Timed Trace Alignment with Metric Temporal Logic over Finite Traces

Giuseppe De Giacomo[1] , **Aniello Murano**[2] , **Fabio Patrizi**[1] , **Giuseppe Perelli**[1]
[1]Sapienza University of Rome, Italy
[2]University of Naples "Federico II", Italy
{degiacomo,patrizi,perelli}@diag.uniroma1.it, murano@na.infn.it

Abstract

Trace Alignment is a prominent problem in Declarative Business Process Management, which consists in identifying a minimal set of modifications that a log trace (produced by a system under execution) requires in order to be made compliant with a temporal specification. In its simplest form, log traces are sequences of events from a finite alphabet and specifications are written in DECLARE, a strict sublanguage of linear-time temporal logic over finite traces (LTL_f). The best approach for trace alignment has been developed in AI, using cost-optimal planning, and handles the whole LTL_f. In this paper, we study the timed version of trace alignment, where events are paired with timestamps and specifications are provided in metric temporal logic over finite traces (MTL_f), essentially a superlanguage of LTL_f. Due to the infiniteness of timestamps, this variant is substantially more challenging than the basic version, as the structures involved in the search are (uncountably) infinite-state, and calls for a more sophisticated machinery based on alternating (timed) automata, as opposed to the standard finite-state automata sufficient for the untimed version. The main contribution of the paper is a provably correct, effective technique for *Timed Trace Alignment* that takes advantage of results on MTL_f decidability as well as on reachability for *well-structured* transition systems.

1 Introduction

Trace alignment (Aalst, van der 2013) is a prominent problem in Declarative Business Process Management (BPM), which consists in minimally modifying, or *repairing*, a *log trace* to make it compliant with an input temporal specification. In its simplest formulation, traces are finite sequences of *events*, corresponding to the logs stored by a system under execution, and specifications capture properties that involve the mutual ordering of events, specified using the DECLARE language (van der Aalst, Pesic, and Schonenberg 2009), which is a fragment of the Linear-time Temporal Logic interpreted over finite traces LTL_f (De Giacomo and Vardi 2013).

While the Business Process (BP) community has devoted many efforts to this problem –see, e.g., promtools.org, (de Leoni, Maggi, and van der Aalst 2012), and (de Leoni, Maggi, and van der Aalst 2015), the best solution technique currently available comes from AI (De Giacomo et al. 2017) and is based on an automata-theoretic approach that takes advantage of the cost-optimal planning technology, e.g., (Helmert 2006; Torralba et al. 2014) to efficiently perform the search. Notably, such a technique is able to deal with any specification expressed in LTL_f, not only those allowed by DECLARE (which are defined in terms of LTL_f), a commonly adopted language in the BPM context.

A more sophisticated variant of the problem is *timed trace alignment*. In this variant, each event in a log trace is paired with a timestamp and the specifications can be expressed in a variety of forms, from pre-defined patterns (Lanz, Weber, and Reichert 2014), to formulas of more structured logics, such as the multi-perspective version of DECLARE, i.e., MP-DECLARE (Burattin, Maggi, and Sperduti 2016). MP-DECLARE is a fragment of Metric First-order Temporal Logic (Chomicki 1995) (MFOTL), a language that combines First-order Logic (FOL) and Metric Temporal Logic (Koymans 1990), both interpreted over finite traces, to express properties that concern event attributes and timestamps. Interestingly, for the timed version of trace alignment with specifications expressed in a rich language such as MP-DECLARE, a solution technique is not available yet, not even for the case where propositional formulas only are allowed in place of FO. Even more, it is not even known whether the existence of a solution is decidable.

In this paper, we study the timed version of trace alignment, with specifications expressed in metric temporal logic over finite traces (MTL_f), which is essentially a superlanguage of LTL_f. Importantly, MTL_f is a strictly more expressive language than the propositional version of MP-DECLARE thus, from the temporal perspective, we are addressing the problem in its entire generality. In this setting, trace alignment requires to repair a trace not only by adding or removing events from a finite alphabet but, crucially, by also adding timestamps from the infinite, uncountable, real space. This results in a substantially more challenging setting than that of the basic variant (De Giacomo et al. 2017), as the presence of time makes the structures involved in the search (uncountably) infinite-state, which requires a more sophisticated technical machinery, based on alternating timed automata, as opposed to the standard finite-state automata sufficient for the untimed version.

The contribution of the paper is threefold. Firstly, it proposes an automata-based formalization of timed trace alignment which reduces the problem of finding a minimal-cost solution (in terms of changes made to the input trace) to the search for an accepting path in an alternating timed automaton. Secondly, based on the above formulation, it shows that the problem is solvable. Thirdly, by proving solvability in a constructive way, the work devises the first technique for actually computing a solution in the timed setting.

Technically, to achieve these results, we build on two works, namely (Ouaknine and Worrell 2007) and (Finkel and Schnoebelen 2001). The former proves decidability of MTL_f by reducing the problem to non-emptiness of alternating timed automata and showing that this corresponds to the search for an accepting path in a *well-structured* transition system, while the latter provides the theoretical results on such transition systems to make the search effective. Broadly speaking, we generalize the approach described in (Ouaknine and Worrell 2007), which is aimed at showing the existence of a solution, to actually perform a search in the space of the solutions, and take advantage of the results from (Finkel and Schnoebelen 2001) to obtain termination of our technique.

We observe that, while the obtained technique is quite demanding from the complexity point of view, it solves the problem in its entire generality, as far as temporal aspects only are concerned, thus can serve as a reference for practical approaches that typically deal with less expressive fragments of MTL_f, such as the propositional fragment of MP-DECLARE.

2 Metric Temporal Logic and Alternating Timed Automata

In this section, we report (slightly modified) notions and results from (Ouaknine and Worrell 2007), which we need to introduce the problem and establish our results. In particular, we review Metric Temporal Logic over finite words and its relationship with alternating timed automata.

2.1 Metric Temporal Logic

We use Metric Temporal Logic over finite words (MTL_f) under the so-called *event-based* semantics where the temporal connectives quantify over a countable (finite, in this paper) set of positions in a (finite) timed word.

Let Σ be a finite alphabet of *events*. The syntax of MTL_f is as follows:

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

where: $e \in \mathcal{P}$ and $I \subseteq \mathbb{R}_{\geq 0}$ can be an open, closed or semi-open interval with endpoints in $\mathbb{N} \cup \{\infty\}$. The classical abbreviations of LTL_f can be suitably adapted to the temporal variant: (release) $\psi_1\, \mathbf{R}_I\, \psi_2 \doteq \neg(\neg\psi_1\, \mathbf{U}\, \neg\psi_2)$ (eventually) $\mathbf{F}_I \doteq \mathsf{true}\, \mathbf{U}_I\, \varphi$ and (always) $\mathbf{G}_I\, \varphi \doteq \neg\, \mathbf{F}_I\, \neg\varphi$. In $\psi_1\, \mathbf{U}_{[0,\infty]}\, \psi_2$, we omit $[0, \infty]$ and simply write $\psi_1\, \mathbf{U}\, \psi_2$.

MTL_f formulas are interpreted over *finite timed words*. A *(finite) timed word* over an event alphabet Σ is a finite sequence $\rho = (\sigma_1, \tau_1) \cdots (\sigma_\ell, \tau_\ell)$, where: $\sigma_i \in \Sigma$; $\tau_i \in \mathbb{R}_{\geq 0}$; and $\tau_i \leq \tau_{i+1}$ for $1 \leq i < \ell$ (monotonicity). A pair $(\sigma, \tau) \in \Sigma \times \mathbb{R}_{\geq 0}$ is a *timed event*; sometimes, to mark the distinction between a timed event and a simple event $\sigma \in \Sigma$, we refer to the latter as an *untimed event*; the value τ is a *timestamp*. By $|\rho|$ we denote the *length* of ρ, i.e., $|\rho| = \ell$.

The semantics of MTL_f is defined based on an MTL_f formula φ, a finite timed word ρ, and a position i s.t. $1 \leq i \leq |\rho|$, as follows:

- $(\rho, i) \models \mathsf{true}$;
- $(\rho, i) \models e$ iff $\sigma_i = e$;
- $(\rho, i) \models \varphi_1 \wedge \varphi_2$ iff $(\rho, i) \models \varphi_1$ and $(\rho, i) \models \varphi_2$;
- $(\rho, i) \models \neg\varphi$ iff $(\rho, i) \not\models \varphi$;
- $(\rho, i) \models \mathbf{X}_I\, \varphi$ iff $i < |\rho|$, $\tau_{i+1} - \tau_i \in I$, and $(\rho, i+1) \not\models \varphi$;
- $(\rho, i) \models \varphi_1\, \mathbf{U}_I\, \varphi_2$ iff $\tau_j - \tau_i \in I$, $(\rho, j) \models \varphi_2$, for some j s.t. $i \leq j < |\rho|$, and $(\rho, k) \models \varphi_1$ for all k s.t. $i \leq k < j$.

