LTL$_f$-based Trace Alignment: a Planning-based Approach

Fabio Patrizi$^1$

Sapienza University of Rome, Italy

patrizi@diag.uniroma1.it

Agents-VIC
May 7, 2021

$^1$Jointly with G. De Giacomo, F.M. Maggi, A. Marrella, A. Murano, G. Perelli
Process Mining: analysis of (business) processes starting from event logs [van der Aalst, 2016]
Events model process activities:

- \( a = \text{register request} \)
- \( b = \text{examine thoroughly} \)
- \( c = \text{examine casually} \)
- \( d = \text{check ticket} \)
- \( e = \text{decide} \)
- \( f = \text{reinitiate request} \)
- \( g = \text{pay compensation} \)
- \( h = \text{reject request} \)

Traces: event sequences modeling a process run

1. \( a, b, d, e, h \)
2. \( a, d, c, e, h \)
3. \( a, c, d, e, f, b, d, e, g \)
4. \( a, d, b, e, h \)

... 

Event Log: set of traces
Problems in Process Mining

Typical problems in Process Mining:

- Process Discovery: discover the underlying process that produced the log [Cook and Wolf, 1995]
- Conformance Checking: check whether a log conforms to a given process model [Rozinat and van der Aalst, 2008]
- Monitoring: monitor the current trace wrt a given process model [Ly et al., 2013]
Declarative Process Mining

**Imperative Process Model**

- Prescriptive specification
- Process fully specified
- Common Models:
  - Petri Net
  - Finite-state automaton
  - BPMN
  - ...

**Declarative Process Model**

- Descriptive specification
- Behaviors subject to constraints
- Behaviors allowed unless violating
- Constraints expressed in:
  - DECLARE
    - [Pesic et al., 2007]
  - LTL$_f$/LDL$_f$
    - [De Giacomo and Vardi, 2013]

Declarative Process Mining (this talk): Event Log + Declarative Process Model
Trace Alignment

Trace Alignment is a form of Conformance Checking

- Traces often *dirty* (involve human activities)
  - *spurious* or *missing* events
  - in general, *discrepancies* may be present
- Process model not enforced

Trace Alignment is the problem of *cleaning* and *repairing* dirty traces to make them compliant with the underlying process model [van der Aalst, 2016]

Solving the problem allows for:

- Identifying trace errors
- Taking corrective actions
- Measuring deviation wrt process model
Trace Alignment [Bose and van der Aalst, 2010] (declarative variant)

Given:
- A trace $\rho$ over a finite event alphabet $\Sigma = \{\sigma_1, \ldots, \sigma_n\}$
- A constraint $\varphi$ (expressed in some formal language, see below)
- A cost function $\text{cost}$ associating non-negative costs to additions and deletions of each event

Find trace $\rho'$ over $\Sigma$ s.t.:
- $\rho'$ satisfies $\varphi$, written $\rho \models \varphi$
- Cost $\text{cost}(\rho, \rho')$ of turning $\rho$ into $\rho'$ is minimal (wrt the cost of changes made to $\rho$)
Trace Alignment (in Declarative Process Mining)

Example

- $\rho = a \ a \ c \ b \ a \ c \ a$
- $\varphi =$ “whenever $a$ occurs, $b$ eventually occurs after $a$”
- all changes cost 1
- $\rho$ does not satisfy $\varphi$ (last $a$ has no matching $b$)
  1. Remove all $a$’s: $\rho_1 = a \ a \ c \ b \ a \ c \ a = c \ b \ c$, $cost(\rho, \rho_1) = 4$
  2. Add $b$ after every $a$: $\rho_2 = a \ b \ a \ b \ c \ a \ b \ c \ a \ b$, $cost(\rho, \rho_1) = 4$
  3. Add $b$ at end only: $\rho_3 = a \ a \ c \ b \ a \ c \ a \ b$, $cost(\rho, \rho_1) = 1$ (solution!)

