Complexity of Timeline-Based Planning over Dense Temporal Domains: Exploring the Middle Ground

Laura Bozzelli Adriano Peron
University of Napoli "Federico II", Napoli, Italy
lr.bozzelli@gmail.com adrperon@unina.it
Alberto Molinari Angelo Montanari
University of Udine, Udine, Italy
molinari.alberto@gmail.com angelo.montanari@uniud.it

In this paper, we address complexity issues for timeline-based planning over dense temporal domains. The planning problem is modeled by means of a set of independent, but interacting, components, each one represented by a number of state variables, whose behavior over time (timelines) is governed by a set of temporal constraints (synchronization rules). While the temporal domain is usually assumed to be discrete, here we consider the dense case. Dense timeline-based planning has been recently shown to be undecidable in the general case; decidability (**NP**-completeness) can be recovered by restricting to purely existential synchronization rules (*trigger-less rules*). In this paper, we investigate the unexplored area of intermediate cases in between these two extremes. We first show that **decidability** and **non-primitive recursive hardness** can be proved by admitting synchronization rules with a trigger, but forcing them to suitably check constraints only in the future with respect to the trigger (*future simple rules*). More "tractable" results can be obtained by additionally constraining the form of intervals in future simple rules: **EXPSPACE**-completeness is guaranteed by avoiding singular intervals, **PSPACE**-completeness by admitting only intervals of the forms $[0,a]$ and $[b,+\infty[$.

1 Introduction

In this paper, we explore the middle ground of timeline-based planning over dense temporal domains. *Timeline-based planning* can be viewed as an alternative to the classical action-based approach to planning. Action-based planning aims at determining a sequence of actions that, given the initial state of the world and a goal, transforms, step by step, the state of the world until a state that satisfies the goal is reached. Timeline-based planning focuses on what has to happen in order to satisfy the goal instead of what an agent has to do. It models the planning domain as a set of independent, but interacting, components, each one consisting of a number of state variables. The evolution of the values of state variables over time is described by means of a set of timelines (sequences of tokens), and it is governed by a set of transition functions, one for each state variable, and a set of synchronization rules, that constrain the temporal relations among state variables. Figure 1 gives an account of these notions.

Timeline-based planning has been successfully exploited in a number of application domains, e.g., [5, 9, 10, 13, 17, 19], but a systematic study of its expressiveness and complexity has been undertaken only very recently. The temporal domain is commonly assumed to be discrete, the dense case being dealt with by forcing an artificial discretization of the domain. In [14], Gigante et al. showed that timeline-based planning with bounded temporal relations and token durations, and no temporal horizon, is **EXPSPACE**-complete and expressive enough to capture action-based temporal planning. Later, in [15], they proved that **EXPSPACE**-completeness still holds for timeline-based planning with unbounded interval relations, and that the problem becomes **NEXPTIME**-complete if an upper bound to the temporal horizon is added.

A. Orlandini, M. Zimmermann (Eds.): 9th Symposium on Games, Automata, Logics and Formal Verification (GandALF'18)
EPTCS 277, 2018, pp. 191–205, doi:10.4204/EPTCS.277.14

Timeline-based planning (TP for short) over a dense temporal domain has been studied in [7]. Having recourse to a dense time domain is important for expressiveness: only in such a domain one can really express interval-based properties of planning domains, and can abstract from unnecessary (or even "forced") details which are often artificially added due to the necessity of discretizing time. The general TP problem has been shown to be **undecidable** even when a single state variable is used. Decidability can be recovered by suitably constraining the logical structure of synchronization rules. In the general case, a synchronization rule allows a universal quantification over the tokens of a timeline (*trigger*). By disallowing the universal quantification and retaining only rules in purely existential form (*trigger-less rules*), the problem becomes **NP**-complete [8]. These two bounds identify a large unexplored area of intermediate cases where to search for a balance between expressiveness and complexity. Investigating such cases is fundamental: as a matter of fact, trigger-less rules can essentially be used only to express initial conditions and the goals of the problem, while trigger rules, much more powerful, are useful to specify invariants and response requirements. Thus one needs somehow a way of re-introducing the latter rules in order to recover their expressive power at least partially.

In this paper, we investigate the restrictions under which the universal quantification of triggers can be admitted though retaining decidability. When a token is "selected" by a trigger, the synchronization rule allows us to compare tokens of the timelines both preceeding (past) and following (future) the trigger token. The first restriction we consider consists in limiting the comparison to tokens in the future with respect to the trigger (*future semantics of trigger rules*). The second restriction we consider imposes that, in a trigger rule, the name of a non-trigger token appears exactly once in the *interval atoms* of the rule (*simple trigger rules*). This syntactical restriction avoids comparisons of multiple token time-events with a non-trigger reference time-event. From the expressiveness viewpoint, even if we do not have a formal statement, we conjecture that future simple trigger rules, together with arbitrary trigger-less rules allow for expressiveness strictly in between MTL [3] and TPTL [4]. Note that, by [7], the TP problem with *simple* trigger rules is already undecidable. In this paper, we show that it becomes decidable, although non-primitive recursive hard, under the *future semantics* of the trigger rules. Better complexity results can be obtained by restricting also the type of intervals used in the simple trigger rules to compare tokens. In particular, we show that future TP with simple trigger rules without singular intervals[1] is **EXPSPACE**-complete. The problem is instead **PSPACE**-complete if we consider only intervals of the forms $[0,a]$ and $[b,+\infty[$. The decidability status of the TP problem with arbitrary trigger rules under the future semantics remains open. However, we show that such a problem is at least non-primitive recursive even under the assumption that the intervals in the rules have the forms $[0,a]$ and $[b,+\infty[$.

Organization of the paper. In Section 2, we recall the TP framework. In Section 3, we establish that future TP with simple trigger rules is decidable, and show membership in **EXPSPACE** (resp., **PSPACE**) under the restriction to non-singular intervals (resp., intervals of the forms $[0,a]$ and $[b,+\infty[$). Matching lower bounds for the last two problems are given in Section 5. In Section 4, we prove non-primitive recursive hardness of TP under the future semantics of trigger rules. Conclusions give an assessment of the work and outline future research themes. All missing proofs and results can be found in [6].

2 Preliminaries

Let $\mathbb{N}$ be the set of natural numbers, $\mathbb{R}_+$ be the set of non-negative real numbers, and *Intv* be the set of intervals in $\mathbb{R}_+$ whose endpoints are in $\mathbb{N}\cup\{\infty\}$. Moreover, let us denote by $Intv_{(0,\infty)}$ the set of intervals

[1] An interval is called *singular* if it has the form $[a,a]$, for $a\in\mathbb{N}$.

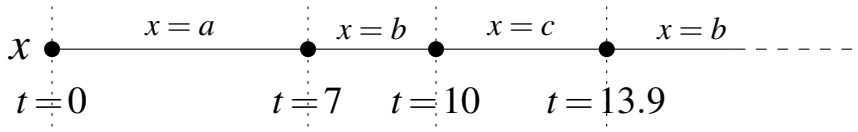

Figure 1: An example of timeline $(a,7)(b,3)(c,3.9)\cdots$ for the state variable $x = (V_x, T_x, D_x)$, where $V_x = \{a,b,c,\ldots\}$, $b \in T_x(a)$, $c \in T_x(b)$, $b \in T_x(c)\ldots$ and $D_x(a) = [5,8]$, $D_x(b) = [1,4]$, $D_x(c) = [2,\infty[\ldots$

$I \in \mathit{Intv}$ such that either I is unbounded, or I is left-closed with left endpoint 0. Such intervals I can be replaced by expressions of the form $\sim n$ for some $n \in \mathbb{N}$ and $\sim \in \{<, \leq, >, \geq\}$. Let w be a finite word over some alphabet. By $|w|$ we denote the length of w. For all $0 \leq i < |w|$, $w(i)$ is the i-th letter of w.

2.1 The TP Problem

In this section, we recall the TP framework as presented in [11, 14]. In TP, domain knowledge is encoded by a set of state variables, whose behaviour over time is described by transition functions and synchronization rules.

Definition 1. A *state variable* x is a triple $x = (V_x, T_x, D_x)$, where V_x is the *finite domain* of the variable x, $T_x : V_x \to 2^{V_x}$ is the *value transition function*, which maps each $v \in V_x$ to the (possibly empty) set of successor values, and $D_x : V_x \to \mathit{Intv}$ is the *constraint function* that maps each $v \in V_x$ to an interval.