A timed word ρ is said to satisfy an MTL_f formula φ, written $\rho \models \varphi$, iff $(\rho, 1) \models \varphi$.

Intuitively, the logic can be seen as a form of LTL_f where the temporal operators are required to be satisfied within a temporal frame defined by the associated interval I. For instance, formulae $\mathbf{X}_{[2,4]}\, e$ and $e\, \mathbf{U}_{[2,4]}\, f$ can be read respectively as: "next event is e and occurs within 2 and 4 (time units from now)" and " within 2 and 4, f occurs and before that e continuously holds". We stress that interval endpoints must be integer. It is immediate to see that, by setting $I = [0, \infty)$, MTL_f reduces to LTL_f.

Example 1. *MTL_f extends the properties expressible in LTL_f by adding time constraints. For example, $\varphi = \mathbf{G}(a \rightarrow \mathbf{F}_{\leq 1}\, b)$ [1] expresses the classical request/grant requirement but constrains every grant event to occur within one time unit from every occurrence of a request event.*

Consider the trace $\rho_1 = (c, 0.5)(a, 1.5)(b, 2.49)(d, 5)$. We have $\rho_1 \models \varphi$, as every occurrence of event a is followed by an occurrence of b within one time unit. On the contrary, for trace $\rho_2 = (c, 0.5)(a, 1.5)(b, 2.51)(d, 5)$ we have $\rho_2 \not\models \varphi$, as the only occurrence of a is not met by any event b within one time unit.

Finally, observe that scaling the time constraints in a formula and the timestamps in a trace preserves satisfaction:
$(c, 50)(a, 150)(b, 249)(d, 500) \models \mathbf{G}(a \rightarrow \mathbf{F}_{\leq 100}\, b)$
$(c, 50)(a, 150)(b, 251)(d, 500) \not\models \mathbf{G}(a \rightarrow \mathbf{F}_{\leq 100}\, b)$.

2.2 1-clock Alternating Timed Automata

Analogously to LTL_f, for every MTL_f formula φ, there exists an automaton $\mathcal{A}_\varphi$ that accepts exactly all the words that satisfy φ. In this setting, $\mathcal{A}_\varphi$ is not a standard (alternating) automaton but a *1-clock alternating timed automaton*. Intuitively, 1ATA's are finite-state alternating automata equipped with a clock that keeps track of the time elapsed since its last reset, and whose transitions, which may reset the clock, depend on clock conditions.

Formally, a *1-clock alternating timed automaton* (1ATA) is a tuple $\mathcal{A} = (\Sigma, L, l_0, F, \delta)$ where: Σ is a finite event alphabet; L is a finite set of *locations*; $l_0 \in L$ is the *initial* location; $F \subseteq L$ is the set of *final* locations; $\delta : L \times \Sigma \mapsto \Phi(L)$ is the *transition function*, with $\Phi(L)$ the following language of formulas ϕ, defined over the set of locations L and the (implicit) clock x:

$$\phi = \mathbf{true} \mid \mathbf{false} \mid \phi \wedge \phi \mid \phi \vee \phi \mid l \mid x \bowtie c \mid x.\phi,$$

[1] For simplicity, we write $[0, z]$ as $\leq z$, and similarly for the other comparisons.

where $c \in \mathbb{N}$, $\bowtie \in \{<, \leq, =, \geq, >\}$, and $l \in L$.

Intuitively, 1ATA's can be thought of as timed, multi-headed, nondeterministic automata. By *timed* we mean that a clock variable is present which stores the time elapsed since the last clock reset and locations are paired with clock values, forming pairs called *states*; by *multi-headed*, we mean that the automaton can be in a set, called *configuration*, of finitely (though unboundedly) many states at once; by *nondeterministic*, we mean that the automaton can move from one to (possibly infinitely) many configurations at each step. State-transitions, which induce configuration transitions, are distinguished into *universal* and *existential*, with the term *alternating* expressing the fact that in the automaton both classes coexist (and not that they strictly alternate). A universal transition defines a move from one to many mandatory states, as if taken in conjunction, while an existential transition defines a move from one to many possible states, as if taken in disjunction. Configuration transitions model either time flow or event occurrences. The former increase clock values while leaving locations unchanged, the latter may change location and reset the clock. Configuration transitions are defined based on a minimal-model semantics of $\Phi(L)$, which we report next.

Consider a 1ATA $\mathcal{A} = (\Sigma, L, l_0, F, \delta)$. A *state* of $\mathcal{A}$ is a pair $(l, v) \in L \times V$, with $V = [0, max] \cup \top$, max the maximum constant mentioned in δ, and $\top$ a special symbol modeling any value $v > max$. Let $Q = L \times V$ be the set of all possible states. Formulae $\phi \in \Phi(L)$ are interpreted over sets of states $M \subseteq Q$ and clock values $v \in V$, as follows:

- $(M, v) \models \mathbf{true}$;
- $(M, v) \not\models \mathbf{false}$;
- $(M, v) \models \phi_1 \wedge \phi_2$ iff $(M, v) \models \phi_1$ and $(M, v) \models \phi_2$;
- $(M, v) \models \phi_1 \vee \phi_2$ iff $(M, v) \models \phi_1$ or $(M, v) \models \phi_2$;
- $(M, v) \models l$ iff $(l, v) \in M$;
- $(M, v) \models x \bowtie c$ iff $v \bowtie c$;
- $(M, v) \models x.\phi$ iff $(M, 0) \models \phi$.

$M \subseteq Q$ is a *model* of $\phi \in \Phi(L)$ wrt v, if $M, v \models \phi$; M is a *minimal model* of ϕ wrt v if it is minimal wrt (strict) set inclusion, i.e., for no other $M' \subset M$, we have $M', v \models \phi$.

We can now introduce the labelled transition system (LTS) $\mathcal{T}_\mathcal{A}$ induced by $\mathcal{A}$, which models how the configurations of $\mathcal{A}$ evolve as timed events occur.

The *induced LTS* $\mathcal{T}_\mathcal{A}$ of a 1ATA $\mathcal{A} = (\Sigma, L, l_0, F, \delta)$ is the tuple $\mathcal{T}_\mathcal{A} = (\Sigma, K, C_0, A, \rightsquigarrow, \rightarrow)$, where:

- Σ is the same alphabet as the one of $\mathcal{A}$;
- $K = 2^Q$, is the set of possible *configurations*;
- $C_0 = \{(l_0, 0)\} \in K$ is the *initial* configuration;
- $A = \{C \in K \mid (l, v) \in C \Rightarrow l \in F\}$ is the set of *accepting* configurations;
- $\rightsquigarrow \subseteq K \times \mathbb{R}_{\geq 0} \times K$ is the *delay* transition relation s.t. $C \overset{d}{\rightsquigarrow} C'$ iff $C' = C + d \doteq \{(l, v + d) \mid (l, v) \in C\}$;
- $\rightarrow \subseteq K \times \Sigma \times K$ is the *event* (instantaneous) transition relation s.t. $C \xrightarrow{e} C'$ iff $C' = \bigcup_{i \in I} M_i$, with $C = \{(l_i, v_i)\}_{i \in I}$ and M_i a minimal model of $\delta(l_i, e)$ wrt to v_i.

Intuitively, delay transitions account for the time elapsed between consecutive events and event transitions account for the location transitions triggered by timed events. Observe that event transitions can reset clocks but not assign them to arbitrary values. Informally, we can think of each minimal model $M_i \subseteq C'$ as the set of successors of state (l_i, v_i) under event e, all of which are entered after the transition.

Runs of $\mathcal{A}$ are defined based on $\mathcal{T}_\mathcal{A}$. A *run of $\mathcal{A}$ on a timed word* $\rho = (\sigma_1, \tau_1) \cdots (\sigma_\ell, \tau_\ell)$ is a sequence

$$\varrho = C_0 \overset{d_1}{\rightsquigarrow} C_1 \xrightarrow{\sigma_1} \cdots \overset{d_\ell}{\rightsquigarrow} C_{2\ell-1} \xrightarrow{\sigma_\ell} C_{2\ell},$$

where C_0 is the initial configuration and $d_i = \tau_i - \tau_{i-1}$, for $i = 1, \ldots, \ell$, with $\tau_0 = 0$, by convention. We refer to ϱ also as a *run of* $\mathcal{T}_\mathcal{A}$, and say that ϱ is a run of $\mathcal{T}_\mathcal{A}$ *on* ρ. The length of a run ϱ as above is $|\varrho| = \ell$.