Observations:
- if $\varphi$ satisfiable, repaired trace always exists
- we can think of $cost(\rho, \rho')$ as the distance of $\rho$ from $\varphi$ (if $\rho' \models \varphi$)
Applications

Some uses of trace alignment in BP:

- Check whether expected process is correctly executed and quantify deviation (costs)
- Identify and correct errors in process execution
- Identify common patterns among log traces

But also in AI:

- Use traces to model agent behaviors (e.g., executed actions)
- Use trace alignment to:
  - compare observed behavior with expected/desired one
  - quantify discrepancies and repair
  - identify behavioral patterns
  - ...

F. Patrizi (Sapienza)
A technique for trace alignment [De Giacomo et al., 2017]:

- automata-based approach
- solved with cost-optimal planning

Experimental results

(Extension to metric temporal constraints)
Standard language in Declarative PM is **DECLARE** [Pesic et al., 2007]

Defined through *templates*:

- **Existence**(a): activity *a* is executed at least once
  \[b\ a\ c\ a,\ b\ c\ c\]

- **Response**(a, *b*): if *a* occurs then *b* eventually occurs after *a*
  \[a\ c\ a,\ a\ c\ a\ b,\ c\ d\ c\]

- **Choice**(a, *b*): *a* or *b* (possibly both) eventually occur
  \[a\ c\ a,\ a\ c\ b,\ c\ c\ b,\ d\ c\ c\]

- **ChainResponse**(a, *b*): whenever *a* occurs, *b* immediately follows
  \[a\ c\ c\ b,\ d\ c\ c,\ d\ a\ b\]

- ...
**DECLARE** is a fragment of \( \text{LTL}_f \)

\[ \varphi := \text{true} \mid a \mid \neg \varphi \mid \varphi \land \varphi \mid X\varphi \mid \varphi U \varphi \]

- \( X\varphi \) ("next \( \varphi \)"): next event satisfies \( \varphi \)
- \( \varphi_1 U \varphi_2 \) ("\( \varphi_1 \) until \( \varphi_2 \)"): an event satisfying \( \varphi_2 \) eventually occurs (possibly now) and all events until then satisfy \( \varphi_1 \)

Standard abbreviations (Boolean as usual):
- \( F\varphi \equiv \text{true} U \varphi \) ("eventually \( \varphi \)"): \( \varphi \) eventually holds
- \( G\varphi \equiv \neg F\neg \varphi \) ("always \( \varphi \)"): \( \varphi \) always holds
Some examples

<table>
<thead>
<tr>
<th>DECLARE template</th>
<th>LTL$_f$ translation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{Existence}(a)$</td>
<td>$F a$</td>
</tr>
<tr>
<td>$\text{Response}(a, b)$</td>
<td>$G(a \rightarrow F b)$</td>
</tr>
<tr>
<td>$\text{Choice}(a, b)$</td>
<td>$F a \lor F b$</td>
</tr>
<tr>
<td>$\text{ChainResponse}(a, b)$</td>
<td>$G(a \rightarrow X b)$</td>
</tr>
</tbody>
</table>

We use full LTL$_f$ (in fact, could go even beyond and use LDL$_f$)
Trace Alignment with $\text{LTL}_f$ Constraints

Example

- $\rho = a\ a\ c\ b\ a\ c\ a$
- $\varphi = \mathbf{G}(a \rightarrow \mathbf{F}b)$ ("whenever $a$ occurs, $b$ eventually occurs after $a$")
- ...

...
**Theorem ([De Giacomo and Vardi, 2013])**

Every $\text{LTL}_f$ formula $\varphi$ has a corresponding (exponential) NFA $A_\varphi$ s.t.

\[
\text{for every trace } \rho: \rho \models \varphi \iff A_\varphi \text{ accepts } \rho
\]

\[
\varphi = F a \\
\varphi = G a \\
\varphi = G(a \rightarrow Fb)
\]
Automata-based Solution

Idea

Given:
- trace $\rho$
- constraint $\varphi$ (one, w.l.o.g.)

