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

Process Mining: analysis of (business) processes starting from *event logs* [van der Aalst, 2016]

Event Log

Events model process activities:

a =register request, b =examine thoroughly

c =examine casually, d = check ticket

e =decide, f =reinitiate request,

g =pay compensation, h =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

case id	event id	properties		
		timestamp	activity	resource
1	35654423	30-12-2010:11.02	register request	Pete
	35654424	31-12-2010:10.06	examine thoroughly	Sue
	35654425	05-01-2011:15.12	check ticket	Mike
	35654426	06-01-2011:11.18	decide	Sara
	35654427	07-01-2011:14.24	reject request	Pete
2	35654483	30-12-2010:11.32	register request	Mike
	35654485	30-12-2010:12.12	check ticket	Mike
	35654487	30-12-2010:14.16	examine casually	Pete
	35654488	05-01-2011:11.22	decide	Sara
	35654489	08-01-2011:12.05	pay compensation	Ellen
3	35654521	30-12-2010:14.32	register request	Pete
	35654522	30-12-2010:15.06	examine casually	Mike
	35654524	30-12-2010:16.34	check ticket	Ellen
	35654525	06-01-2011:09.18	decide	Sara
	35654526	06-01-2011:12.18	reinitiate request	Sara
	35654527	06-01-2011:13.06	examine thoroughly	Sean
	35654530	08-01-2011:11.43	check ticket	Pete
	35654531	09-01-2011:09.55	decide	Sara
	35654533	15-01-2011:10.45	pay compensation	Ellen
4	35654641	06-01-2011:15.02	register request	Pete
	35654643	07-01-2011:12.06	check ticket	Mike
	35654644	08-01-2011:14.43	examine thoroughly	Sean
	35654645	09-01-2011:12.02	decide	Sara
	35654647	12-01-2011:15.44	reject request	Ellen

Problems in Process Mining

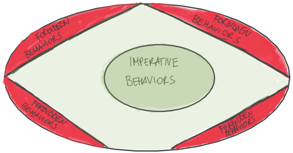
Typical problems in Process Mining:

- 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]
  - $\text{LTL}_f/\text{LDL}_f$ [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

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 (in Declarative Process Mining)

Trace Alignment [Bose and van der Aalst, 2010] (declarative variant)

Given:

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

Find trace ρ' over Σ s.t.:

- ρ' satisfies φ, written $\rho \models \varphi$
- Cost $cost(\rho, \rho')$ of turning ρ into ρ' is minimal (wrt the cost of changes made to ρ)

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

- ρ does not satisfy φ (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: $\rho_2 = a\ b\ a\ b\ c\ b\ a\ b\ c\ a\ b$, $cost(\rho, \rho_1) = 4$
  3. Add b at end only: $\rho_3 = a\ a\ c\ b\ a\ c\ a\ b$, $cost(\rho, \rho_1) = 1$ (solution!)

Observations:

- if φ satisfiable, repaired trace always exists
- we can think of $cost(\rho, \rho')$ as the distance of ρ from φ (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
  - ...

This Talk

1. A technique for trace alignment [De Giacomo et al., 2017]:
   - automata-based approach
   - solved with cost-optimal planning
2. Experimental results
3. (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 LTL$_f$

LTL$_f$ (linear-time temporal logic over finite traces [De Giacomo and Vardi, 2013])

$\varphi := \mathit{true} \mid a \mid \neg\varphi \mid \varphi \wedge \varphi \mid \mathbf{X}\varphi \mid \varphi \mathbf{U} \varphi$

- $\mathbf{X}\varphi$ ("next φ"): next event satisfies φ
- $\varphi_1 \mathbf{U} \varphi_2$ ("φ_1 until φ_2"): an event satisfying φ_2 eventually occurs (possibly now) and all events until then satisfy φ_1

Standard abbreviations (Boolean as usual):