A *token* for a variable x is a pair (v,d) consisting of a value $v \in V_x$ and a duration $d \in \mathbb{R}_+$ such that $d \in D_x(v)$. Intuitively, a token for x represents an interval of time where the state variable x takes value v. The behavior of the state variable x is specified by means of *timelines* which are non-empty sequences of tokens $\pi = (v_0,d_0)\ldots(v_n,d_n)$ consistent with the value transition function T_x, that is, such that $v_{i+1} \in T_x(v_i)$ for all $0 \leq i < n$. The *start time* $\mathsf{s}(\pi,i)$ and the *end time* $\mathsf{e}(\pi,i)$ of the i-th token ($0 \leq i \leq n$) of the timeline π are defined as follows: $\mathsf{e}(\pi,i) = \sum_{h=0}^{i} d_h$ and $\mathsf{s}(\pi,i) = 0$ if $i = 0$, and $\mathsf{s}(\pi,i) = \sum_{h=0}^{i-1} d_h$ otherwise. See Figure 1 for an example.

Given a finite set SV of state variables, a *multi-timeline* of SV is a mapping Π assigning to each state variable $x \in SV$ a timeline for x. Multi-timelines of SV can be constrained by a set of *synchronization rules*, which relate tokens, possibly belonging to different timelines, through temporal constraints on the start/end-times of tokens (time-point constraints) and on the difference between start/end-times of tokens (interval constraints). The synchronization rules exploit an alphabet Σ of token names to refer to the tokens along a multi-timeline, and are based on the notions of *atom* and *existential statement*.

Definition 2. An *atom* is either a clause of the form $o_1 \leq_I^{e_1,e_2} o_2$ (*interval atom*), or of the forms $o_1 \leq_I^{e_1} n$ or $n \leq_I^{e_1} o_1$ (*time-point atom*), where $o_1, o_2 \in \Sigma$, $I \in \mathit{Intv}$, $n \in \mathbb{N}$, and $e_1, e_2 \in \{\mathsf{s},\mathsf{e}\}$.

An atom ρ is evaluated with respect to a Σ-*assignment* λ_Π *for a given multi-timeline* Π which is a mapping assigning to each token name $o \in \Sigma$ a pair $\lambda_\Pi(o) = (\pi,i)$ such that π is a timeline of Π and $0 \leq i < |\pi|$ is a position along π (intuitively, (π,i) represents the token of Π referenced by the name o). An interval atom $o_1 \leq_I^{e_1,e_2} o_2$ *is satisfied by* λ_Π if $e_2(\lambda_\Pi(o_2)) - e_1(\lambda_\Pi(o_1)) \in I$. A point atom $o \leq_I^{e} n$ (resp., $n \leq_I^{e} o$) *is satisfied by* λ_Π if $n - e(\lambda_\Pi(o)) \in I$ (resp., $e(\lambda_\Pi(o)) - n \in I$).

Definition 3. An *existential statement* $\mathscr{E}$ for a finite set SV of state variables is a statement of the form:

$$\mathscr{E} := \exists o_1[x_1 = v_1]\cdots\exists o_n[x_n = v_n].\mathscr{C}$$

where $\mathscr{C}$ is a conjunction of atoms, $o_i \in \Sigma$, $x_i \in SV$, and $v_i \in V_{x_i}$ for each $i = 1,\ldots,n$. The elements $o_i[x_i = v_i]$ are called *quantifiers*. A token name used in $\mathscr{C}$, but not occurring in any quantifier, is said to be *free*. Given a Σ-assignment λ_Π for a multi-timeline Π of SV, we say that λ_Π *is consistent with the existential statement* $\mathscr{E}$ if for each quantified token name o_i, $\lambda_\Pi(o_i) = (\pi,h)$ where $\pi = \Pi(x_i)$ and the h-th token of π has value v_i. A multi-timeline Π of SV *satisfies* $\mathscr{E}$ if there exists a Σ-assignment λ_Π for Π consistent with $\mathscr{E}$ such that each atom in $\mathscr{C}$ is satisfied by λ_Π.

Definition 4. A *synchronization rule* $\mathscr{R}$ for a finite set SV of state variables is a rule of one of the forms

$$o_0[x_0 = v_0] \rightarrow \mathscr{E}_1 \vee \mathscr{E}_2 \vee \ldots \vee \mathscr{E}_k, \quad \top \rightarrow \mathscr{E}_1 \vee \mathscr{E}_2 \vee \ldots \vee \mathscr{E}_k,$$

where $o_0 \in \Sigma$, $x_0 \in SV$, $v_0 \in V_{x_0}$, and $\mathscr{E}_1,\ldots,\mathscr{E}_k$ are *existential statements*. In rules of the first form (*trigger rules*), the quantifier $o_0[x_0 = v_0]$ is called *trigger*, and we require that only o_0 may appear free in $\mathscr{E}_i$ (for $i = 1,\ldots,n$). In rules of the second form (*trigger-less rules*), we require that no token name appears free. A trigger rule $\mathscr{R}$ is *simple* if for each existential statement $\mathscr{E}$ of $\mathscr{R}$ and each token name o distinct from the trigger, there is at most one *interval atom* of $\mathscr{E}$ where o occurs.

Intuitively, a trigger $o_0[x_0 = v_0]$ acts as a universal quantifier, which states that *for all* the tokens of the timeline for the state variable x_0, where the variable x_0 takes the value v_0, at least one of the existential statements $\mathscr{E}_i$ must be true. Trigger-less rules simply assert the satisfaction of some existential statement. The intuitive meaning of the *simple* trigger rules is that they disallow simultaneous comparisons of multiple time-events (start/end times of tokens) with a non-trigger reference time-event. The semantics of synchronization rules is formally defined as follows.

Definition 5. Let Π be a multi-timeline of a set SV of state variables. Given a *trigger-less rule* $\mathscr{R}$ of SV, Π *satisfies* $\mathscr{R}$ if Π satisfies some existential statement of $\mathscr{R}$. Given a *trigger rule* $\mathscr{R}$ of SV with trigger $o_0[x_0 = v_0]$, Π *satisfies* $\mathscr{R}$ if for every position i of the timeline $\Pi(x_0)$ for x_0 such that $\Pi(x_0) = (v_0,d)$, there is an existential statement $\mathscr{E}$ of $\mathscr{R}$ and a Σ-assignment λ_Π for Π which is consistent with $\mathscr{E}$ such that $\lambda_\Pi(o_0) = (\Pi(x_0),i)$ and λ_Π satisfies all the atoms of $\mathscr{E}$.

In the paper, we focus on a stronger notion of satisfaction of trigger rules, called *satisfaction under the future semantics*. It requires that all the non-trigger selected tokens do not start *strictly before* the start-time of the trigger token.

Definition 6. A multi-timeline Π of SV *satisfies under the future semantics* a trigger rule $\mathscr{R} = o_0[x_0 = v_0] \rightarrow \mathscr{E}_1 \vee \mathscr{E}_2 \vee \ldots \vee \mathscr{E}_k$ if Π satisfies the trigger rule obtained from $\mathscr{R}$ by replacing each existential statement $\mathscr{E}_i = \exists o_1[x_1 = v_1]\cdots\exists o_n[x_n = v_n].\mathscr{C}$ with $\exists o_1[x_1 = v_1]\cdots\exists o_n[x_n = v_n].\mathscr{C} \wedge \bigwedge_{i=1}^{n} o_0 \leq^{\mathsf{s},\mathsf{s}}_{[0,+\infty[} o_i$.

A TP domain $P = (SV,R)$ is specified by a finite set SV of state variables and a finite set R of synchronization rules modeling their admissible behaviors. A *plan of* P is a multi-timeline of SV satisfying all the rules in R. A *future plan of* P is defined in a similar way, but we require that the fulfillment of the trigger rules is under the future semantics. We are interested in the following decision problems: (i) *TP problem:* given a TP domain $P = (SV,R)$, is there a plan for P? (ii) *Future TP problem:* similar to the previous one, but we require the existence of a future plan.