Define:
- Augmented trace automaton $\mathcal{T}^+$: accepts all modifications of $\rho$
- Augmented constraint automaton $\mathcal{A}^+$:
  accepts all traces that satisfy $\varphi$ plus all modifications of $\rho$ that satisfy $\varphi$

Find minimal-cost $\rho'$ s.t.:
- $\rho'$ is accepted by $\mathcal{T}^+$
- $\rho'$ is accepted by $\mathcal{A}^+$
Augmented trace automaton $\mathcal{T}^+$

Example:

- $\rho = a \ a \ c \ b \ a \ c \ a$

Accepts all modifications of $\rho$, with changes marked, e.g.:

1. Remove all $a$'s: $\text{del}_a \ \text{del}_a \ c \ b \ \text{del}_a \ c \ \text{del}_a$

2. Add $b$ after every $a$: $a \ \text{add}_b \ a \ \text{add}_b \ c \ b \ a \ \text{add}_b \ c \ a \ \text{add}_b$

3. Add $b$ at end only: $a \ a \ c \ b \ a \ c \ a \ \text{add}_b$
Augmented constraint automaton $\mathcal{A}^+$

Example:

- $\varphi = G(a \rightarrow Fb)$ (“whenever $a$ occurs, $b$ eventually occurs after $a$”)

Plain Constraint Automaton

Augmented Constraint automaton

Observe: accepts also traces that are not modifications of $\rho$
Automata-based Solution

Problem

Find **minimal-cost** $\rho'$ s.t.:

- $\rho'$ is accepted by $T^+$, i.e., changes wrt $\rho$ are marked by adds and dels
- $\rho'$ is accepted by $A^+$, i.e., $\rho'$ “repairs” $\rho$

Corresponds to finding minimal-cost accepting path $\rho'$ on product automaton $T^+ \times A^+$

- $\rho = a \ a \ c \ b \ a \ c \ a$, $\varphi = G(a \rightarrow Fb)$
- $\rho' = a \ a \ c \ b \ a \ c \ a \ add \ b$
Automata-based Solution

Complexity

Theorem

Trace alignment can be solved in time polynomial wrt $|\rho| \times 2^{|\varphi|}$

- Searching min-cost path in $T^+ \times A^+$: $O(N \log N)$, with $N = |T^+ \times A^+| = |T^+| \times |A^+|
- $|T^+| = O(|\rho|)$, time: $O(|\rho|)$
- $|A_\varphi| = O(2^{|\varphi|})$, time: $O(2^{|\varphi|})$
- $|A^+|: O(|A_\varphi|)$, time: $O(|A_\varphi|)$
Trace alignment as minimal-cost planning

Use planning to search for minimal-cost $\rho'$

- **Domain:**
  - Models product automaton $T^+ \times A^+$
  - *sync* actions with null cost model events
  - *add* and *del* actions with positive costs model changes to input trace
  - Dynamics model *synchronous* execution of all augmented automata (trace+constraints)

- **Problem:**
  - Initial state: all automata in their starting state
  - Goal: all automata in a final state

- **Solution:**
  - Minimal-cost goal-reaching sequence of actions
Trace alignment as minimal-cost planning

Use planning to search for minimal-cost $\rho'$

- Actions with costs model events and changes to $\rho$
- Adds and dels have (possibly different) positive cost
- Augmented automata (trace and constraints) modeled by domain
- Solution: sequence of actions that takes all automata to accepting state

TASK: find minimal-cost plan
Trace alignment as minimal-cost planning

- \( \rho = a \quad c \quad c \)
- \( \varphi_1 = F_a \)
- \( \varphi_2 = G(a \to Fb) \)
- \( \rho' = a \quad c \quad add \_b \quad c \)
Solving trace alignment with planning