- $\mathbf{F}\varphi \doteq \mathit{true}\mathbf{U}\varphi$ ("eventually φ"): φ eventually holds
- $\mathbf{G}\varphi \doteq \neg\mathbf{F}\neg\varphi$ ("always φ"): φ always holds

SAPIENZA UNIVERSITÀ DI ROMA

Some examples

DECLARE template	LTL$_f$ translation
Existence(a)	$\mathbf{F}a$
Response(a, b)	$\mathbf{G}(a \rightarrow \mathbf{F}b)$
Choice(a, b)	$\mathbf{F}a \vee \mathbf{F}b$
ChainResponse(a, b)	$\mathbf{G}(a \rightarrow \mathbf{X}b)$

We use full LTL$_f$ (in fact, could go even beyond and use LDL$_f$)

Trace Alignment with 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")
- ...

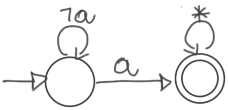
LTL$_f$ Formulas as NFAs

Theorem ([De Giacomo and Vardi, 2013])

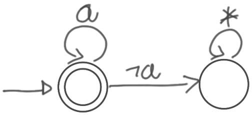
Every LTL$_f$ formula φ has a corresponding (exponential) NFA $\mathcal{A}_\varphi$ s.t.

$$\textit{for every trace } \rho : \ \rho \models \varphi \Leftrightarrow \ \mathcal{A}_\varphi \textit{ accepts } \rho$$

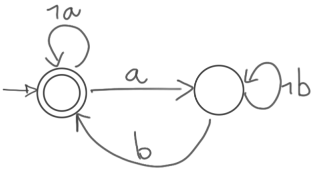
$\varphi = \mathbf{F}a$



$\varphi = \mathbf{G}a$



$\varphi = \mathbf{G}(a \rightarrow \mathbf{F}b)$



Given:

- trace ρ
- constraint φ (one, w.l.o.g.)

Define:

- *Augmented trace automaton* $\mathcal{T}^+$: accepts all modifications of ρ
- *Augmented constraint automaton* $\mathcal{A}^+$:
  accepts all traces that satisfy φ **plus** all modifications of ρ that satisfy φ

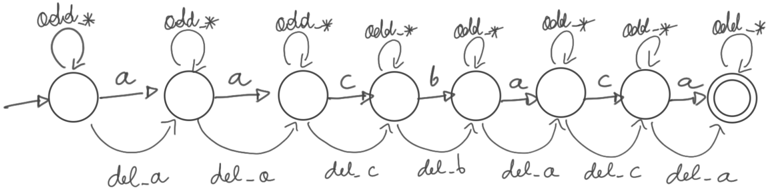
Find miminal-cost ρ' s.t.:

- ρ' is accepted by $\mathcal{T}^+$
- ρ' is accepted by $\mathcal{A}^+$

Augmented trace automaton $\mathcal{T}^+$

Example:

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



Accepts all modifications of ρ, with changes **marked**, e.g.:

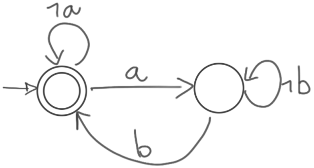
1. Remove all a's: $del\_a\ del\_a\ c\ b\ del\_a\ c\ del\_a$
2. Add b after every a: $a\ add\_b\ a\ add\_b\ c\ b\ a\ add\_b\ c\ a\ add\_b$
3. Add b at end only: $a\ a\ c\ b\ a\ c\ a\ add\_b$

Augmented constraint automaton $\mathcal{A}^+$

Example:

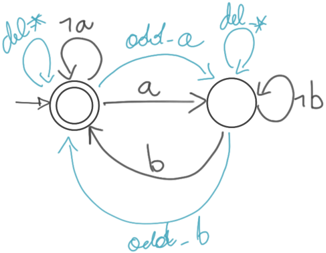
- $\varphi = \mathbf{G}(a \rightarrow \mathbf{F}b)$("whenever a occurs, b eventually occurs after a")