Table 1 summarizes all the decidability and complexity results described in the following. We consider mixes of restrictions of the TP problem involving trigger rules with future semantics, simple trigger rules, and intervals in atoms of trigger rules which are non-singular or in $Intv_{(0,\infty)}$.

3 Solving the future TP problem with simple trigger rules

Recently, we have shown that the TP problem is undecidable even if the trigger rules are assumed to be simple [7]. In this section, we show that decidability can be recovered assuming that the trigger rules

	TP problem	Future TP problem
Unrestricted	Undecidable	(Decidable?) Non-primitive recursive-hard
Simple trigger rules	Undecidable	Decidable (non-primitive recursive)
Simple trigger rules, non-singular intervals	?	**EXPSPACE**-complete
Simple trigger rules, intervals in $Intv_{(0,\infty)}$	?	**PSPACE**-complete
Trigger-less rules only	**NP**-complete	//

Table 1: Decidability and complexity of restrictions of the TP problem.

are *simple* and *interpreted under the future semantics*. Moreover, we establish that under the additional assumption that intervals in trigger rules are non-singular (resp., are in $Intv_{(0,\infty)}$), the problem is in **EXPSPACE** (resp., **PSPACE**). The decidability status of future TP with arbitrary trigger rules remains open. In Section 4, we prove that the latter problem is at least non-primitive recursive even if intervals in rules and in the constraint functions of the state variables are assumed to be in $Intv_{(0,\infty)}$.

The rest of this section is organized as follows: in Subsection 3.1, we recall the framework of Timed Automata (TA) [1] and Metric Temporal logic (MTL) [18]. In Subsection 3.2, we reduce the future TP problem with simple trigger rules to the existential model-checking problem for TA against MTL over *finite timed words*. The latter problem is known to be decidable [20].

3.1 Timed automata and the logic MTL

Let us recall the notion of timed automaton (TA) [1] and the logic MTL [18]. Let Σ be a finite alphabet. A *timed word* w over Σ is a *finite* word $w = (a_0,\tau_0)\cdots(a_n,\tau_n)$ over $\Sigma\times\mathbb{R}_+$ (τ_i is the time at which a_i occurs) such that $\tau_i \leq \tau_{i+1}$ for all $0 \leq i < n$ (monotonicity). The timed word w is also denoted by (σ,τ), where σ is the finite untimed word $a_0\cdots a_n$ and τ is the sequence of timestamps $\tau_0\cdots\tau_n$. A *timed language* over Σ is a set of timed words over Σ.

Timed Automata (TA). Let C be a finite set of clocks. A clock valuation $val : C \to \mathbb{R}_+$ for C is a mapping assigning a non-negative real value to each clock in C. For $t \in \mathbb{R}_+$ and a reset set $Res \subseteq C$, $(val+t)$ and $val[Res]$ denote the valuations over C defined as follows: for all $c \in C$, $(val+t)(c) = val(c)+t$, and $val[Res](c) = 0$ if $c \in Res$ and $val[Res](c) = val(c)$ otherwise. A *clock constraint* over C is a conjunction of atomic formulas of the form $c \in I$ or $c-c' \in I$, where $c,c' \in C$ and $I \in Intv$. For a clock valuation val and a clock constraint θ, val satisfies θ, written $val \models \theta$, if, for each conjunct $c \in I$ (resp., $c-c' \in I$) of θ, $val(c) \in I$ (resp., $val(c)-val(c') \in I$). We denote by $\Phi(C)$ the set of clock constraints over C.

Definition 7. A TA over Σ is a tuple $\mathscr{A} = (\Sigma, Q, q_0, C, \Delta, F)$, where Q is a finite set of (control) states, $q_0 \in Q$ is the initial state, C is the finite set of clocks, $F \subseteq Q$ is the set of accepting states, and $\Delta \subseteq Q \times \Sigma \times \Phi(C) \times 2^C \times Q$ is the transition relation. The *maximal constant of* $\mathscr{A}$ is the greatest integer occurring as endpoint of some interval in the clock constraints of $\mathscr{A}$.

Intuitively, in a TA $\mathscr{A}$, while transitions are instantaneous, time can elapse in a control state. The clocks progress at the same speed and can be reset independently of each other when a transition is executed, in such a way that each clock keeps track of the time elapsed since the last reset. Moreover, clock constraints are used as guards of transitions to restrict the behavior of the automaton.

Formally, a configuration of $\mathscr{A}$ is a pair (q,val), where $q \in Q$ and val is a clock valuation for C. A run r of $\mathscr{A}$ on a timed word $w = (a_0,\tau_0)\cdots(a_n,\tau_n)$ over Σ is a sequence of configurations $r = (q_0,val_0)\cdots(q_{n+1},val_{n+1})$ starting at the initial configuration (q_0,val_0), with $val_0(c) = 0$ for all $c \in C$ and

- for all $0 \leq i \leq n$ we have (we let $\tau_{-1} = 0$): $(q_i, a_i, \theta, Res, q_{i+1}) \in \Delta$ for some $\theta \in \Phi(C)$ and reset set Res, $(val_i + \tau_i - \tau_{i-1}) \models \theta$ and $val_{i+1} = (val_i + \tau_i - \tau_{i-1})[Res]$.

The run r is *accepting* if $q_{n+1} \in F$. The *timed language* $\mathscr{L}_T(\mathscr{A})$ of $\mathscr{A}$ is the set of timed words w over Σ such that there is an accepting run of $\mathscr{A}$ over w.

The logic MTL. We now recall the framework of Metric Temporal Logic (MTL) [18], a well-known timed linear-time temporal logic which extends standard LTL with time constraints on until modalities.

For a finite set $\mathscr{P}$ of atomic propositions, the set of MTL formulas φ over $\mathscr{P}$ is defined as follows:

$$\varphi ::= \top \mid p \mid \varphi \vee \varphi \mid \neg\varphi \mid \varphi \mathsf{U}_I \varphi$$

where $p \in \mathscr{P}$, $I \in Intv$, and U_I is the standard *strict timed until* MTL modality. MTL formulas over $\mathscr{P}$ are interpreted over timed words over $2^{\mathscr{P}}$. Given an MTL formula φ, a timed word $w = (\sigma, \tau)$ over $2^{\mathscr{P}}$, and a position $0 \leq i < |w|$, the satisfaction relation $(w,i) \models \varphi$, meaning that φ holds at position i of w, is defined as follows (we omit the clauses for atomic propositions and Boolean connectives):

- $(w,i) \models \varphi_1 \mathsf{U}_I \varphi_2 \Leftrightarrow$ there is $j > i$ such that $(w,j) \models \varphi_2, \tau_j - \tau_i \in I,$ and $(w,k) \models \varphi_1$ for all $i < k < j$.

A *model of* φ is a timed word w over $2^{\mathscr{P}}$ such that $(w,0) \models \varphi$. The *timed language* $\mathscr{L}_T(\varphi)$ of φ is the set of models of φ. The *existential model-checking problem for TA against MTL* is the problem of checking for a TA $\mathscr{A}$ over $2^{\mathscr{P}}$ and an MTL formula φ over $\mathscr{P}$ whether $\mathscr{L}_T(\mathscr{A}) \cap \mathscr{L}_T(\varphi) \neq \emptyset$.

In MTL, we use standard shortcuts: $\mathsf{F}_I\varphi$ stands for $\varphi \vee (\top \mathsf{U}_I \varphi)$ (*timed eventually*), and $\mathsf{G}_I\varphi$ stands for $\neg\mathsf{F}_I\neg\varphi$ (*timed always*). We also consider two fragments of MTL, namely, MITL (Metric Interval Temporal Logic) and $\mathsf{MITL}_{(0,\infty)}$ [2]: MITL is obtained by allowing only non-singular intervals in *Intv*, while $\mathsf{MITL}_{(0,\infty)}$ is the fragment of MITL obtained by allowing only intervals in $Intv_{(0,\infty)}$. The *maximal constant* of an MTL formula φ is the greatest integer occurring as an endpoint of some interval of (the occurrences of) the U_I modality in φ.

3.2 Reduction to existential model checking of TA against MTL

In this section, we solve the future TP problem with simple trigger rules by an exponential-time reduction to the existential model-checking problem for TA against MTL.