The problem can be encoded in PDDL 2.1 [Fox and Long, 2003]

```pddl
(define (domain alignment)
  (:requirements :typing :disjunctive-preconditions :conditional-effects
  :universal-preconditions :action-costs)
  (:types trace_state automaton_state - state activity)
  (:predicates
    (trace t1 - trace_state ?e - activity t2 - trace_state)
    (cur_state ?s - state)
    (automaton s1 - automaton_state ?e - activity s2 - automaton_state)
    (final_state ?s - state)
  )
  (:functions total-cost)
)

(:action add
  :parameters (?e - activity)
  :effect (and
    (increase (total-cost) 1)
    (forall (?s1 s2 - automaton_state)
      (when (and (cur_state ?s1) (automaton ?s1 ?e ?s2))
        (and (not (cur_state ?s1))(cur_state ?s2)))
    )
  )
)

(:action sync
  :parameters (t1 - trace_state ?e - activity t2 - trace_state)
  :precondition (and (cur_state t1)(trace t1 ?e t2))
  :effect (and
    (not (cur_state t1))
    (cur_state t2)
    (forall (?s1 s2 - automaton_state)
      (when (and (cur_state ?s1) (automaton ?s1 ?e ?s2))
        (and (not (cur_state ?s1))(cur_state ?s2)))
    )
  )
)

(:action del
  :parameters (?t1 - trace_state ?e - activity ?t2 - trace_state)
  :precondition (and (cur_state ?t1)(trace ?t1 ?e ?t2))
  :effect (and
    (increase (total-cost) 1)
    (not (cur_state ?t1)) (cur_state ?t2)
  )
)
```

Various encodings are possible, huge impact on performance
Experiments

- Tested (cost-optimal) planners:
  - Fast-Downward [Helmert, 2006]
  - SymBA* [Torralba et al., 2016]
  - (Assumed unitary costs for adds and dels)

- Compared against
  - Ad-hoc SOA approach implemented in ProM
    [de Leoni and van der Aalst, 2013, de Leoni et al., 2015]

- Dataset
  - Real-life logs (loan application in Dutch financial institute), 16 constraints
  - Synthetic logs with 10, 15 and 20 constraints
Results

planning vs ad-hoc on real-life logs

<table>
<thead>
<tr>
<th>trace length</th>
<th>no. traces</th>
<th>Fast-Downward</th>
<th>SymbA-2\textsuperscript{*}</th>
<th>de Leoni et al.</th>
<th>Alignment Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>3-50</td>
<td>607</td>
<td>2.47</td>
<td>2.92</td>
<td>0.15</td>
<td>0.63</td>
</tr>
<tr>
<td>51-75</td>
<td>38</td>
<td>2.59</td>
<td>4.13</td>
<td>0.45</td>
<td>1.02</td>
</tr>
<tr>
<td>76-100</td>
<td>5</td>
<td>2.65</td>
<td>4.99</td>
<td>2.78</td>
<td>2.4</td>
</tr>
<tr>
<td>101-128</td>
<td>4</td>
<td>2.66</td>
<td>5.61</td>
<td>5.88</td>
<td>2.5</td>
</tr>
</tbody>
</table>