Plain Constraint Automaton



$\Longrightarrow$

Augmented Constraint automaton



Observe: accepts also traces that are not modifications of ρ

Automata-based Solution

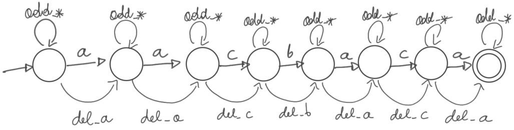
Problem

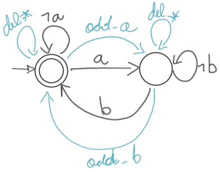
Find **miminal-cost** ρ' s.t.:

- ρ' is accepted by $\mathcal{T}^+$, i.e., changes wrt ρ are marked by adds and dels
- ρ' is accepted by $\mathcal{A}^+$, i.e., ρ' "repairs" ρ

Corresponds to finding minimal-cost accepting path ρ' on product automaton $\mathcal{T}^+ \times \mathcal{A}^+$

- $\rho = a\ a\ c\ b\ a\ c\ a$, $\varphi = \mathbf{G}(a \rightarrow \mathbf{F}b)$
- $\rho' = a\ a\ c\ b\ a\ c\ a\ add\_b$





SAPIENZA UNIVERSITÀ DI ROMA

Theorem

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

- *Searching min-cost path in* $\mathcal{T}^+ \times \mathcal{A}^+$*:* $\mathcal{O}(N \log N)$*, with* $N = |\mathcal{T}^+ \times \mathcal{A}^+| = |\mathcal{T}^+| \times |\mathcal{A}^+|$
- $|\mathcal{T}^+| = \mathcal{O}(|\rho|)$*, time:* $\mathcal{O}(|\rho|)$
- $|\mathcal{A}_\varphi| = \mathcal{O}(2^{|\varphi|})$*, time:* $\mathcal{O}(2^{|\varphi|})$
- $|\mathcal{A}^+|$*:* $\mathcal{O}(|\mathcal{A}_\varphi|)$*, time:* $\mathcal{O}(|\mathcal{A}_\varphi|)$

Trace alignment as minimal-cost planning

Use planning to search for minimal-cost ρ'

- Domain:
  - Models product automaton $\mathcal{T}^+ \times \mathcal{A}^+$
  - *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

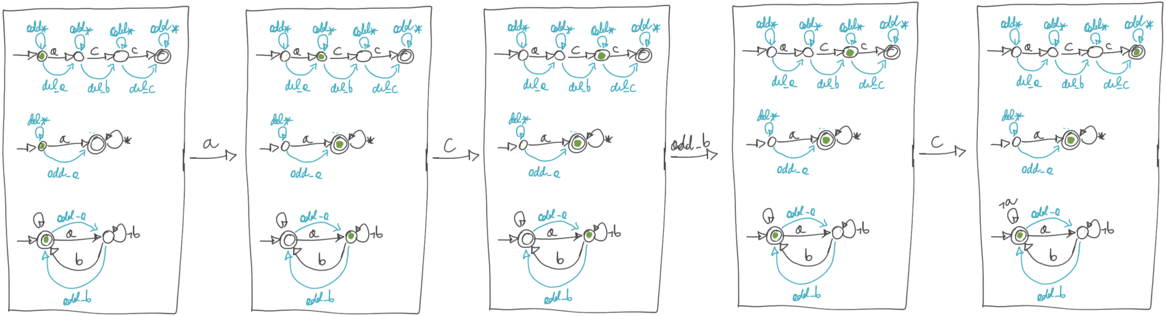
Sapienza Università di Roma

Use planning to search for minimal-cost ρ'

- Actions with costs model events and changes to ρ
- 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

- $\rho = a\ c\ c$
- $\varphi_1 = \mathbf{F} a$
- $\varphi_2 = \mathbf{G}(a \rightarrow \mathbf{F} b)$
- $\rho' = a\ c\ add\_b\ c$