Notice that, as a result of $\mathcal{A}$'s existential nondeterminism, a timed word ρ can give raise to many runs of $\mathcal{A}$, while universal nondeterminism is responsible for the presence of many states in each configuration. A run is *accepting* if so is its last configuration $C_{2\ell}$, and a timed word ρ is *accepted* by $\mathcal{A}$ (equivalently, $\mathcal{T}_\mathcal{A}$), if there exists a run of $\mathcal{A}$ (eq., $\mathcal{T}_\mathcal{A}$) on ρ that is accepting. The language of finite timed words accepted by $\mathcal{A}$ is denoted by $\mathcal{L}_f(\mathcal{A})$.

Example 2. *(Ouaknine and Worrell 2007) propose a general construction procedure for the 1ATA $\mathcal{A}_\varphi$ associated with an* MTL$_f$ *φ. The construction is omitted here. We show, however, its application on the formula φ of Ex. 1.*

First, φ is rewritten in negation normal form:

$$\varphi = \mathsf{false}\,\mathbf{R}(\neg a \vee (\mathsf{true}\,\mathbf{U}_{\leq 1}\, b))$$

Then, the automaton $\mathcal{A}_\varphi = (\Sigma, L, l_0, F, \delta)$ is obtained as follows:

- $\Sigma = \{a, b\}$*;*
- $L = \{l_0, l_1, l_2, \neg a, b, \}$*; where*
  - $l_0 = \varphi_{init}$*,*
  - $l_1 = \mathsf{false}\,\mathbf{R}(\neg a \vee (\mathsf{true}\,\mathbf{U}_{\leq 1}\, b)$*, and*
  - $l_2 = \mathsf{true}\,\mathbf{U}_{\leq 1}\, b$*;*
- $F = \{l_0, l_1\}$*;*
- *The transition function δ is defined as follows:* [2]
  - $\delta(l_0, \sigma) = x.(\delta(l_1, \sigma))$*;*
  - $\delta(l_1, \sigma) = (x.(\delta(\neg a \vee (l_2), \sigma)) \vee x \notin \mathbb{R}) \wedge (x.\delta(\mathsf{false}, \sigma) \vee l_1) = (x.(\delta(\neg a, \sigma) \vee \delta(l_2))) \wedge l_1$*;*
  - $\delta(\neg a \vee l_2, \sigma) = \delta(\neg a, \sigma) \vee \delta(l_2, \sigma)$*;*
  - $\delta(l_2, \sigma) = (\delta(b, \sigma) \wedge x \leq 1) \vee (x.\delta(\mathsf{true}, \sigma) \wedge l_2)$*;*
  - $\delta(b, \sigma) = \begin{cases} \mathbf{true} & \textit{if } b = \sigma \\ \mathbf{false} & \textit{o/w} \end{cases}$
  - $\delta(\neg a, \sigma) = \begin{cases} \mathbf{false} & \textit{if } a = \sigma \\ \mathbf{true} & \textit{o/w} \end{cases}$

[2]We define only those transitions that can actually occur in some run over a timed word.

The construction essentially follows the syntactic tree of the formula, like in the case of alternating automata for LTL_f*, while adding the time constraints occurring in temporal operators to the transition function.*

Consider the execution of this automaton, shown in Fig. 1, on the word $\rho_1 = (c, 0.5)(a, 1.5)(b, 2.49)(d, 5)$ *of Ex. 1. Notice that event* a *generates the configuration with location*

$$
\begin{aligned}
&\{(l_0, 0)\} \overset{0.5}{\rightsquigarrow} \{(l_0, 0.5)\} \xrightarrow{c} \\
&\{(l_1, 0.5)\} \overset{1}{\rightsquigarrow} \{(l_1, 1.5)\} \xrightarrow{a} \\
&\{(l_1, 1.5), (l_2, 0)\} \overset{0.99}{\rightsquigarrow} \{(l_1, 2.49), (l_2, 0.99)\} \xrightarrow{b} \\
&\{(l_1, 2.99)\} \overset{2.51}{\rightsquigarrow} \{(l_1, 5)\} \xrightarrow{d} \\
&\{(l_1, 5)\}
\end{aligned}
$$

Figure 1: Execution of the automaton $\mathcal{A}_\varphi$ over the trace ρ_1.

l_2*, which is then removed from the configuration by* b*, only if its associated clock has value less or equal than* 1.

(Ouaknine and Worrell 2007) shows that *emptiness of* $\mathcal{A}$, i.e., the problem of checking whether $\mathcal{A}$ contains some accepting run, is decidable, thus showing that MTL_f (satisfiability) is decidable. The main challenge in obtaining this result has to do not only with the fact that, by the presence of real values, $\mathcal{T}_\mathcal{A}$ contains infinitely many configurations, but also, and more critically, with the fact that one cannot straightforwardly derive a finite structure abstracting $\mathcal{T}_\mathcal{A}$ to perform the check. This is overcome by resorting to the theory of *well-structured transition systems* (Finkel and Schnoebelen 2001), which offers a technique to solve, essentially, a variant of reachability. In this paper, we adapt that technique to deal with the more general problem of computing a finite set of accepting runs, that will be needed in order to solve the *timed trace alignment problem*.

3 Timed Trace Alignment

In this section, we generalize the framework for trace alignment introduced in (De Giacomo et al. 2017) to the case where events are *timed*, i.e., have an associated timestamp.

3.1 The problem

We model log traces as finite timed words, and specifications as MTL_f formulas. This is consistent with the standard setting in BPM where, in particular, specifications are expressed in the propositional variant of MP-DECLARE (Burattin, Maggi, and Sperduti 2016), which is a fragment of MTL_f.

Since log traces are obtained as logs of actual systems under execution, we can assume that they have rational timestamps; for technical reasons, it will be convenient to make them integer by introducing a suitable scale factor, used also to scale all the other involved values. We thus assume, without loss of generality, that timestamps occurring in log traces are integer only. This is a standard assumption in the BPM context (Burattin, Maggi, and Sperduti 2016). Apart from this, real values are allowed in general anywhere else.

We consider fixed the alphabet Σ of events. Given a log trace $\rho = (\sigma_1, \tau_1) \cdots (\sigma_\ell, \tau_\ell)$, we consider three types of atomic operations applicable to timed events in ρ: *skip*, i.e., the timed event is left unchanged; *delete*, i.e., a timed event is deleted from its position; and *add*, i.e., a new timed event is added at a certain position (by preserving timestamp monotonicity). We call *repair* any modification of a trace, through *add* or *delete* operations. Obviously, through repairs one can produce new log traces ρ', in fact all possible ones. We focus on those modifications that preserve timestamp monotonicity, i.e., s.t. ρ' is again a timed word.

Let $cost(\rho, \rho')$ be the function that takes as input two log traces ρ and ρ' and returns the minimal number of repairs needed to transform ρ into ρ'.

Definition 1 (Trace alignment)**.** *An instance of* trace alignment *is a pair* $\mathcal{I} = (\rho, \varphi)$ *where* ρ *is a log trace and* φ *an* MTL_f *formula, both over the same event alphabet* Σ*. A solution to* $\mathcal{I}$ *is a trace* ρ' *that minimizes* $cost(\rho, \rho')$ *and s.t.* $\rho' \models \varphi$*.*

Example 3. *Consider the* MTL_f *formula* φ *and the traces* ρ_1 *and* ρ_2 *of Ex. 1. The pairs* $\mathcal{I}_1 = (\rho_1, \varphi)$ *and* $\mathcal{I}_2(\rho_2, \varphi)$ *are instances of timed trace alignment. Since* $\rho_1 \models \varphi$*, the minimal cost solution for* $\mathcal{I}_1$ *is* ρ_1 *itself, as* $\rho_1 \models \varphi$ *and* $cost(\rho_1, \rho_1) = 0$*. As to* $\mathcal{I}_2$*, two optimal solutions are* $\hat{\rho}_2 = (c, 0.5)(b, 2.51)(d, 5)$*, which is obtained from* ρ_2 *by removing* $(a, 1.5)$*, and* $\hat{\rho}'_2 = (c, 0.5)(a, 1.5)(b, 2.49)(b, 2.51)(d, 5)$*, obtained by adding* $(b, 2.49)$*.*

We observe that since MTL_f is decidable (Ouaknine and Worrell 2007), so is checking whether, given φ, a solution exists. However, in this paper, we are interested in finding an optimal trace that solves the problem, not just establishing its existence. For this, we need to review the framework used for obtaining decidability and then exploit it to obtain our results.

3.2 An Automata-based Approach for Timed Trace Alignment

Timed trace alignment can be reduced to the problem of finding an accepting run in a suitable 1ATA obtained as the cross-product of two further 1ATA's, the *augmented trace automaton* and the *augmented constraint automaton*.