In the following, we fix an instance $P = (SV, R)$ of the problem such that the trigger rules in R are simple. The *maximal constant* of P, denoted by K_P, is the greatest integer occurring in the atoms of R and in the constraint functions of the variables in SV.

The proposed reduction consists of three steps: (i) first, we define an encoding of the multi-timelines of SV by means of timed words over $2^{\mathscr{P}}$ for a suitable finite set $\mathscr{P}$ of atomic propositions, and show how to construct a TA $\mathscr{A}_{SV}$ over $2^{\mathscr{P}}$ accepting such encodings; (ii) next, we build an MTL formula $\varphi_\forall$ over $\mathscr{P}$ such that for each multi-timeline Π of SV and encoding w_Π of Π, w_Π is a model of $\varphi_\forall$ if and only if Π satisfies all the trigger rules in R under the future semantics; (iii) finally, we construct a TA $\mathscr{A}_\exists$ over $2^{\mathscr{P}}$ such that for each multi-timeline Π of SV and encoding w_Π of Π, w_Π is accepted by $\mathscr{A}_\exists$ if and only if Π satisfies all the trigger-less rules in R. Hence, there is a future plan of (SV, R) if and only if $\mathscr{L}_T(\mathscr{A}_{SV}) \cap \mathscr{L}_T(\mathscr{A}_\exists) \cap \mathscr{L}_T(\varphi_\forall) \neq \emptyset$.

For each $x \in SV$, let $x = (V_x, T_x, D_x)$. Given an interval $I \in Intv$ and a natural number $n \in \mathbb{N}$, $n + I$ (resp., $n - I$) denotes the set of non-negative real numbers $\tau \in \mathbb{R}_+$ such that $\tau - n \in I$ (resp., $n - \tau \in I$). Note that $n + I$ (resp., $n - I$) is a (possibly empty) interval in *Intv* whose endpoints can be trivially calculated. For an atom ρ in R involving a time constant (*time-point atom*), let $I(\rho)$ be the interval in *Intv* defined as follows:

- if ρ is of the form $o \leq_I^e n$ (resp., $n \leq_I^e o$), then $I(\rho) := n - I$ (resp., $I(\rho) = n + I$).

We define $Intv_R := \{J \in Intv \mid J = I(\rho)$ for some time-point atom ρ occurring in a trigger rule of $R\}$.

Encodings of multi-timelines of SV. We assume that for distinct state variables x and x', the sets V_x and $V_{x'}$ are disjunct. We exploit the following set $\mathscr{P}$ of propositions to encode multi-timelines of SV:

$$\mathscr{P} := \bigcup_{x \in SV} Main_x \cup Deriv,$$

$$Main_x := (\{beg_x\} \cup V_x) \times V_x \cup V_x \times \{end_x\}, \qquad Deriv := Intv_R \cup \{p_>\} \cup \bigcup_{x \in SV} \bigcup_{v \in V_x} \{past_v^{\mathsf{s}}, past_v^{\mathsf{e}}\}.$$

Intuitively, we use the propositions in $Main_x$ to encode a token along a timeline for x. The propositions in $Deriv$, as explained below, represent enrichments of the encoding, used for translating simple trigger rules in MTL formulas under the future semantics. The tags beg_x and end_x in $Main_x$ are used to mark the start and the end of a timeline for x. In particular, a token tk with value v along a timeline for x is encoded by two events: the *start-event* (occurring at the start time of tk) and the *end-event* (occurring at the end time of tk). The start-event of tk is specified by a main proposition of the form (v_p, v), where either $v_p = beg_x$ (tk is the first token of the timeline) or v_p is the value of the x-token preceding tk. The end-event of tk is instead specified by a main proposition of the form (v, v_s), where either $v_s = end_x$ (tk is the last token of the timeline) or v_s is the value of the x-token following tk. Now, we explain the meaning of the propositions in $Deriv$. Elements in $Intv_R$ reflect the semantics of the time-point atoms in the trigger rules of R: for each $I \in Intv_R$, I holds at the current position if the current timestamp τ satisfies $\tau \in I$. The tag $p_>$ keeps track of whether the current timestamp is strictly greater than the previous one. Finally, the propositions in $\bigcup_{x \in SV} \bigcup_{v \in V_x} \{past_v^{\mathsf{s}}, past_v^{\mathsf{e}}\}$ keep track of past token events occurring at timestamps *coinciding* with the current timestamp. We first define the encoding of timelines for $x \in SV$.

A *code for a timeline for x* is a timed word w over $2^{Main_x \cup Deriv}$ of the form

$$w = (\{(beg_x, v_0)\} \cup S_0, \tau_0), (\{(v_0, v_1)\} \cup S_1, \tau_1) \ldots (\{(v_n, end_x)\} \cup S_{n+1}, \tau_{n+1})$$

where, for all $0 \leq i \leq n+1$, $S_i \subseteq Deriv$, and (i) $v_{i+1} \in T_x(v_i)$ if $i < n$; (ii) $\tau_0 = 0$ and $\tau_{i+1} - \tau_i \in D_x(v_i)$ if $i \leq n$; (iii) $S_i \cap Intv_R$ is the set of intervals $I \in Intv_R$ such that $\tau_i \in I$, and $p_> \in S_i$ iff either $i = 0$ or $\tau_i > \tau_{i-1}$; (iv) for all $v \in V_x$, $past_v^{\mathsf{s}} \in S_i$ (resp., $past_v^{\mathsf{e}} \in S_i$) iff there is $0 \leq h < i$ such that $\tau_h = \tau_i$ and $v = v_h$ (resp., $\tau_h = \tau_i$, $v = v_{h-1}$ and $h > 0$). Note that the length of w is at least 2. The timed word w encodes the timeline for x of length $n+1$ given by $\pi = (v_0, \tau_1)(v_1, \tau_2 - \tau_1) \cdots (v_n, \tau_{n+1} - \tau_n)$. Note that in the encoding, τ_i and τ_{i+1} represent the start time and the end time of the i-th token of the timeline ($0 \leq i \leq n$).

Next, we define the encoding of a multi-timeline for SV. For a set $P \subseteq \mathscr{P}$ and $x \in SV$, let $P[x] := P \setminus \bigcup_{y \in SV \setminus \{x\}} Main_y$. A *code for a multi-timeline for SV* is a timed word w over $2^{\mathscr{P}}$ of the form $w = (P_0, \tau_0) \cdots (P_n, \tau_n)$ such that the following conditions hold: (i) for all $x \in SV$, the timed word obtained from $(P_0[x], \tau_0) \cdots (P_n[x], \tau_n)$ by removing the pairs $(P_i[x], \tau_i)$ such that $P_i[x] \cap Main_x = \emptyset$ is a code of a timeline for x; (ii) $P_0[x] \cap Main_x \neq \emptyset$ for all $x \in SV$ (initialization).

One can easily construct a TA $\mathscr{A}_{SV}$ over $2^{\mathscr{P}}$ accepting the encodings of the multi-timelines of SV. In particular, the TA $\mathscr{A}_{SV}$ uses a clock c_x for each state variable x for checking the time constraints on the duration of the tokens for x. Two additional clocks $c_>$ and c_{glob} are exploited for capturing the meaning of the proposition $p_>$ and of the propositions in $Intv_R$ (in particular, c_{glob} is a clock which measures the current time and is never reset). Hence, we obtain the following result (for details, see [6]).

Proposition 8. *One can construct a TA $\mathscr{A}_{SV}$ over $2^{\mathscr{P}}$, with $2^{O(\Sigma_{x \in SV} |V_x|)}$ states, $|SV| + 2$ clocks, and maximal constant $O(K_P)$, such that $\mathscr{L}_T(\mathscr{A}_{SV})$ is the set of codes for the multi-timelines of SV.*

Encodings of simple trigger rules by MTL formulas. We now construct an MTL formula $\varphi_\forall$ over $\mathscr{P}$ capturing the trigger rules in R, which, by hypothesis, are simple, under the future semantics.