Total alignment time (seconds) vs Length of the trace.
Results

planning vs ad-hoc on synthetic logs

<table>
<thead>
<tr>
<th>Trace length</th>
<th>Fast-Downward</th>
<th>SymbA-2+</th>
<th>de Leoni et al.</th>
<th>Alignment Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 const. modified</td>
<td>10 constraints</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1-50</td>
<td>0.62</td>
<td>1.95</td>
<td>0.34</td>
<td>1.77</td>
</tr>
<tr>
<td>51-100</td>
<td>0.85</td>
<td>3.63</td>
<td>1.37</td>
<td>2.11</td>
</tr>
<tr>
<td>101-150</td>
<td>1.15</td>
<td>6.4</td>
<td>5.9</td>
<td>3.03</td>
</tr>
<tr>
<td>151-200</td>
<td>1.46</td>
<td>10.75</td>
<td>12.98</td>
<td>3.79</td>
</tr>
<tr>
<td>15 constraints</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1-50</td>
<td>1.97</td>
<td>3.49</td>
<td>1.08</td>
<td>1.71</td>
</tr>
<tr>
<td>51-100</td>
<td>2.79</td>
<td>5.3</td>
<td>6.64</td>
<td>2.23</td>
</tr>
<tr>
<td>101-150</td>
<td>3.61</td>
<td>8.26</td>
<td>24.05</td>
<td>3.07</td>
</tr>
<tr>
<td>151-200</td>
<td>5.12</td>
<td>13.63</td>
<td>91.39</td>
<td>4.2</td>
</tr>
<tr>
<td>20 constraints</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1-50</td>
<td>17.63</td>
<td>12.42</td>
<td>3.99</td>
<td>1.87</td>
</tr>
<tr>
<td>51-100</td>
<td>19.05</td>
<td>15.02</td>
<td>34.91</td>
<td>2.61</td>
</tr>
<tr>
<td>101-150</td>
<td>23.23</td>
<td>20.45</td>
<td>87.89</td>
<td>3.35</td>
</tr>
<tr>
<td>151-200</td>
<td>27.49</td>
<td>28.97</td>
<td>223.47</td>
<td>4.2</td>
</tr>
</tbody>
</table>

Our approach outperforms ProM by several orders of magnitude when the model becomes larger and the noise increases.

The ad-hoc approach implemented in ProM is faster only for short traces with little noise.
Results

1. Automata-based approach implemented with planning technology
   (available at: tinyurl.com/glymp9d)

2. Outperforms ad-hoc solutions
   - More scalable
   - More flexible (other planners can be plugged)

3. PDDL encoding has huge impact, usually little preprocessing required – great benefit
Ongoing Extensions

- Extending the planning-based approach to $\text{LDL}_f$ [De Giacomo and Vardi, 2013], i.e., $\text{LTL}_f$ with RE (as expressive as FSA)
- Testing new encodings
- Decidability of the problem for $\text{MTL}_f$ [Koymans, 1990], the time-metric extension of $\text{LTL}_f$ (see below)
Timed Trace Alignment

Ongoing

- Timed extension of Trace Alignment

- Conceptually analogous to basic version but:
  - Events have (rational) timestamps
  - Constraints involve metric temporal properties

- Results in a **significantly more challenging** setting
Timed Trace Alignment

Example

- $\rho = (a, 0) (a, 0.7) (c, 1.25) (a, 2.56) (c, 3.04) (b, 3.7)$
- $\varphi =$ “whenever $a$ occurs, $b$ eventually occurs within 2 time units (tu)”
- costs as before –no dependence on time

- $\rho$ does not satisfy $\varphi$: first two $a$’s have no matching $b$ within 2 tu
  - $(a, 0) (a, 0.7) (c, 1.25) (a, 2.56) (c, 3.04) (b, 3.7)$
Metric Temporal Logic ($\text{MTL}_f$)

$\varphi ::= \text{true} \mid e \mid \varphi \land \varphi \mid \neg \varphi \mid X_I \varphi \mid \varphi U_I \varphi$

$I$: open, closed, or semi-open interval with integer (or $\infty$) endpoints

- $X_I \varphi$: next event occurs within interval $I$ and satisfies $\varphi$
- $\varphi_1 U_I \varphi_2$: an event occurs within interval $I$ which satisfies $\varphi_2$ and all events until then satisfy $\varphi_1$

Standard abbreviations essentially same as $\text{LTL}_f$ (but with interval subscripts)

$\text{MTL}_f$ captures the timed extension of DECLARE
Timed Constraint Specification