Definition 2 (Trace Automaton)**.** *Given a log trace* $\rho = (\sigma_1, \tau_1) \cdots (\sigma_\ell, \tau_\ell)$ *over* Σ*, the* trace automaton of ρ *is the 1ATA* $\mathcal{A}_\rho = (\Sigma_\rho, L_\rho, l_{\rho 0}, F_\rho, \delta_\rho)$*, s.t.:*

- $\Sigma_\rho = \Sigma$;
- $L_\rho = \{l_0, l_1, \ldots, l_\ell\}$;
- $l_{\rho 0} = l_0$;
- $F_\rho = \{l_\ell\}$;
- δ_ρ *is s.t.* $\delta_\rho(l_i, \sigma_{i+1}) = l_{i+1} \wedge x = \tau_{i+1}$*, for* $i = 0, \ldots, \ell - 1$*, and is undefined otherwise.*

Intuitively, $\mathcal{A}_\rho$ is a $1ATA$ that accepts exactly ρ, i.e., $\mathcal{L}(\mathcal{A}_\rho) = \{\rho\}$. We recall that timestamps are integer, thus the formulae from $\Phi(L)$ returned by δ can check that timestamps have a specific (integer) value. This would not be possible if timestamps were not integer, as $\Phi(L)$ formulae can compare clocks only against integers.

Example 4. *Consider again the trace ρ_2 of Ex. 1 with all timestamps* scaled *to natural numbers. The corresponding trace automaton $\mathcal{A}_{\rho_2}$ is depicted in Fig. 2. Observe that every transition is triggered only when the correct event appears at its corresponding timestamp. It is easy to see that the automaton accepts only the trace ρ_2. Moreover, observe that such 1ATA is* deterministic, *in the sense that every reachable configuration has a single successor.*

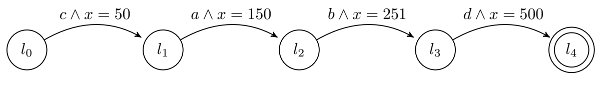

Figure 2: The trace automaton $\mathcal{A}_{\rho_2}$ for the trace ρ_2.

We then augment $\mathcal{A}_\rho$ in such a way that it accepts all the traces obtainable from ρ by applying some repairs. We represent repairs by means of fresh untimed events added to Σ.

Definition 3 (Augmented Trace Automaton)**.** *Given a log trace ρ, consider its corresponding trace automaton $\mathcal{A}_\rho = (\Sigma_\rho, L_\rho, l_{\rho 0}, F_\rho, \delta_\rho)$. The* augmented trace automaton *of ρ is the 1ATA $\mathcal{A}^+_\rho = (\Sigma^+_\rho, L^+_\rho, l^+_{\rho 0}, F^+_\rho, \delta^+_\rho)$, where:*

- $\Sigma^+_\rho = \Sigma \cup \{\mathsf{del}\} \cup \bigcup_{\sigma\in\Sigma}\{\mathsf{add}_\sigma\}$, *with* del *and each* add_σ *fresh events;*
- $L^+_\rho = L_\rho = \{l_0, l_1, \ldots, l_\ell\}$*;*
- $l^+_{\rho 0} = l_{\rho 0} = l_0$*;*
- $F^+_\rho = F_\rho = \{l_\ell\}$*;*
- δ_ρ *is s.t.:*
  - $\delta^+_\rho(l_i, \sigma_{i+1}) = \delta_\rho(l_i, \sigma_{i+1})$, *for* $i = 0, \ldots, \ell - 1$*;*
  - $\delta^+_\rho(l_i, \mathsf{del}) = \delta_\rho(l_i, \sigma_{i+1})$, *for* $i = 0, \ldots, \ell - 1$*;*
  - $\delta^+_\rho(l, \mathsf{add}_\sigma) = l$, *for all* $\sigma \in \Sigma$*;*
  - δ^+_ρ *is undefined otherwise.*

Intuitively, the automaton $\mathcal{A}^+_\rho$ accepts sequences of atomic operations applied to ρ. A skip operation occurs when the very same timed event occurring in ρ is read, in which case the transition simply copycats that of the trace automaton (on same location and timed event). When del is read, the automaton moves to the next state as if the deleted event had been read; as a consequence, the accepted word will contain the symbol del instead of the deleted one. Observe that the timestamp of the timed event containing del must be the same as that of the deleted timed event; in this way, each occurrence of del in the accepted word can be associated with the timed event deleted from ρ. Finally, when add occurs, the automaton reads the event and loops on its current state; as a consequence, the accepted word may contain arbitrary many occurrences of add.

Observe that the words ρ^+ read by $\mathcal{A}^+_\rho$ are not proper log traces, as some events they contain are not from Σ; rather, they are sequences of atomic operations applied to ρ. However, one can easily obtain the corresponding repaired trace $\hat{\rho}$ by simply removing all the timed events containing del and replacing each timed event $(\mathsf{add}_\sigma, \tau)$ with (σ, τ). For this reason, with a slight abuse of notation, we use traces containing repairs such as ρ^+ as (their corresponding) log traces.

Example 5. *Consider again the trace automaton $\mathcal{A}_{\rho_2}$ of Ex. 4. The augmentation is obtained by including transitions on* add *and* del *as shown in Fig. 3. Every transition from l_i to l_{i+1} in the trace automaton, with $i \in \{0, 1, 2, 3\}$, is paired with a new transition labeled by* del, *denoting that the automaton moves to the successor locations on delete operations. Moreover, for every location l_i, a loopy transition on add_σ is added per untimed event $\sigma \in \Sigma$, denoting that a new event has been included in the trace.*

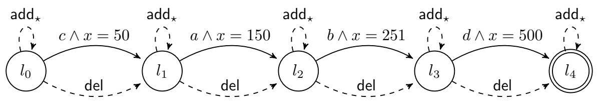

Figure 3: The augmented trace automaton $\mathcal{A}^+_{\rho_2}$ for the trace ρ_2.

The other 1ATA we define is the *augmented constraint automaton.* Given an MTL_f formula φ, let $\mathcal{A}_\varphi = (\Sigma_\varphi, L_\varphi, l_{\varphi 0}, F_\varphi, \delta_\varphi)$ be a 1ATA accepting all and only the log traces ρ s.t. $\rho \models \varphi$. We call $\mathcal{A}_\varphi$ the *constraint automaton.* We now augment the constraint automaton to obtain a new automaton accepting all and only the timed words obtained by repairing ρ.

Definition 4 (Augmented Constraint Automaton)**.** *Given an MTL_f formula φ, consider its corresponding 1ATA $\mathcal{A}_\varphi = (\Sigma_\varphi, L_\varphi, l_{\varphi 0}, F_\varphi, \delta_\varphi)$. The* augmented constraint automaton *of $\mathcal{I}$' is the 1ATA $\mathcal{A}^+_\varphi = (\Sigma^+_\varphi, L^+_\varphi, l^+_{\varphi 0}, F^+_\varphi, \delta^+_\varphi)$, where:*

- $\Sigma^+_\varphi = \Sigma \cup \{\mathsf{del}\} \cup \bigcup_{\sigma\in\Sigma}\{\mathsf{add}_\sigma\}$ *is the same input alphabet as that of the augmented trace automaton;*
- $L^+_\varphi = L_\varphi$*;*
- $l^+_{\varphi 0} = l_{\varphi 0}$*;*
- $F^+_\varphi = F_\varphi$*;*
- δ_φ *is s.t.:*
  - *for all $\sigma \in \Sigma_\varphi$ and $l \in L_\varphi$, we have that* $\delta^+_\varphi(l, \sigma) = \delta_\varphi(l, \sigma)$, *and* $\delta^+_\varphi(l, \mathsf{add}_\sigma) = \delta_\varphi(l, \sigma)$*;*
  - $\delta^+_\varphi(l, \mathsf{del}) = l$*;*
  - δ^+_φ *is undefined otherwise.*

Intuitively, the augmentation $\mathcal{A}^+_\varphi$ copycats the execution of $\mathcal{A}_\varphi$ when it reads either an original symbol σ or the corresponding addition add_σ.

Example 6. *Consider the* MTL *formula φ of Ex. 1 and the corresponding 1ATA $\mathcal{A}_\varphi$ shown in Ex. 2. The transition function of the augmented constraint automaton $\mathcal{A}^+_\varphi$ is then defined as:*

- $\delta^+_\varphi(l_i, \sigma) = \delta^+_\varphi(l_i, \mathsf{add}_\sigma) = \delta(l_i, \sigma)$ $\qquad (i = 0, 1, 2)$*;*
- $\delta^+_\varphi(\neg a, \sigma) = \delta^+_\varphi(\neg a, \mathsf{add}_\sigma) = \delta(\neg a, \sigma)$*;*
- $\delta^+_\varphi(b, \sigma) = \delta^+_\varphi(b, \mathsf{add}_\sigma) = \delta(b, \sigma)$*;*
- $\delta^+_\varphi(l, \mathsf{del}) = l$, *for every* $l \in L^+_\varphi$

where δ is the transition function of the automaton $\mathcal{A}_\varphi$.