Proposition 9. *If the trigger rules in R are simple, then one can construct in linear-time an MTL formula $\varphi_\forall$, with maximal constant $O(K_P)$, such that for each multi-timeline Π of SV and encoding w_Π of Π, w_Π is a model of $\varphi_\forall$ iff Π satisfies all the trigger rules in R under the future semantics. Moreover, $\varphi_\forall$ is an MITL formula (resp., $\mathsf{MITL}_{(0,\infty)}$ formula) if the intervals in the trigger rules are non-singular (resp., belong to $Intv_{(0,\infty)}$). Finally, $\varphi_\forall$ has $O(\sum_{x\in SV}|V_x|+N_a)$ distinct subformulas, where N_a is the overall number of atoms in the trigger rules in R.*

Proof. We first introduce some auxiliary propositional (Boolean) formulas over $\mathscr{P}$. Let $x \in SV$ and $v \in V_x$. We denote by $\psi(\mathsf{s},v)$ and $\psi(\mathsf{e},v)$ the two propositional formulas over $Main_x$ defined as follows:

$$\psi(\mathsf{s},v) := (beg_x,v) \vee \bigvee_{u\in V_x}(u,v) \qquad \psi(\mathsf{e},v) := (v,end_x) \vee \bigvee_{u\in V_x}(v,u)$$

Intuitively, $\psi(\mathsf{s},v)$ (resp., $\psi(\mathsf{e},v)$) states that a start-event (resp., end-event) for a token for x with value v occurs at the current time. We also exploit the formula $\psi_{\neg x} := \neg\bigvee_{m\in Main_x} m$ asserting that no event for a token for x occurs at the current time. Additionally, for an MTL formula θ, we exploit the MTL formula $EqTime(\theta) := \theta \vee [\neg p_> \mathsf{U}_{\geq 0}(\neg p_> \wedge \theta)]$ which is satisfied by a code of a multi-timeline of SV at the current time, if θ eventually holds at a position whose timestamp coincides with the current timestamp.

The MTL formula $\varphi_\forall$ has a conjunct $\varphi_{\mathscr{R}}$ for each trigger rule $\mathscr{R}$. Let $\mathscr{R}$ be a trigger rule of the form $o_t[x_t = v_t] \rightarrow \mathscr{E}_1 \vee \mathscr{E}_2 \vee \ldots \vee \mathscr{E}_k$. Then, the MTL formula $\varphi_{\mathscr{R}}$ is given by

$$\varphi_{\mathscr{R}} := \mathsf{G}_{\geq 0}\big(\psi(\mathsf{s},v_t) \rightarrow \bigvee_{i=1}^{k}\Phi_{\mathscr{E}_i}\big)$$

where the MTL formula $\Phi_{\mathscr{E}_i}$, with $1 \leq i \leq k$, ensures the fulfillment of the existential statement $\mathscr{E}_i$ of the trigger rule $\mathscr{R}$ under the future semantics. Let $\mathscr{E} \in \{\mathscr{E}_1,\ldots,\mathscr{E}_k\}$, O be the set of token names existentially quantified by $\mathscr{E}$, $\mathbf{A}$ be the set of *interval* atoms in $\mathscr{E}$, and, for each $o \in O$, $v(o)$ be the value of the token referenced by o in the associated quantifier. In the construction of $\Phi_{\mathscr{E}}$, we crucially exploit the assumption that $\mathscr{R}$ is simple: for each token name $o \in O$, there is at most one atom in $\mathbf{A}$ where o occurs.

For each token name $o \in \{o_t\} \cup O$, we denote by $Intv^{\mathsf{s}}_o$ (resp., $Intv^{\mathsf{e}}_o$) the set of intervals $J \in Intv$ such that $J = I(\rho)$ for some time-point atom ρ occurring in $\mathscr{E}$, which imposes a time constraint on the start time (resp., end time) of the token referenced by o. Note that $Intv^{\mathsf{s}}_o, Intv^{\mathsf{e}}_o \subseteq \mathscr{P}$, and we exploit the propositional formulas $\xi^{\mathsf{s}}_o = \bigwedge_{I\in Intv^{\mathsf{s}}_o} I$ and $\xi^{\mathsf{e}}_o = \bigwedge_{I\in Intv^{\mathsf{e}}_o} I$ to ensure the fulfillment of the time constraints imposed by the time-point atoms associated with the token o. The MTL formula $\Phi_{\mathscr{E}}$ is given by:

$$\Phi_{\mathscr{E}} := \xi^{\mathsf{s}}_{o_t} \wedge [\psi_{\neg x_t}\mathsf{U}_{\geq 0}(\psi(\mathsf{e},v_t) \wedge \xi^{\mathsf{e}}_{o_t})] \wedge \bigwedge_{\rho\in\mathbf{A}} \chi_\rho,$$

where, for each atom $\rho \in \mathbf{A}$, the formula χ_ρ captures the future semantics of ρ.

The construction of χ_ρ depends on the form of ρ. We distinguishes four cases.

- $\rho = o \leq_I^{e_1,e_2} o_t$ and $o \neq o_t$. We assume $0 \in I$ (the other case being simpler). First, assume that $e_2 = \mathsf{s}$. Under the future semantics, ρ holds iff the start time of the trigger token o_t coincides with the e_1-time of token o. Hence, in this case ($e_2 = \mathsf{s}$), χ_ρ is given by:

$$\chi_\rho := \xi^{e_1}_o \wedge \big(past^{e_1}_{v(o)} \vee EqTime(\psi(e_1,v(0)))\big).$$

If instead $e_2 = \mathsf{e}$, then χ_ρ is defined as follows:

$$\chi_\rho := [\psi_{\neg x_t} \mathsf{U}_{\geq 0}\{\xi_o^{e_1} \wedge \psi(e_1, v(0)) \wedge \psi_{\neg x_t} \wedge (\psi_{\neg x_t} \mathsf{U}_I \psi(\mathsf{e}, v_t))\}] \vee [(\psi(e_1, v(0)) \vee past^{e_1}_{v(o)}) \wedge \xi_o^{e_1}] \vee [\psi_{\neg x_t} \mathsf{U}_{\geq 0}\{\psi(\mathsf{e}, v_t) \wedge \mathit{EqTime}(\psi(e_1, v(0)) \wedge \xi_o^{e_1})\}]$$

The first disjunct considers the case where the e_1-event of token o occurs strictly between the start-event and the end-event of the trigger token o_t (along the encoding of a multi-timeline of SV). The second considers the case where the e_1-event of token o precedes the start-event of the trigger token: thus, under the future semantics, it holds that the e_1-time of token o coincides with the start time of the trigger token. Finally, the third disjunct considers the case where the e_1-event of token o follows the end-event of the trigger token (in this case, the related timestamps have to coincide).

- $\rho = o_t \leq_I^{e_1,e_2} o$ and $o \neq o_t$. We assume $e_1 = \mathsf{e}$ and $0 \in I$ (the other cases being simpler). Then,

$$\chi_\rho = [\psi_{\neg x_t} \mathsf{U}_{\geq 0}(\psi(\mathsf{e}, u_t) \wedge \mathsf{F}_I(\psi(e_2, v(o)) \wedge \xi_o^{e_2}))] \vee [\psi_{\neg x_t} \mathsf{U}_{\geq 0}(\psi(\mathsf{e}, u_t) \wedge past^{e_2}_{v(o)} \wedge \xi_o^{e_2})],$$

  where the second disjunct captures the situation where the e_2-time of o coincides with the end time of the trigger token o_t, but the e_2-event of o occurs before the end-event of the trigger token.
- $\rho = o_t \leq_I^{e_1,e_2} o_t$. This case is straightforward and we omit the details.
- $\rho = o_1 \leq_I^{e_1,e_2} o_2$, $o_1 \neq o_t$ and $o_2 \neq o_t$. We assume $o_1 \neq o_2$ and $0 \in I$ (the other cases are simpler). Then,

$$\chi_\rho := [past^{e_1}_{v(o_1)} \wedge \xi_o^{e_1} \wedge \mathsf{F}_I(\psi(e_2, v(o_2)) \wedge \xi_o^{e_2})] \vee [\mathsf{F}_{\geq 0}\{\psi(e_1, v(o_1)) \wedge \xi_o^{e_1} \wedge \mathsf{F}_I(\psi(e_2, v(o_2)) \wedge \xi_o^{e_2})\}] \vee [past^{e_1}_{v(o_1)} \wedge \xi_o^{e_1} \wedge past^{e_2}_{v(o_2)} \wedge \xi_o^{e_2}] \vee [past^{e_2}_{v(o_2)} \wedge \xi_o^{e_2} \wedge \mathit{EqTime}(\psi(e_1, v(o_1)) \wedge \xi_o^{e_1})] \vee [\mathsf{F}_{\geq 0}\{\psi(e_2, v(o_2)) \wedge \xi_o^{e_2} \wedge \mathit{EqTime}(\psi(e_1, v(o_1)) \wedge \xi_o^{e_1})\}]$$

  The first two disjuncts handle the cases where (under the future semantics) the e_1-event of token o_1 precedes the e_2-event of token o_2, while the last three disjuncts consider the dual situation. In the latter case, the e_1-time of token o_1 and the e_2-time of token o_2 are equal.