Solving trace alignment with planning

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

```
(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

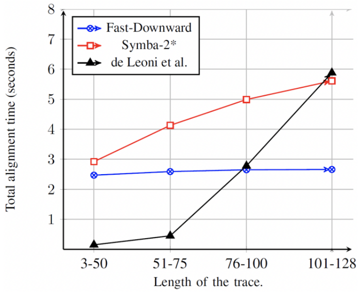
Experiments

- 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

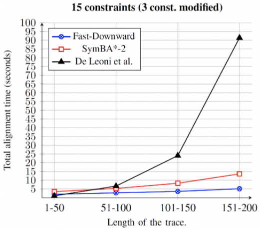
trace length	no. traces	Fast-Downward	SymbA-2*	de Leoni et al.	Alignment Cost
Real-life log		16 constraints			
3-50	607	2.47	2.92	0.15	0.63
51-75	38	2.59	4.13	0.45	1.02
76-100	5	2.65	4.99	2.78	2.4
101-128	4	2.66	5.61	5.88	2.5

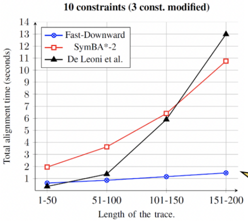


Results

planning vs ad-hoc on synthetic logs

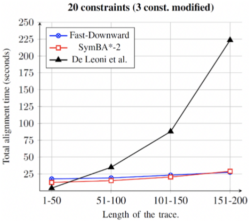
Trace length	Fast-Downward	SymbA-2*	de Leoni et al.	Alignment Cost
3 const. modified	10 constraints			
1-50	0.62	1.95	0.34	1.77
51-100	0.85	3.63	1.37	2.11
101-150	1.15	6.4	5.9	3.03
151-200	1.46	10.75	12.98	3.79
	15 constraints			
1-50	1.97	3.49	1.08	1.71
51-100	2.79	5.3	6.64	2.23
101-150	3.61	8.26	24.05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





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 LDL_f [De Giacomo and Vardi, 2013], i.e., LTL_f with RE
  (as expressive as FSA)

- Testing new encodings

- Decidability of the problem for MTL_f [Koymans, 1990], the time-metric extension of 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

SAPIENZA UNIVERSITÀ DI ROMA

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
- ρ does not satisfy φ: first two a's have no matching b within 2 tu
  - $(a, 0)\ (a, 0.7)\ (c, 1.25)\ (a, 2.56)\ (c, 3.04)\ (b, 3.7)$

SAPIENZA UNIVERSITÀ DI ROMA

Metric Temporal Logic (MTL_f)

$\varphi ::= \mathit{true} \mid e \mid \varphi \wedge \varphi \mid \neg\varphi \mid \mathbf{X}_I\varphi \mid \varphi \mathbf{U}_I \varphi$

I: open, closed, or semi-open interval with integer (or ∞) endpoints

- $\mathbf{X}_I\varphi$: next event occurs within interval I and satisfies φ
- $\varphi_1 \mathbf{U}_I \varphi_2$: an event occurs within interval I which satisfies φ_2 and all events until then satisfy φ_1

Standard abbreviations essentially same as LTL_f (but with interval subscripts)

MTL_f captures the timed extension of DECLARE

Timed Constraint Specification

Example

"whenever a occurs, b eventually occurs within 2 time units (tu)":

- $\varphi = \mathbf{G}_{[0,\infty]}(a \rightarrow \mathbf{F}_{[0,2]} b)$

Each MTL_f formula φ admits an automaton accepting the traces that satisfy φ

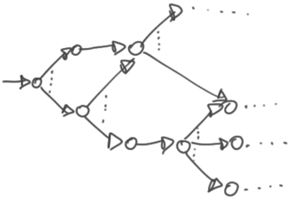
- 1-clock Alternating Timed Automata (1-ATA)
- Unfortunately, 1-ATA induce infinite-state, infinite-branching automata
- Automata-based Approach for NFA not suitable as-is

MTL_f is decidable [Ouaknine and Worrell, 2007]: can check whether trace ρ exists s.t. $\rho \models \varphi$

But: how to search for **optimal** ρ?

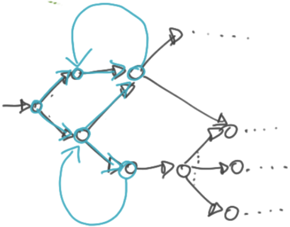
- Main obstacles: state- and branching-infiniteness

From this:



$\Longrightarrow$

To this:



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

- 1-ATA semantics **abstraction**