The relevance of the above defined (augmented) automata $\mathcal{A}^+_\rho$ and $\mathcal{A}^+_\varphi$ wrt (timed) trace alignment resides in the fact that they can be suitably combined in a further automaton, namely the *product automaton*, which can be used to find a solution to the problem, if any.

Definition 5 (Product automaton). *Given two 1ATA's $\mathcal{A}_i = (\Sigma, L_i, l_{i0}, F_i, \delta_i)$ ($i = 1, 2$) over the same input alphabet Σ, with disjoint sets of location L_1 and L_2 the* product *of $\mathcal{A}_1$ and $\mathcal{A}_2$ is the 1ATA $\mathcal{A}_1 \otimes \mathcal{A}_2 = (\Sigma, L_1 \cup L_2 \cup \{l_0\}, l_0, F_1 \cup F_2, \delta)$, where l_0 is a fresh location not in L_1 or L_2 and δ is s.t.:*

- *$\delta(l_0, \sigma) = \delta_1(l_{10}, \sigma) \wedge \delta_2(l_{20}, \sigma)$;*
- *$\delta(l, \sigma) = \delta_i(l, \sigma)$, if $l \in L_i$ (recall that $L_1 \cap L_2 = \emptyset$).*

The so-defined automaton is s.t. $\mathcal{L}(\mathcal{A}_1 \otimes \mathcal{A}_2) = \mathcal{L}(\mathcal{A}_1) \cap \mathcal{L}(\mathcal{A}_2)$, i.e., it accepts all and only those words accepted by both $\mathcal{A}_1$ and $\mathcal{A}_2$.

Let ρ be a timed log trace and φ an MTL_f formula, both over Σ. Consider the corresponding (augmented) trace and constraint automata $\mathcal{A}^+_\rho$ and $\mathcal{A}^+_\varphi$, and their product $\mathcal{A}_{\rho,\varphi} = \mathcal{A}^+_\rho \otimes \mathcal{A}^+_\varphi$. The following result says that $\mathcal{L}(\mathcal{A}_{\rho,\varphi})$ is the solution space of the instance (ρ, φ).

Theorem 1. *The product automaton $\mathcal{A}_{\rho,\varphi}$ accepts all and only those repairs ρ^+ of ρ such that $\hat{\rho} \models \varphi$, where $\hat{\rho}$ is obtained from ρ^+ by removing the occurrences of (del, τ) and replacing the occurrences of $(\mathsf{add}_\sigma, \tau)$ with (σ, τ).*

Example 7. *Consider the instance $\mathcal{I} = (\rho_2, \varphi)$, with: $\rho_2 = (c, 50)(a, 150)(b, 251)(d, 500)$ the log trace obtained by scaling that of Ex. 1 of a factor 100, thus making all the timestamps integer; and φ the formula from the same example, with the integer constants occurring in the temporal operators scaled by 100 as well. Let $\mathcal{A}^+_{\rho_2}$ be the augmented trace automaton of ρ_2, $\mathcal{A}^+_\varphi$ the augmented constraint automaton of φ, and $\mathcal{A}_{\rho_2,\varphi} = \mathcal{A}^+_{\rho_2} \otimes \mathcal{A}^+_\varphi$ the corresponding product automaton.*
The word $\hat{\rho_2} = (c, 50)(a, 150)(\mathsf{add}_b, 240)(\mathsf{del}, 251)(d, 500)$ is accepted by both $\mathcal{A}^+_{\rho_2}$ and $\mathcal{A}^+_\varphi$, hence $\mathcal{A}_{\rho_2,\varphi}$, thus it is immediate to see that $\hat{\rho}_2$ is a (possibly non-optimal) solution to $\mathcal{I}$.

Therefore, we can limit the search for the optimal solution only to the words accepted by $\mathcal{A}_{\rho,\varphi}$, which can be obtained, more specifically, starting from the runs of the LTS $\mathcal{T}_{\rho,\varphi}$ induced by $\mathcal{A}_{\rho,\varphi}$. Clearly, it is not obvious how the search should be performed, as $\mathcal{T}_{\rho,\varphi}$ is in general infinite-state, with uncountably many nodes and transitions, a feature that constitutes a major obstacle in approaching the problem.

4 Solving Timed Trace Alignment

Ideally, in order to find an optimal solution to timed trace alignment, one should explore the set of accepting runs of the LTS $\mathcal{T}_{\rho,\varphi}$ induced by $\mathcal{A}_{\rho,\varphi}$, and then select the corresponding accepted word, if any, containing a minimal number of repairs. Obviously, by $\mathcal{T}_{\rho,\varphi}$'s infiniteness, a brute-force approach is not viable. In this section we show that the search can be effectively performed on a finite fragment of $\mathcal{T}_{\rho,\varphi}$. This is done by suitably adapting the construction presented in (Ouaknine and Worrell 2007) to the present setting.

4.1 The Induced Rational LTS

Consider a 1ATA $\mathcal{A} = (\Sigma, L, l_0, F, \delta)$, take a corresponding state (l, v), and assume that the clock value v is not integer, i.e., $\lfloor v \rfloor < v < \lceil v \rceil$. For an event $e \in \Sigma$, consider the transition $\phi = \delta(l, e)$ and let (l', v') be a state in the minimal model M of $\delta(l, e)$ wrt v. In other words, (l', v') is a "successor" of (l, v) under e. Now, change the clock value v by an arbitrary (positive or negative) quantity ϵ s.t. $\lfloor v \rfloor < v + \epsilon < \lceil v \rceil$ and consider the state $(l, v + \epsilon)$. By the semantics of $\Phi(L)$, it follows that if $v' \neq 0$ the state $(l', v' + \epsilon)$ is a successor (under e) of $(l, v + \epsilon)$, and if $v' = 0$, then $(l', 0)$ is a successor. This occurs because clock conditions in ϕ can compare v only against integer constants, thus are insensitive to changes in v that are small enough to keep v in the interval $(\lfloor v \rfloor, \lceil v \rceil)$. Clearly, this is not true for an integer v, as arbitrary small changes can be (potentially) detected by clock conditions in ϕ. Notice that if (l', v') is an accepting state then so is (l', w), for any w, as being accepting for a state depends only on the location.

The intuition above essentially says that location transitions outgoing from a state (l, v) are invariant wrt delays that maintain $v \in (\lfloor v \rfloor, \lceil v \rceil)$. Thus, since when looking for an accepting run we are ultimately interested in the evolution of locations, and because time steps do not affect locations directly, we can focus only on those time steps that move a clock from an interval to the next one. This is formalized below and lifted from states to configurations.

Partition the interval $V = [0, max] \cup \top$ (recall that max is the maximum constant mentioned in $\mathcal{A}$ and $\top$ stands for any value in (max, ∞)) into so-called *clock regions*: $r_0 = \{0\}, r_1 = (0, 1), \ldots, r_{2i} = \{i\}, r_{2i+1} = (i, i + 1), \ldots, r_{2max+1} = \{\top\}$. For a clock value v, let $r(v) = r_i$ iff $v \in r_i$.

Now, consider the LTS $\mathcal{W}_\mathcal{A} = (\Sigma \cup \{\epsilon\}, K, C_0, A, \to)$, where:

- $K \subseteq 2^Q$, is the set of possible configurations, with Q the set of possible $\mathcal{A}$-states;
- $C_0 = \{(l_0, 0)\} \in K$ is the initial configuration;
- $A = \{C \in K \mid (l, v) \in C \Rightarrow l \in F\}$ is the set of *accepting* configurations;
- $\to \subseteq K \times \Sigma \cup \{\epsilon\} \times K$ is the transition relation s.t.:
  - $C \xrightarrow{\epsilon} C'$ iff either $C = C' = \emptyset$ or $C' = C + d$, with:
    1. $d = (1 - \mu)/2$ and μ the maximum fractional part among all the clock values occurring in C (0 for $\top$), if there exists some state $(l, v) \in C$ with an integer $v \in [0, max]$;
    2. $d = (1 - \mu)$ otherwise.
  - $C \xrightarrow{a} C'$, with $a \in \Sigma$, if $C \xrightarrow{a} C'$ as in the LTS $\mathcal{T}_\mathcal{A}$ induced by $\mathcal{A}$.

Essentially, the delay d in the ϵ-transitions is chosen in such a way that, after the transition, only the clock values that, in the current configuration, are closest to their next region enter it. If there is no "next region", i.e., all clocks are beyond max, the default value $d = 1$ is used. Obviously, all clocks uniformly increase their value by d.