Note that the MTL formula $\varphi_\forall$ is an MITL formula (resp., $\mathsf{MITL}_{(0,\infty)}$ formula) if the intervals in the trigger rules are non-singular (resp., belong to $\mathit{Intv}_{(0,\infty)}$). This concludes the proof. □

Encoding of trigger-less rules by TA. We note that an existential statement in a trigger-less rule requires the existence of an *a priori bounded number* of temporal events satisfying mutual temporal relations. Hence, one can easily construct a TA which guesses such a chain of events and checks the temporal relations by clock constraints and clock resets. Thus, by the well-known effective closure of TA under language union and intersection [1], we obtain the following result (for details, see [6]).

Proposition 10. *One can construct in exponential time a TA $\mathscr{A}_\exists$ over $2^{\mathscr{P}}$ such that, for each multi-timeline Π of SV and encoding w_Π of Π, w_Π is accepted by $\mathscr{A}_\exists$ iff Π satisfies all the trigger-less rules in R. Moreover, $\mathscr{A}_\exists$ has $2^{O(N_q)}$ states, $O(N_q)$ clocks, and maximal constant $O(K_P)$, where N_q is the overall number of quantifiers in the trigger-less rules of R.*

Conclusion of the construction. By applying Propositions 8–10 and well-known results about TA and MTL over finite timed words [1, 20], we obtain the main result of this section.

Theorem 11. *The future TP problem with simple trigger rules is decidable. Moreover, if the intervals in the atoms of the trigger rules are non-singular (resp., belong to $\mathit{Intv}_{(0,\infty)}$), then the problem is in* **EXPSPACE** *(resp., in* **PSPACE***).*

Proof. We fix an instance $P = (SV, R)$ of the problem with maximal constant K_P. Let $N_v := \sum_{x \in SV} |V_x|$, N_q be the overall number of quantifiers in the trigger-less rules of R, and N_a be the overall number of atoms in the trigger rules of R. By Propositions 8–10 and the effective closure of TA under language intersection [1], we can build a TA $\mathscr{A}_P$ and an MTL formula $\varphi_\forall$ such that there is a future plan of P iff $\mathscr{L}_T(\mathscr{A}_P) \cap \mathscr{L}_T(\varphi_\forall) \neq \emptyset$. Moreover, $\mathscr{A}_P$ has $2^{O(N_q+N_v)}$ states, $O(N_q + |SV|)$ clocks, and maximal constant $O(K_P)$, while $\varphi_\forall$ has $O(N_a + N_v)$ distinct subformulas and maximal constant $O(K_P)$. By [20], checking non-emptiness of $\mathscr{L}_T(\mathscr{A}_P) \cap \mathscr{L}_T(\varphi_\forall)$ is decidable. Hence, the first part of the theorem holds. For the second part, assume that the intervals in the trigger rules are non-singular (resp., belong to $Intv_{(0,\infty)}$). By Proposition 9, $\varphi_\forall$ is an MITL (resp., $\mathsf{MITL}_{(0,\infty)}$) formula. Thus, by [2], one can build a TA $\mathscr{A}_\forall$ accepting $\mathscr{L}_T(\varphi_\forall)$ having $2^{O(K_P \cdot (N_a+N_v))}$ states, $O(K_P \cdot (N_a + N_v))$ clocks (resp., $O(2^{(N_a+N_v)})$ states, $O(N_a + N_v)$ clocks), and maximal constant $O(K_P)$. Non-emptiness of a TA $\mathscr{A}$ can be solved by an **NPSPACE** search algorithm in the *region graph* of $\mathscr{A}$ which uses space logarithmic in the number of control states of $\mathscr{A}$ and polynomial in the number of clocks and in the length of the encoding of the maximal constant of $\mathscr{A}$ [1]. Thus, since $\mathscr{A}_P$, $\mathscr{A}_\forall$, and the intersection $\mathscr{A}_\wedge$ of $\mathscr{A}_P$ and $\mathscr{A}_\forall$ can be constructed on the fly, and the search in the region graph of $\mathscr{A}_\wedge$ can be done without explicitly constructing $\mathscr{A}_\wedge$, the result follows. □

4 Non-primitive recursive hardness of the future TP problem

In this section, we establish the following result.

Theorem 12. *Future TP with* one state variable *is non-primitive recursive-hard even under one of the following two assumptions:* either *(1) the trigger rules are simple,* or *(2) the intervals are in* $Intv_{(0,\infty)}$.

Theorem 12 is proved by a polynomial-time reduction from the halting problem for *Gainy counter machines* [12], a variant of standard Minsky machines, where the counters may erroneously increase. Fix such a machine $M = (Q, q_{init}, q_{halt}, n, \Delta)$, where (i) Q is a finite set of (control) locations, $q_{init} \in Q$ is the initial location, and $q_{halt} \in Q$ is the halting location, (ii) $n \in \mathbb{N} \setminus \{0\}$ is the number of counters, and (iii) $\Delta \subseteq Q \times L \times Q$ is a transition relation over the instruction set $L = \{\mathsf{inc}, \mathsf{dec}, \mathsf{zero}\} \times \{1, \ldots, n\}$. We adopt the following notational conventions. For an instruction $op \in L$, let $c(op) \in \{1, \ldots, n\}$ be the counter associated with op. For a transition $\delta \in \Delta$ of the form $\delta = (q, op, q')$, define $from(\delta) := q$, $op(\delta) := op$, $c(\delta) := c(op)$, and $to(\delta) := q'$. We denote by op_{init} the instruction $(\mathsf{zero}, 1)$. W.l.o.g., we make these assumptions: (i) for each transition $\delta \in \Delta$, $from(\delta) \neq q_{halt}$ and $to(\delta) \neq q_{init}$, and (ii) there is exactly one transition in Δ, denoted δ_{init}, having as source the initial location q_{init}.

An M-configuration is a pair (q, ν) consisting of a location $q \in Q$ and a counter valuation $\nu : \{1, \ldots, n\} \to \mathbb{N}$. Given two valuations ν and ν', we write $\nu \geq \nu'$ iff $\nu(c) \geq \nu'(c)$ for all $c \in \{1, \ldots, n\}$.

The *gainy semantics* is obtained from the standard Minsky semantics by allowing *incrementing* errors. Formally, M induces a transition relation, denoted by $\longrightarrow_{gainy}$, defined as follows: for configurations (q, ν) and (q', ν'), and instructions $op \in L$, $(q, \nu) \xrightarrow{op}_{gainy} (q', \nu')$ if the following holds, where $c = c(op)$: (i) $(q, op, q') \in \Delta$ and $\nu'(c') \geq \nu(c')$ for all $c' \in \{1, \ldots, n\} \setminus \{c\}$; (ii) $\nu'(c) \geq \nu(c) + 1$ if $op = (\mathsf{inc}, c)$; (iii) $\nu'(c) \geq \nu(c) - 1$ if $op = (\mathsf{dec}, c)$; (iv) $\nu(c) = 0$ if $op = (\mathsf{zero}, c)$.

A (gainy) computation of M is a finite sequence of global gainy transitions

$$(q_0, \nu_0) \xrightarrow{op_0}_{gainy} (q_1, \nu_1) \xrightarrow{op_1}_{gainy} \cdots \xrightarrow{op_{k-1}}_{gainy} (q_k, \nu_k)$$

M halts if there is a computation starting at the *initial* configuration (q_{init}, ν_{init}), where $\nu_{init}(c) = 0$ for all $c \in \{1, \ldots, n\}$, and leading to some halting configuration (q_{halt}, ν). The halting problem is to decide whether a given gainy machine M halts, and it was proved to be decidable and non-primitive recursive [12]. We prove the following result, from which Theorem 12 directly follows.

