set of techniques aimed at determining if a trace t is *interesting* wrt a given temporal specification φ, called a constraint.

- **Problem**: *"ex falso quod libet"*, i.e. a constraint can be satisfied even though it is never activated
- **Solution**: the Janus approach (Cecconi et al. 2018) proposed the *reactive constraint* (RCon) as $\Psi \doteq \alpha \mapsto \varphi$ where:
  - α is the activation condition
  - φ is an LTLp_f formula (i.e. LTL_f augmented with PLTL)

  for computing the *interestingness degree* function $\zeta(\Psi, t)$

Example: $\Psi = a \mapsto (\ominus b \vee \Diamond c)$

Given a trace $t = \langle d, f, a, f, c, a, f, b, a, f \rangle$, the $\zeta(\Psi, t) = \frac{2}{3} = 0.667$

Two main drawbacks:

α only a single task and implementation limited to DECLARE constraints

Our generalization of the Janus approach extends the original approach:

New constraint representation - RCon

$$\Psi \doteq \bigvee_{j=1}^{m} (\varphi^{\blacktriangleleft} \wedge \varphi^{\blacktriangledown} \wedge \varphi^{\blacktriangleright})_j$$

where $\varphi^{\blacktriangleleft}$ is a pure past formula, $\varphi^{\blacktriangledown}$ is a propositional formula on the current instant that triggers potential interest and $\varphi^{\blacktriangleright}$ is a pure future formula

New *interestingness degree* function $\eta(\Psi, t)$

$$\eta(\Psi, t) = \begin{cases} \dfrac{|\{t \models \bigvee_{j=1}^{m} (\varphi^{\blacktriangleleft} \wedge \varphi^{\blacktriangledown} \wedge \varphi^{\blacktriangleright})_j\}|}{|\{t \models \bigvee_{j=1}^{m} \varphi^{\blacktriangledown}\}|}, & \text{if } |\{t \models \bigvee_{j=1}^{m} \varphi^{\blacktriangledown}\}| \neq 0; \\ 0, & \text{otherwise} \end{cases}$$

Theorem

The $\eta(\Psi, t)$ function is a generalization of the $\zeta(\Psi, t)$ function

Implementation and Results

JANUS Python package:

- any type of constraint formula
- automata generation through LTL$_f$2DFA

Results:

Example: the *Sepsis*[1] event log

- Given an RCon:

$$\Psi = (\ominus \mathit{ER\ Registration} \wedge \color{red}{(\mathit{Leucocytes} \wedge \mathit{LacticAcid})} \wedge \mathit{True}) \vee$$
$$(\mathit{True} \wedge \color{red}{(\mathit{Leucocytes} \wedge \mathit{LacticAcid})} \wedge \Diamond \mathit{CRP})$$

- $t = \{$'ER Registration', ('ER Triage','ER Sepsis Triage), ('LacticAcid', 'IV Liquid'), ('Leucocytes', 'LacticAcid'), 'CRP', 'LacticAcid', ('Leucocytes', 'LacticAcid'), ('Leucocytes', 'IV Antibiotics'),'IV Liquid', 'Release A'$\}$
- $\eta(\Psi, t) = \frac{1}{2} = 0.5$

[1] *Sepsis* reports trajectories of patients showing symptoms of sepsis in a Dutch hospital

Conclusions

Thesis results:

- Provided the LTL$_f$2DFA tool which implements the translation procedure from LTL$_f$/PLTL to DFA
- Proposed and implemented the FOND4LTL$_f$/PLTL approach in compiling LTL$_f$/PLTL goals along with the original planning domain, specified in PDDL
- Extended the Janus approach both theoretically and practically

Future works:

- Investigate the LTLp$_f$ logic (i.e. LTL$_f$ and PLTL merged) for dealing directly with mixed formulas
- Extend our research to *Partially Observable Non Deterministic* (POND) domains
- Provide a tool that automatically separates LTLp$_f$ formulas
- Optimize and enrich all tools developed.