- finite-branching
- finite-state

Abstraction obtained by combining two results:

- (Computable) Abstraction for 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)

- 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 LDL_f and new PDDL encodings (ongoing)
- Study well-behaved fragments of MTL_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

Bose, R. P. J. C. and van der Aalst, W. M. P. (2010).
Trace alignment in process mining: Opportunities for process diagnostics.
In *Proc. of BPM '10*.

Cook, J. E. and Wolf, A. L. (1995).
Automating process discovery through event-data analysis.
In *Proc. of ICSE'95*.

De Giacomo, G., Maggi, F. M., Marrella, A., and Patrizi, F. (2017).
On the disruptive effectiveness of automated planning for ltl*f*-based trace alignment.
In *Proc. of AAAI'17*.

De Giacomo, G. and Vardi, M. Y. (2013).
Linear temporal logic and linear dynamic logic on finite traces.
In *Proc. of IJCAI'13*.

de Leoni, M., Maggi, F. M., and van der Aalst, W. M. P. (2015).
An alignment-based framework to check the conformance of declarative process models and to preprocess event-log data.
Inf. Syst., 47:258–277.

de Leoni, M. and van der Aalst, W. M. P. (2013).
Aligning event logs and process models for multi-perspective conformance checking: An approach based on integer linear programming.
In *Proc. of BPM '13*.

Finkel, A. and Schnoebelen, P. (2001).
Well-structured transition systems everywhere!
Theor. Comput. Sci., 256(1-2):63–92.

References II

Fox, M. and Long, D. (2003).
PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains.
J. Artif. Intell. Res.(JAIR), 20:61–124.

Helmert, M. (2006).
The fast downward planning system.
J. Artif. Intell. Res., 26:191–246.

Henriksen, J. G., Jensen, J. L., Jørgensen, M. E., Klarlund, N., Paige, R., Rauhe, T., and Sandholm, A. (1995).
Mona: Monadic second-order logic in practice.
In *Proc. of TACAS '95*.

Koymans, R. (1990).
Specifying real-time properties with metric temporal logic.
Real Time Syst., 2(4):255–299.

Ly, L. T., Maggi, F. M., Montali, M., Rinderle-Ma, S., and van der Aalst, W. M. P. (2013).
A framework for the systematic comparison and evaluation of compliance monitoring approaches.
In *Proc. of EDOC'13*.

Ouaknine, J. and Worrell, J. (2007).
On the Decidability and Complexity of Metric Temporal Logic over Finite Words.
Log. Methods Comput. Sci., 3(1).

Pesic, M., Schonenberg, H., and van der Aalst, W. M. P. (2007).
DECLARE: full support for loosely-structured processes.
In *Proc. of EDOC '07*.

References III

Rozinat, A. and van der Aalst, W. M. P. (2008).
Conformance checking of processes based on monitoring real behavior.
Inf. Syst., 33(1):64–95.

Torralba, Á., López, C. L., and Borrajo, D. (2016).
Abstraction heuristics for symbolic bidirectional search.
In *Proc. of IJCAI '16*.

van der Aalst, W. M. P. (2016).
Process Mining - Data Science in Action, Second Edition.
Springer.

Zhu, S., Tabajara, L. M., Li, J., Pu, G., and Vardi, M. Y. (2017).
Symbolic ltlf synthesis.
In *Proc. of IJCAI '17*.