Proposition 13. *One can construct in polynomial time a TP instance $P = (\{x_M\}, R_M)$ s.t. the trigger rules in R_M are simple (resp., the intervals in P are in $Intv_{(0,\infty)}$) and M halts iff there is a future plan of P.*

Proof. We focus on the reduction where the intervals in P are in $Intv_{(0,\infty)}$. At the end of the proof, we show how to adapt the construction for the case of simple trigger rules with arbitrary intervals.

First, we define a suitable encoding of a computation of M as a timeline for x_M. For this, we exploit the finite set of symbols $V := V_{main} \cup V_{sec} \cup V_{dummy}$ corresponding to the finite domain of the state variable x_M. The sets of *main* values V_{main} is given by $V_{main} := \{(\delta, op) \in \Delta \times L \mid op \neq (\mathsf{inc}, c) \text{ if } op(\delta) = (\mathsf{zero}, c)\}$. The set of *secondary* values V_{sec} is defined as ($\#_{\mathsf{inc}}$ and $\#_{\mathsf{dec}}$ are two special symbols used as markers): $V_{sec} := V_{main} \times \{1, \ldots, n\} \times 2^{\{\#_{\mathsf{inc}}, \#_{\mathsf{dec}}\}}$. Finally, the set of *dummy values* is $(V_{main} \cup V_{sec}) \times \{dummy\}$.

Intuitively, in the encoding of an M-computation a main value (δ, op) keeps track of the transition δ used in the current step of the computation, while op represents the instruction exploited in the previous step (if any) of the computation. The set V_{sec} is used for encoding counter values, while the set V_{dummy} is used for specifying punctual time constraints by means of non-simple trigger rules over $Intv_{(0,\infty)}$. For a word $w \in V^*$, we denote by $||w||$ the length of the word obtained from w by removing dummy symbols.

For $c \in \{1, \ldots, n\}$ and $v_{main} = (\delta, op) \in V_{main}$, the *set $Tag(c, v_{main})$ of markers of counter c for the main value v_{main}* is the subset of $\{\#_{\mathsf{inc}}, \#_{\mathsf{dec}}\}$ defined as follows: (i) $\#_{\mathsf{inc}} \in Tag(c, v_{main})$ iff $op = (\mathsf{inc}, c)$; (ii) $\#_{\mathsf{dec}} \in Tag(c, v_{main})$ iff $op(\delta) = (\mathsf{dec}, c)$;

A *c-code for the main value $v_{main} = (\delta, op)$* is a finite word w_c over V_{sec} such that *either* (i) w_c is empty and $\#_{\mathsf{inc}} \notin Tag(c, v_{main})$, *or* (ii) $op(\delta) \neq (\mathsf{zero}, c)$ and $w_c = (v_{main}, c, Tag(c, v_{main}))(v_{main}, c, \emptyset, dummy)^{h_0} \cdot (v_{main}, c, \emptyset) \cdot (v_{main}, c, \emptyset, dummy)^{h_1} \cdots (v_{main}, c, \emptyset) \cdot (v_{main}, c, \emptyset, dummy)^{h_n}$ for some $n \geq 0$ and $h_0, h_1, \ldots, h_n \geq 0$. The c-code w_c encodes the value for counter c given by $||w_c||$. Intuitively, w_c can be seen as an interleaving of secondary values with dummy ones, the latter being present only for technical aspects, but not encoding any counter value.

A *configuration-code w for a main value $v_{main} = (\delta, op) \in V_{main}$* is a finite word over V of the form $w = v_{main} \cdot (v_{main}, dummy)^h \cdot w_1 \ldots w_n$, where $h \geq 0$ and for each counter $c \in \{1, \ldots, n\}$, w_c is a c-code for the main value v_{main}. The configuration-code w encodes the M-configuration$(from(\delta), \nu)$, where $\nu(c) = ||w_c||$ for all $c \in \{1, \ldots, n\}$. Note that if $op(\delta) = (\mathsf{zero}, c)$, then $\nu(c) = 0$ and $op \neq (\mathsf{inc}, c)$. Moreover, the marker $\#_{\mathsf{inc}}$ occurs in w iff op is an increment instruction, and in such a case $\#_{\mathsf{inc}}$ marks the first symbol of the encoding $w_{c(op)}$ of counter $c(op)$. Intuitively, if the operation performed in the previous step of the computation increments counter c, then the tag $\#_{\mathsf{inc}}$ "marks" the unit of the counter c in the current configuration which has been added by the increment. Additionally, the marker $\#_{\mathsf{dec}}$ occurs in w iff δ is a decrement instruction and the value of counter $c(\delta)$ in w is non-null; in such a case, $\#_{\mathsf{dec}}$ marks the first symbol of the encoding $w_{c(\delta)}$ of counter $c(\delta)$. Intuitively, if the operation to be performed in the current step decrements counter c and the current value of c is non-null, then the tag $\#_{\mathsf{dec}}$ marks the unit of the counter c in the current configuration which has to be removed by the decrement.

A *computation*-code is a sequence of configuration-codes $\pi = w_{(\delta_0, op_0)} \cdots w_{(\delta_k, op_k)}$, where, for all $0 \leq i \leq k$, $w_{(\delta_i, op_i)}$ is a configuration-code with main value (δ_i, op_i), and whenever $i < k$, it holds that $to(\delta_i) = from(\delta_{i+1})$ and $op(\delta_i) = op_{i+1}$. Note that by our assumptions $to(\delta_i) \neq q_{halt}$ for all $0 \leq i < k$, and $\delta_j \neq \delta_{init}$ for all $0 < j \leq k$. The computation-code π is *initial* if the first configuration-code $w_{(\delta_0, op_0)}$ is $(\delta_{init}, op_{init})$ (which encodes the initial configuration), and it is *halting* if for the last configuration-code $w_{(\delta_k, op_k)}$ in π, it holds that $to(\delta_k) = q_{halt}$. For all $0 \leq i \leq k$, let (q_i, ν_i) be the M-configuration encoded by the configuration-code $w_{(\delta_i, op_i)}$ and $c_i = c(\delta_i)$. The computation-code π is *well-formed* if, additionally, for all $0 \leq j \leq k-1$, the following holds: (i) $\nu_{j+1}(c) \geq \nu_j(c)$ for all $c \in \{1, \ldots, n\} \setminus \{c_j\}$ (*gainy monotonicity*); (ii) $\nu_{j+1}(c_j) \geq \nu_j(c_j) + 1$ if $op(\delta_j) = (\mathsf{inc}, c_j)$ (*increment req.*); (iii) $\nu_{j+1}(c_j) \geq \nu_j(c_j) - 1$ if $op(\delta_j) = (\mathsf{dec}, c_j)$ (*decrement req.*). Clearly, M halts *iff* there is an initial and halting well-formed computation-code.

Definition of x_M and R_M. We now define a state variable x_M and a set R_M of synchronization rules for x_M with intervals in $Intv_{(0,\infty)}$ such that the untimed part of every *future plan* of $P = (\{x_M\}, R_M)$ is an initial and halting well-formed computation-code. Thus, M halts iff there is a future plan of P.

Formally, variable x_M is given by $x_M = (V, T, D)$, where, for each $v \in V$, $D(v) =]0,\infty[$ if $v \notin V_{dummy}$, and $D(v) = [0,\infty[$ otherwise. Thus, we require that the duration of a non-dummy token is always greater than zero (*strict time monotonicity*). The value transition function T of x_M ensures the following.

Claim 14. The untimed parts of the timelines for x_M whose first token has value $(\delta_{init}, op_{init})$ correspond to the prefixes of initial computation-codes. Moreover, $(\delta_{init}, op_{init}) \notin T(v)$ for all $v \in V$.

By construction, it is a trivial task to define T so that the previous requirement is fulfilled. Let $V_{halt} = \{(\delta, op) \in V_{main} \mid to(\delta) = q_{halt}\}$. By Claim 14 and the assumption that $from(\delta) \neq q_{halt}$ for each transition $\delta \in \Delta$, for the initialization and halting requirements, it suffices to ensure that a timeline has a token with value $(\delta_{init}, op_{init})$ and a token with value in V_{halt}. This is captured by the trigger-less rules $\top \to \exists o[x_M = (\delta_{init}, op_{init})].\top$ and $\top \to \bigvee_{v \in V_{halt}} \exists o[x_M = v].\top$.