Appendix A

- Finite trace π: denotes a finite sequence of consecutive instants of time
- An LTL_f formula φ is *true* in π, in notation $\pi \models \varphi$, if $\pi, 0 \models \varphi$
- A PLTL formula φ is *true* in π, in notation $\pi \models \varphi$, if $\pi, last \models \varphi$

Example: translation procedure for $\varphi = \Diamond G$

FOL translation: $fol(\varphi, 0) = \exists y.0 \leq y \leq last \wedge G(y)$

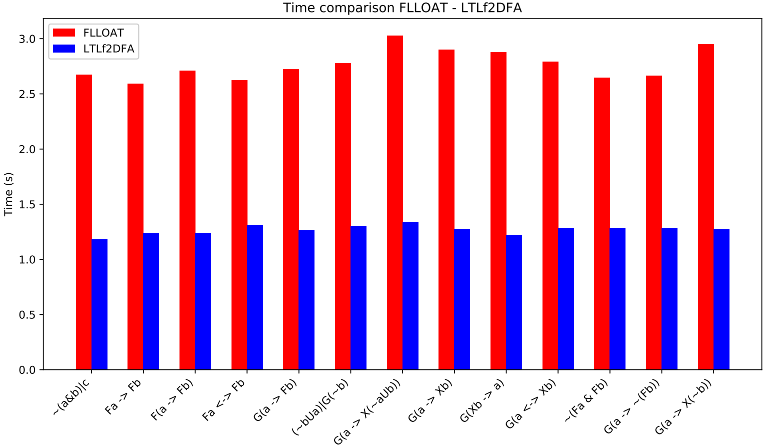
MONA program: `m2l-str; var2 G; ex1 y: 0<=y & y<=max($) & y in G`

Results

- an LTL_f formula can be reduced to DFA in double-exponential time
- a PLTL formula can be reduced to DFA in single exponential time

Appendix A1

LTL$_f$2DFA vs FLLOAT:



Appendix B

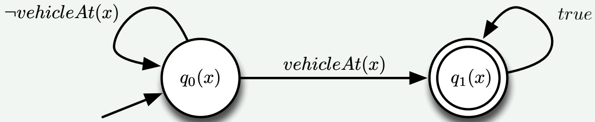
Definition

We define a mapping function f as follows:

$$f : \mathcal{O} \rightarrow \mathcal{V} \tag{1}$$

where $\mathcal{O}$ is the set of objects $\{o_0, \ldots, o_k\}$ and $\mathcal{V}$ is a set of variables $\{x_0, \ldots, x_k\}$

Example of parametric DFA: $\varphi = \Diamond vehicleAt(l22)$



Appendix B1

Strong Plan - Pure adversarial game

A strong plan is a strategy that is guaranteed to achieve the goal regardless of non-determinism.

Strong Cyclic Plan - Fairness

Strong cyclic solutions guarantee goal reachability only under the assumption of *fairness*. In the presence of fairness it is supposed that all action outcomes, in a given state, would occur infinitely often.

Appendix C

Theorem

Any propositional temporal formula φ can be rewritten as a boolean combination of pure temporal formulas. (Gabbay, 1989)

Definition of ζ

$\zeta : \mathcal{R} \times \Sigma^* \to [0,1] \subseteq \mathbb{R}$ is defined as follows:

$$\zeta(\Psi, t) = \begin{cases} \dfrac{|\{i : t, i \models \alpha \text{ and } t, i \models \varphi\}|}{|\{i : t, i \models \alpha\}|}, & \text{if } |\{i : t, i \models \alpha\}| \neq 0; \\ 0, & \text{otherwise} \end{cases}$$

Equivalence between *reactive constraints*

$$\alpha \mapsto \varphi \equiv \bigvee_{j=1}^{m} (\varphi^{\blacktriangleleft} \wedge (\varphi^{\blacktriangledown} \wedge \alpha) \wedge \varphi^{\blacktriangleright})_j$$