Since event transitions are a subset of $\mathcal{T}_\mathcal{A}$'s, we can think of $\mathcal{W}_\mathcal{A}$ as an LTS obtained from $\mathcal{T}_\mathcal{A}$ by considering only selected time steps, in particular those that make the clock values of each state move to their next region as discussed above (equivalently, despite the formal difference due to the fact that $\mathcal{T}_\mathcal{A}$ has two transition relations while $\mathcal{W}_\mathcal{A}$ one only, we can see $\mathcal{W}_\mathcal{A}$ as a fragment of $\mathcal{T}_\mathcal{A}$). Let $\overset{\epsilon^*}{\to}$ denote the reflexive transitive closure of $\overset{\epsilon}{\to}$ in $\mathcal{W}_\mathcal{A}$. We have the following result.

Theorem 2. *If $\varrho = C_0 \overset{\epsilon^*}{\to} C_1 \overset{\sigma_1}{\to} \cdots \overset{\epsilon^*}{\to} C_{2\ell-1} \overset{\sigma_\ell}{\to} C_{2\ell}$ is a run of $\mathcal{W}_\mathcal{A}$ then $\varrho' = C_0 \overset{d_1}{\rightsquigarrow} C_1 \overset{\sigma_1}{\to} \cdots \overset{d_\ell}{\rightsquigarrow} C_{2\ell-1} \overset{\sigma_\ell}{\to} C_{2\ell}$ is a run of $\mathcal{T}_\mathcal{A}$, where, for $i = 0, \dots, \ell-1$, each d_i is the (rational) delay s.t. $C_{2i+1} = C_{2i} + d_{i+1}$.*

Proof. Immediate consequence of $\mathcal{W}_\mathcal{A}$'s definition. □

In other words, $\mathcal{W}_\mathcal{A}$-runs can be rewritten as runs of $\mathcal{T}_\mathcal{A}$, and thus as $\mathcal{A}$-runs.

Given a timed word $\rho = (\sigma_1, \tau_1), \dots, (\sigma_\ell, \tau_\ell)$, a $\mathcal{W}_\mathcal{A}$-run $\varrho = C_0 \overset{\epsilon^*}{\to} C_1 \overset{\sigma_1}{\to} \cdots \overset{\epsilon^*}{\to} C_{2\ell-1} \overset{\sigma_\ell}{\to} C_{2\ell}$ is said to be a *run of $\mathcal{W}_\mathcal{A}$ on ρ*, if the delay associated with the i-th ϵ-transition is $d_i = \tau_i - \tau_{i-1}$ ($\tau_0 = 0$ by convention), formally, if $C_{2i+1} = C_{2i} + (\tau_{i+1} - \tau_i)$, for $i = 0, \dots, \ell-1$. The notions of accepting run and accepted word extend to $\mathcal{W}_\mathcal{A}$ in the obvious way.

Notice that, by the definition of $\mathcal{W}_\mathcal{A}$, all clock values of $\mathcal{W}_\mathcal{A}$-states are rational; for this reason, $\mathcal{W}_\mathcal{A}$-configurations are called *rational configurations*, and $\mathcal{W}_\mathcal{A}$ the *rational LTS induced by $\mathcal{A}$*. By this and the fact that every state admits only a finite number of successors, namely one under ϵ and finitely many under the events, it follows that $\mathcal{W}_\mathcal{A}$ contains only countably many states, as opposed to the uncountably many of $\mathcal{T}_\mathcal{A}$ due to the infinitely many delays it allows for. As a consequence, since given a configuration C, its successors C' s.t. $C \overset{\sigma}{\to} C'$ on all events σ are computable (through the transition function of $\mathcal{A}$), we have that $\mathcal{W}_\mathcal{A}$ can be effectively constructed, up to a fixed depth.

Theorem 3. *The induced rational LTS $\mathcal{W}_\mathcal{A}$ of a 1ATA $\mathcal{A}$ is computable up to a given depth.*

The relationship of interest between $\mathcal{T}_\mathcal{A}$ and $\mathcal{W}_\mathcal{A}$ can be established by introducing the following equivalence relation among configurations. Let $\equiv \subseteq 2^Q \times 2^Q$ be the equivalence relation between configurations s.t. $C \equiv C'$ iff there exists a bijection $f : C \mapsto C'$ between the states in C and C' s.t.: (i) if $f(l, v) = f(l', v')$ then $l = l'$ and $v, v' \in r_i$, for some i; and (ii) if $f(l, v) = f(l, v')$ and $f(g, w) = f(g, w')$ then $frac(v) \leq frac(v')$ implies $frac(w) \leq frac(w')$, where $frac(\cdot)$ is the function that returns the fractional part of a number. Intuitively, two configurations are equivalent iff they contain exactly the same state, modulo modifications that preserve: (i) locations and clock regions, and (ii) the relative order of their clock values wrt the distance from the respective previous regions. Notice that equivalent configurations are either both accepting or none.

We have the following crucial result.

Theorem 4. *Given a 1ATA $\mathcal{A}$, the LTS $\mathcal{T}_\mathcal{A}$ induced by $\mathcal{A}$ admits a run $\varrho = C_0 \overset{d_1}{\rightsquigarrow} C_1 \overset{\sigma_1}{\to} \cdots \overset{d_\ell}{\rightsquigarrow} C_{2\ell-1} \overset{\sigma_\ell}{\to} C_{2\ell}$, iff the rational LTS $\mathcal{W}_\mathcal{A}$ induced by $\mathcal{A}$ admits a run $\varrho' = C_0 \overset{\epsilon^*}{\to} C'_1 \overset{\sigma_1}{\to} \cdots \overset{\epsilon^*}{\to} C'_{2\ell-1} \overset{\sigma_\ell}{\to} C'_{2\ell}$, s.t. $C_i \equiv C'_i$, for $i = 0, \dots, 2\ell$.*

Proof. The If-part is an immediate consequence of Thm. 2. The Only-if part is in the proof of Prop. 4.9 of (Ouaknine and Worrell 2007), which states that a configuration C is reachable (from C_0) in $\mathcal{T}_\mathcal{A}$ only if a configuration C' s.t. $C' \equiv C$ is reachable in $\mathcal{W}_\mathcal{A}$. The proof shows how to construct a $\mathcal{W}_\mathcal{A}$-run that reaches C' from a $\mathcal{T}_\mathcal{A}$-run that reaches C. When applied to ϱ, the construction produces ϱ'. □

The result above implies that we can search for an accepting run of $\mathcal{T}_\mathcal{A}$ by performing the search in $\mathcal{W}_\mathcal{A}$.

4.2 Searching over the Induced Rational LTS

In this section, we take advantage of the results above to search for an optimal solution to timed trace alignment in the rational LTS. We start by deriving useful bounds on length and cost of all optimal solution traces.

Theorem 5. *Let $\mathcal{I} = (\rho, \varphi)$ be an instance of timed trace alignment and ρ' a minimal-length trace accepted by $\mathcal{A}_\varphi$, with $|\rho'| = m$. If $\hat{\rho}^*$ is an optimal solution to $\mathcal{I}$ then:*

- *$cost(\rho, \hat{\rho}^*) \leq |\rho| + m$, and*
- *$|\rho| \leq |\hat{\rho}^*| \leq 2|\rho| + m$.*

Proof. Let $\rho = (\sigma_1, \tau_1) \cdots (\sigma_\ell, \tau_\ell)$ and $\rho' = (\sigma'_1, \tau'_1) \cdots (\sigma'_m, \tau'_m)$. Consider the ordered sequence $\tau''_1 \leq \ldots \leq \tau''_{\ell+m}$ obtained from the union of the timestamps in ρ and ρ' [3]. Let $\hat{\rho}$ be the trace obtained by shuffling del and add operations in such a way as to transform ρ into ρ', i.e.:

$$\hat{\rho} = (\mathsf{op}_1, \tau''_1) \cdots (\mathsf{op}_{\ell+m}, \tau''_{\ell+m})$$

with

$$\mathsf{op}_i = \begin{cases} \mathsf{add}_{\sigma_j}, \text{ if } \tau''_i = \tau'_j \\ \mathsf{del}, \text{ otherwise} \end{cases}$$

Obviously, $\hat{\rho}$ is a (possibly non-optimal) solution to $\mathcal{I}$, as so is ρ', with cost $cost(\rho, \hat{\rho}) = |\rho| + m$, due to the fact that both del and add repairs have unitary cost. Consequently, for an optimal solution $\hat{\rho}^*$ we have $cost(\rho, \hat{\rho}^*) \leq |\rho| + m$.