Example

“whenever $a$ occurs, $b$ eventually occurs within 2 time units (tu)”: 

$$ \varphi = G_{[0,\infty]}(a \rightarrow F_{[0,2]}b) $$
Each $\text{MTL}_f$ formula $\varphi$ admits an automaton accepting the traces that satisfy $\varphi$

- 1-clock Alternating Timed Automata (1-ATA)
- Unfortunately, 1-ATA induce infinite-state, infinite-branching automata
- Automata-based Approach for NFA not suitable as-is

$\text{MTL}_f$ is decidable [Ouaknine and Worrell, 2007]: can check whether trace $\rho$ exists s.t. $\rho \models \varphi$

But: how to search for optimal $\rho$?

- Main obstacles: state- and branching-infiniteness
Timed Trace Alignment
Solution Intuition

From this:

- 1-ATA semantics
- infinite-branching
- infinite-state

To this:

- 1-ATA semantics abstraction
- finite-branching
- finite-state
Abstraction obtained by combining two results:
- (Computable) Abstraction for $\text{MTL}_f$ decidability [Ouaknine and Worrell, 2007]
- Reachability for well-structured transition systems [Finkel and Schnoebelen, 2001]

We extended the results to visit the whole (abstract) space of traces

Essentially same approach as for basic version:
- Construct Abstraction of Timed augmented trace 1-ATA
- Construct Abstraction of Timed augmented constraint 1-ATA
- Combine both in a (abstract) Product Automaton
- Search for optimal trace in Product Automaton

Complexity: non-primitive recursive (essentially, non-elementary)
Timed Trace Alignment

- Formalization as path-search on a 1-ATA
- Problem is solvable
- Complexity: non-primitive recursive 😊
Complexity

Untimed version

- Exponential wrt size of constraints
- Constraints typically small
- Behaves well in practice with planning-based approach (worst-case rarely shows up)

Timed version

- Non-elementary
- Constraints in practice are small
- Impact in practice? (Future work)
- Examples of tools with non-elem complexity but well-behaving in practice:
  - MONA: non-elementary
    [Henriksen et al., 1995]
  - Acceptable performance in practice
    [Zhu et al., 2017]
- (There is hope!)
Conclusions & Future Work

Summing up:
- Problem originated in Declarative Process Mining
- Practical and Speculative Relevance also to AI (Agent Behaviors)
- Two variants considered: untimed and timed
- AI-planning technique provides best results in untimed setting
- Timed setting solvable but practicality to be assessed

Future work:
- Extend untimed version to $L\bar{D}L_f$ and new PDDL encodings (ongoing)
- Study well-behaved fragments of $M\bar{T}L_f$ or effects of constraints on repairs (e.g., fixed number of adds within two events)
- Implement solution approaches for timed variant (planning-based?)
- Consider data-aware setting (where activities carry a payload) –particularly relevant to KR
Thank you!

Questions?
References I

Trace alignment in process mining: Opportunities for process diagnostics.
In Proc. of BPM '10.

Automating process discovery through event-data analysis.
In Proc. of ICSE'95.

On the disruptive effectiveness of automated planning for LTLF-based trace alignment.
In Proc. of AAAI'17.

Linear temporal logic and linear dynamic logic on finite traces.

An alignment-based framework to check the conformance of declarative process models and to preprocess event-log data.
Inf. Syst., 47:258–277.

Aligning event logs and process models for multi-perspective conformance checking: An approach based on integer linear programming.

Well-structured transition systems everywhere!
PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains.

The fast downward planning system.

Mona: Monadic second-order logic in practice.
In *Proc. of TACAS ’95*.

Specifying real-time properties with metric temporal logic.

A framework for the systematic comparison and evaluation of compliance monitoring approaches.
In *Proc. of EDOC’13*.

On the Decidability and Complexity of Metric Temporal Logic over Finite Words.

DECLARE: full support for loosely-structured processes.
In *Proc. of EDOC ’07*. 