Regarding the length of $\hat{\rho}^*$, observe that, by Thm. 1, $\hat{\rho}^*$ must be accepted by the product automaton $\mathcal{A}_{\rho,\varphi}$, thus by the augmented trace automaton $\mathcal{A}^+_\rho$, which is immediate to see that requires $|\hat{\rho}^*| \geq |\rho|$. Moreover, since skip repairs have null cost, their inclusion in a solution does not affect the solution's cost, but only its length. Thus, since there can be at most $|\rho|$ skips, it follows that $cost(\rho, \hat{\rho}^*) \geq |\hat{\rho}^*| - |\rho|$. By combining the two bounds on $cost(\rho, \hat{\rho}^*)$, we obtain

$$|\hat{\rho}^*| - |\rho| \leq cost(\rho, \hat{\rho}^*) \leq |\rho| + m,$$

which implies $|\hat{\rho}^*| - |\rho| \leq |\rho| + m$, thus $|\hat{\rho}^*| \leq 2|\rho| + m$. □

[3] We assume, without loss of generality, that all timestamps are different.

Thus, there exists an upper bound on the length of the optimal solutions. We now address the crucial point of its computation and then show how it can be used to find an optimal solution to timed trace alignment.

(Ouaknine and Worrell 2007) showed that the existence of an accepting run in a 1ATA $\mathcal{A}$ can be checked by traversing its induced rational LTS $\mathcal{W}_{\mathcal{A}}$, searching for an accepting configuration. Since $\mathcal{W}_{\mathcal{A}}$ is in general infinite, the traversal is carried out subject to a termination criterion, which is derived by resorting to the theory of *well-structured* transition systems.

Definition 6 (Downward well-structured transition system)**.** *A* downward well-structured *transition system is a finite-branching transition system $\mathcal{T} = (Q, \preceq, \rightarrow)$ equipped with a relation $\preceq \subseteq Q \times Q$ between states, s.t.:*

- *$\preceq$ is a* well-quasi-ordering*, i.e., a reflexive and transitive relation s.t. for every infinite sequence $q_1, q_2, \ldots$, there exist two indices $i < j$ with $q_i \preceq q_j$;*
- *$\preceq$ is* downward compatible *with $\rightarrow$, that is, for all $q_1' \preceq q_1$ and transition $q_1 \rightarrow q_2$, there exists a sequence $q_1' \rightarrow \cdots \rightarrow q_2'$ s.t. $q_2' \preceq q_2$.*

For this class of transition systems, under the assumption that $\preceq$ is decidable and that the successors of every state are computable, reachability of a set of states $V \subseteq Q$ from some initial state q_0 is decidable (Finkel and Schnoebelen 2001); in particular, a reachable state in V can be sought by traversing the states of $\mathcal{T}$ starting from q_0, moving to the next state according to $\rightarrow$, and terminating the search along a branch whenever a state q is reached s.t., for a previously visited state q', it holds that $q' \preceq q$.

Since it can be proven that the rational LTS $\mathcal{W}_{\mathcal{A}}$ induced by a 1ATA $\mathcal{A}$ is indeed a downward well-structured transition system, where containment between configurations $\subseteq$ is the well-quasi-order $\preceq$, it follows that reachability of an accepting $\mathcal{T}_{\mathcal{A}}$-configuration is decidable (Ouaknine and Worrell 2007). By combining these results, we easily obtain the following.

Theorem 6. *The set of minimal-length words accepted by a 1ATA $\mathcal{A}$ is computable.*

Proof. Consequence of the fact that $\mathcal{W}_{\mathcal{A}}$ is a downward well-structured transition system, Thm. 5 of (Finkel and Schnoebelen 2001), and Thm. 4. The search can be performed by traversing $\mathcal{W}_{\mathcal{A}}$ in a breadth-first fashion, terminating the search along a branch –i.e., *closing* the branch– as soon as a configuration C is reached which includes some previously visited configuration C', i.e., $C' \subseteq C$. The search terminates either at the first level containing an accepting configuration, in which case all the accepting runs up to that level are collected, or when all branches are closed, meaning that no accepting run exists. Termination in the latter case is guaranteed by Thm. 5 of (Finkel and Schnoebelen 2001). The fact that the collected runs are minimal-length $\mathcal{T}_{\mathcal{A}}$ accepting runs is a consequence of the above construction and Thm. 4. From the collected accepting runs, it is immediate to obtain the corresponding words. $\square$

This result yields the following.

Corollary 1. *The length m of a shortest word accepted by a 1ATA $\mathcal{A}$ is computable.*

Proof. The value m can be easily obtained by executing the search procedure described in Thm. 6, storing the current depth m, and terminating as soon as an accepting run, if any, is found. If no accepting run is found, m can be set to a special value, e.g., negative, to indicate that the solution $\mathcal{L}(\mathcal{A})$ is empty. $\square$

Now, consider a timed-trace-alignment instance $\mathcal{I} = (\rho, \varphi)$. Take the product automaton $\mathcal{A}_{\rho,\varphi} = \mathcal{A}_{\rho}^{+} \otimes \mathcal{A}_{\varphi}^{+}$, its induced LTS $\mathcal{T}_{\rho,\varphi}$, and the induced rational LTS $\mathcal{W}_{\rho,\varphi}$. The next result implies that an optimal solution to $\mathcal{I} = (\rho, \varphi)$ can be searched for on $\mathcal{W}_{\rho,\varphi}$ instead of on $\mathcal{T}_{\rho,\varphi}$.

Theorem 7. *If there exists a solution $\hat{\rho}$ to a timed-trace-alignment instance $\mathcal{I} = (\rho, \varphi)$ then there exists a solution $\hat{\rho}'$ with $cost(\rho, \hat{\rho}) = cost(\rho, \hat{\rho}')$ that is accepted by the induced rational LTS $\mathcal{W}_{\rho,\varphi}$ of $\mathcal{A}_{\rho,\varphi}$.*

Proof. Let $\rho = (\sigma_1, \tau_1) \cdots (\sigma_\ell, \tau_\ell)$ and $\hat{\rho} = (\hat{\sigma}_1, \hat{\tau}_1) \cdots (\hat{\sigma}_\ell, \hat{\tau}_m)$. Define n as the number of timed events occurring in both ρ and $\hat{\rho}$, i.e., s.t. $(\hat{\sigma}_i, \hat{\tau}_i) = (\sigma_i, \tau_i)$. Obviously, $cost(\rho, \hat{\rho}) = |\hat{\rho}| - n$. We next show the existence of a trace $\hat{\rho}'$ s.t. $|\hat{\rho}'| = |\hat{\rho}|$ which contains n timed events that match those of ρ.

Since $\hat{\rho}$ is a solution to $\mathcal{I}$, by Thm. 1, $\hat{\rho} \in \mathcal{L}(\mathcal{A}_{\rho,\varphi})$. Let $\hat{\varrho} = C_0 \overset{d_1}{\rightsquigarrow} C_1 \xrightarrow{\hat{\sigma}_1} \cdots \overset{d_m}{\rightsquigarrow} C_{2m-1} \xrightarrow{\hat{\sigma}_m} C_{2m}$, with $d_i = \hat{\tau}_i - \hat{\tau}_{i-1}$, ($i = 1, \ldots, m$, recall that $\hat{\tau}_0 = 0$), be an accepting run of $\mathcal{T}_{\rho,\varphi}$ on $\hat{\rho}$. By Thm. 4, $\mathcal{W}_{\rho,\varphi}$ admits a trace $\varrho' = C_0 \xrightarrow{\epsilon^*} C_1' \xrightarrow{\hat{\sigma}_1} \cdots \xrightarrow{\epsilon^*} C_{2m-1}' \xrightarrow{\hat{\sigma}_m} C_{2m}'$, s.t. $C_i \equiv C_i'$. Since $C_{2m} \equiv C_{2m}'$, ϱ' is accepting. Then, by Thm. 2, there exists an accepting run of $\mathcal{T}_{\rho,\varphi}$, i.e., $\hat{\varrho}' = C_0 \xrightarrow{d_1'} C_1' \xrightarrow{\hat{\sigma}_1} \cdots \xrightarrow{d_m'} C_{2m-1}' \xrightarrow{\hat{\sigma}_m} C_{2m}'$.

Let $\hat{\rho}' = (\hat{\sigma}_1', \hat{\tau}_1') \cdots (\hat{\sigma}_m', \hat{\tau}_m')$ be the log trace that induces $\hat{\varrho}'$ (formally, $\hat{\varrho}'$ is a run of $\mathcal{T}_{\rho,\varphi}$ on $\hat{\rho}'$). Since $\hat{\varrho}'$ is accepting, $\hat{\rho}' \in \mathcal{L}(\mathcal{A}_{\rho,\varphi})$. Observe that $|\hat{\rho}| = |\hat{\rho}'|$ and that while $\hat{\sigma}_i = \hat{\sigma}_i'$ ($i = 1, \ldots, m$), the corresponding timestamps $\hat{\tau}_i$ and $\hat{\tau}_i'$ may, in general, differ. In fact, since $\hat{\rho}$ is accepted by $\mathcal{A}_{\rho,\varphi}$, thus by $\mathcal{A}_{\rho}^{+}$, every timed event $(\hat{\sigma}_i, \hat{\tau}_i)$ where $\hat{\sigma}_i$ is neither an add nor a del must be s.t. $(\hat{\sigma}_i, \hat{\tau}_i) = (\sigma_j, \tau_j)$, for some j. This holds analogously for $\hat{\rho}'$. As a consequence, $\hat{\rho}$ and $\hat{\rho}'$ have the same number n of timed events occurring also in ρ, thus, by what discussed above:

$$cost(\rho, \hat{\rho}) = |\hat{\rho}| - n = |\hat{\rho}'| - n = cost(\rho, \hat{\rho}').$$

$\square$

We thus obtain our main result.

Theorem 8. *The following holds for an instance $\mathcal{I} = (\rho, \varphi)$ of timed trace alignment:*

- *checking whether $\mathcal{I}$ admits a solution is decidable;*
- *if $\mathcal{I}$ admits a solution then an optimal solution is computable.*

Proof. By applying Corollary 1 (and Theorem 6) to the augmented trace automaton $\mathcal{A}^+_\varphi$, we can compute the length m of a shortest trace accepted by $\mathcal{A}^+_\varphi$. If $m < 0$ then $\mathcal{I}$ admits no solution.

Otherwise, if $m \geq 0$, we construct the fragment $\hat{\mathcal{W}}_{\rho,\varphi}$ of $\mathcal{W}_{\rho,\varphi}$, up to depth $2|\rho| + m$, with ϵ-transitions not contributing to the depth (we are counting the timed events). By Theorem 3, we have that $\hat{\mathcal{W}}_{\rho,\varphi}$ is computable. Then, we compute the set $\hat{\Gamma}$ of accepting runs ϱ of $\hat{\mathcal{W}}_{\rho,\varphi}$ such that $|\rho| \leq |\varrho| \leq 2|\rho| + m$. By Theorem 2, the runs in $\hat{\Gamma}$ can be rewritten as accepting runs of $\mathcal{T}_{\rho,\varphi}$. Call Γ the resulting set of $\mathcal{T}_{\rho,\varphi}$-runs and let Ψ be the set of traces that induce the runs in Γ. Since Γ contains only accepting runs, all the traces in Ψ are solutions. In addition, by Theorem 5, Ψ contains the optimal solutions. But then, since Ψ is finite and $cost(\rho, \hat{\rho})$ is computable, we can find the minimum-cost trace $\hat{\rho}$ in Ψ, which is the optimal solution. □

5 On Time Complexity

The central result of this work is the computability theorem, i.e., Thm. 8 and, more in general, the whole approach, which constitutes a general theoretical reference for future techniques and concrete implementations. In this section, we analyze the complexity of the proposed technique and briefly discuss some practical implications.

The procedure described in Thm. 8 operates as follows. Firstly, it performs a search on the induced LTS of the augmented constraint automaton $\mathcal{A}^+_\varphi$ to compute the length m of the shortest solution; secondly, it traverses a fragment of the LTS induced by the product automaton $\mathcal{A}_{\rho,\varphi}$ to compute an over-approximation Ψ of the set of optimal solutions; and finally, it extracts the minimal-cost solution among those in Ψ. As explained, the traversals are performed on finite fragments of the two LTSs.

Clearly, MTL$_f$ satisfiability is reducible to our technique, as a formula φ is satisfiable iff an instance $\mathcal{I} = (\rho, \varphi)$ admits a solution, no matter which trace ρ is provided in input. We know that MTL$_f$ satisfiability has non-primitive recursive complexity (Ouaknine and Worrell 2007), meaning that no algorithm exists for solving the problem with a running time expressible as a primitive-recursive function of the input size. Thus, since the class of problems with primitive recursive complexity strictly includes the class of problems with elementary complexity, it follows that our technique has non-elementary complexity.

Obviously, the obtained bound constitutes a major limitation to practical applicability of the approach. Nonetheless, it must be noted that, in the BPM context, constraint specifications are usually obtained as conjunction of simple requirements written according to specific pre-defined templates. This is the case, e.g., of MP-DECLARE (Burattin, Maggi, and Sperduti 2016), whose propositional version, the one of interest in this case, is a proper fragment of MTL$_f$. The use of simple templates might, on the one hand, prevent the "worst-case" from occurring frequently, thus mitigating, itself, the impact on complexity. An example of this effect is offered by MONA (Henriksen et al. 1995): a tool for the translation of Monadic Second-order Logic specifications into DFAs, which, despite adopting a technique with non-elementary complexity, shows practical results that outperform approaches with better theoretical bounds (Zhu et al. 2017). In this respect, it would be interesting to identify, among the most common patters used in the BPM community, those that show best practical performance. Moreover, our work can also be used for isolating further well-behaved classes by exploiting insights on timestamps that are easier or difficult to deal with. On the other hand, specific techniques could be devised for the set of templates of interest only, thus avoiding the complexity due to the generality of the whole MTL$_f$.

6 Conclusions

In this paper, we have shown that checking whether an instance of timed trace alignment –an emerging open problem in Declarative BPM– admits a solution is decidable and that, in the affirmative case, an optimal solution is effectively computable. To do so, we have formalized the problem as the search for an accepting path in an alternating timed automaton and, by taking advantage of and generalizing previous results on the decidability of MTL$_f$, we have devised a constructive technique to actually compute an optimal solution. The resulting technique is the first solution to timed trace alignment, with specifications expressed in MTL$_f$, which is the logic underlying propositional MP-DECLARE, a commonly used language in Declarative BPM.

While, in the general case, demanding from a computational point of view, we can expect that specifications of practical interest are seldom representative of the worst-case, thus mitigating the computational cost in practical cases. In this respect, an interesting future work is that of devising a technique specifically tailored for the propositional fragment of MP-DECLARE and actually testing its implementation, possibly based on the planning technology, on a dataset with log traces stemming from practical contexts. Another, more theoretical, possible extension of this work is that of studying the problem for full MP-DECLARE, i.e., when specifications involve a first-order representation of the states, to express properties concerning event attributes.

Acknowledgements

Work partially supported by the ERC Advanced Grant WhiteMech (No. 834228) and by the EU ICT-48 2020 project TAILOR (No. 952215).

References

Aalst, van der, W. M. 2013. Business process management: a comprehensive survey. *ISRN Software Engineering* 2013.

Burattin, A.; Maggi, F. M.; and Sperduti, A. 2016. Conformance checking based on multi-perspective declarative process models. *Expert Syst. Appl.* 65:194–211.

Chomicki, J. 1995. Efficient checking of temporal integrity constraints using bounded history encoding. *ACM Trans. Database Syst.* 20(2):149–186.

De Giacomo, G., and Vardi, M. Y. 2013. Linear temporal logic and linear dynamic logic on finite traces. In Rossi, F., ed., *IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013*, 854–860. IJCAI/AAAI.

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 Singh, S. P., and Markovitch, S., eds., *AAAI*, 3555–3561. AAAI Press.

de Leoni, M.; Maggi, F. M.; and van der Aalst, W. M. P. 2012. Aligning Event Logs and Declarative Process Models for Conformance Checking. In *10th Int. Conf. on Business Process Management (BPM 2012)*.

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

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

Helmert, M. 2006. The Fast Downward Planning System. *J. Artif. Intell. Res.(JAIR)* 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 Brinksma, E.; Cleaveland, R.; Larsen, K. G.; Margaria, T.; and Steffen, B., eds., *Tools and Algorithms for Construction and Analysis of Systems, First International Workshop, TACAS '95, Aarhus, Denmark, May 19-20, 1995, Proceedings*, volume 1019 of *Lecture Notes in Computer Science*, 89–110. Springer.

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

Lanz, A.; Weber, B.; and Reichert, M. 2014. Time patterns for process-aware information systems. *Requir. Eng.* 19(2):113–141.

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

Torralba, A.; Alcazar, V.; Borrajo, D.; Kissmann, P.; and Edelkamp, S. 2014. Symba: A symbolic bidirectional a planner. In *International Planning Competition*, 105–108.

van der Aalst, W.; Pesic, M.; and Schonenberg, H. 2009. Declarative Workflows: Balancing Between Flexibility and Support. *Computer Science - R&D* 23(2):99–113.

Zhu, S.; Tabajara, L. M.; Li, J.; Pu, G.; and Vardi, M. Y. 2017. Symbolic ltlf synthesis. In Sierra, C., ed., *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, 1362–1369. ijcai.org.