Finally, the crucial well-formedness requirement is captured by the trigger rules in R_M which express the following punctual time constraints. Note that we take advantage of the dense temporal domain to allow for the encoding of arbitrarily large values of counters in two time units.

- *2-Time distance between consecutive main values:* the overall duration of the sequence of tokens corresponding to a configuration-code amounts exactly to two time units. By Claim 14, strict time monotonicity, and the halting requirement, it suffices to ensure that each token tk having a main value in $V_{main} \setminus V_{halt}$ is eventually followed by a token tk' such that tk' has a main value and $\mathsf{s}(tk') - \mathsf{s}(tk) = 2$. To this aim, for each $v \in V_{main} \setminus V_{halt}$, we have the following non-simple trigger rule with intervals in $Intv_{(0,\infty)}$ which uses a dummy-token for capturing the punctual time constraint:

$$o[x_M = v] \to \bigvee_{u \in V_{main}} \bigvee_{u_d \in V_{dummy}} \exists o'[x_M = u] \exists o_d[x_M = u_d].o \leq^{\mathsf{s},\mathsf{s}}_{[1,+\infty[} o_d \wedge o_d \leq^{\mathsf{s},\mathsf{s}}_{[1,+\infty[} o' \wedge o \leq^{\mathsf{s},\mathsf{s}}_{[0,2]} o'.$$

- For a counter $c \in \{1,\ldots,n\}$, let $V_c \subseteq V_{sec}$ be the set of secondary states given by $V_{main} \times \{c\} \times 2^{\{\#_{\mathsf{inc}}, \#_{\mathsf{dec}}\}}$. We require that each token tk with a V_c-value of the form $((\delta, op), c, Tag)$ such that $c \neq c(\delta)$ and $to(\delta) \neq q_{halt}$ is eventually followed by a token tk' with a V_c-value such that $\mathsf{s}(tk') - \mathsf{s}(tk) = 2$. Note that our encoding, Claim 14, strict time monotonicity, and 2-Time distance between consecutive main values guarantee that the previous requirement captures *gainy monotonicity*. Thus, for each counter c and $v \in V_c$ such that v is of the form $((\delta, op), c, Tag)$, where $c \neq c(\delta)$ and $to(\delta) \neq q_{halt}$, we have the following non-simple trigger rule over $Intv_{(0,\infty)}$:

$$o[x_M = v] \to \bigvee_{u \in V_c} \bigvee_{u_d \in V_{dummy}} \exists o'[x_M = u] \exists o_d[x_M = u_d].o \leq^{\mathsf{s},\mathsf{s}}_{[1,+\infty[} o_d \wedge o_d \leq^{\mathsf{s},\mathsf{s}}_{[1,+\infty[} o' \wedge o \leq^{\mathsf{s},\mathsf{s}}_{[0,2]} o'.$$

- For capturing the increment and decrement requirements, by construction, it suffices to enforce that (i) each token tk with a V_c-value of the form $((\delta, op), c, Tag)$ such that $to(\delta) \neq q_{halt}$ and $\delta = (\mathsf{inc}, c)$ is eventually followed by a token tk' with a V_c-value which is *not* marked by the tag $\#_{\mathsf{inc}}$ such that $\mathsf{s}(tk') - \mathsf{s}(tk) = 2$, and (ii) each token tk with a V_c-value of the form $((\delta, op), c, Tag)$ such that $to(\delta) \neq q_{halt}$, $\delta = (\mathsf{dec}, c)$, and $\#_{\mathsf{dec}} \notin Tag$ is eventually followed by a token tk' with a V_c-value such that $\mathsf{s}(tk') - \mathsf{s}(tk) = 2$. These requirements can be expressed by non-simple trigger rules with intervals in $Intv_{(0,\infty)}$ similar to the previous ones.

Finally, to prove Proposition 13 for the case of simple trigger rules with arbitrary intervals, it suffices to remove the dummy values and replace the conjunction $o \leq^{\mathsf{s},\mathsf{s}}_{[1,+\infty[} o_d \wedge o_d \leq^{\mathsf{s},\mathsf{s}}_{[1,+\infty[} o' \wedge o \leq^{\mathsf{s},\mathsf{s}}_{[0,2]} o'$ in the previous trigger rules with the *punctual* atom $o \leq^{\mathsf{s},\mathsf{s}}_{[2,2]} o'$. This concludes the proof of Proposition 13. □

5 Hardness of future TP with simple rules and non-singular intervals

In this section, we first consider the future TP problem with simple trigger rules and non-singular intervals, and prove that it is **EXPSPACE**-hard by a polynomial-time reduction from a *domino-tiling problem for grids with rows of single exponential length*, which is known to be **EXPSPACE**-complete [16]. Since the reduction is standard, we refer the reader to [6] for the details of the construction.

Theorem 15. *The future TP problem with simple trigger rules and non-singular intervals is* **EXPSPACE***-hard (under polynomial-time reductions).*

We now focus on the special case with intervals of the forms $[0,a]$, with $a \in \mathbb{N} \setminus \{0\}$, and $[b,+\infty[$, with $b \in \mathbb{N}$, only, proving that it is **PSPACE**-hard by reducing periodic SAT to it in polynomial time.

The problem *periodic SAT* is defined as follows [21]. We are given a Boolean formula φ in *conjunctive normal form*, defined over two sets of variables, $\Gamma = \{x_1, \ldots, x_n\}$ and $\Gamma^{+1} = \{x_1^{+1}, \ldots, x_n^{+1}\}$, namely, $\varphi = \bigwedge_{t=1}^{m} (\bigvee_{x \in (\Gamma \cup \Gamma^{+1}) \cap L_t^+} x \vee \bigvee_{x \in (\Gamma \cup \Gamma^{+1}) \cap L_t^-} \neg x)$, where m is the number of conjuncts of φ and, for $1 \leq t \leq m$, L_t^+ (resp., L_t^-) is the set of variables occurring non-negated (resp., negated) in the t-th conjunct of φ. Moreover, the formula φ^j, for $j \in \mathbb{N} \setminus \{0\}$, is defined as φ in which we replace each variable $x_i \in \Gamma$ by a fresh one x_i^j, and $x_i^{+1} \in \Gamma^{+1}$ by x_i^{j+1}. Periodic SAT is to decide the satisfiability of the (infinite-length) formula $\Phi = \bigwedge_{j \in \mathbb{N} \setminus \{0\}} \varphi^j$, that is, deciding the existence of a truth assignment of (infinitely many) variables x_i^j, for $i = 1, \ldots, n$, $j \in \mathbb{N} \setminus \{0\}$, satisfying Φ. Periodic SAT is **PSPACE**-complete [21]; in particular membership to such a class is proved by showing that one can equivalently check satisfiability of the (finite-length) formula $\Phi_f = \bigwedge_{j=1}^{2^{2n}+1} \varphi^j$. Intuitively, 2^{2n} is the number of possible truth assignments to variables of $\Gamma \cup \Gamma^{+1}$, thus, after $2^{2n}+1$ copies of φ, we can find a repeated assignment: from that point, we can just copy the previous assignments. We now reduce periodic SAT to our problem. Hardness also holds when only a single state variable is involved, and also restricting to intervals of the form $[0,a]$.

Theorem 16. *The future TP problem with simple trigger rules and intervals* $[0,a]$*, with* $a \in \mathbb{N} \setminus \{0\}$*, is* **PSPACE***-hard (under polynomial-time reductions).*

Proof. Let us define the state variable $y = (V, T, D)$, where

1. $V = \{\$, \tilde{\$}, stop\} \cup \{x_i^\top, x_i^\bot, \tilde{x_i}^\top, \tilde{x_i}^\bot \mid i = 1, \ldots, n\}$,
2. $T(\$) = \{x_1^\top, x_1^\bot\}$, $T(\tilde{\$}) = \{\tilde{x_1}^\top, \tilde{x_1}^\bot\}$ and $T(stop) = \{stop\}$,
3. for $i = 1, \ldots, n-1$, $T(x_i^\top) = T(x_i^\bot) = \{x_{i+1}^\top, x_{i+1}^\bot\}$,
4. for $i = 1, \ldots, n-1$, $T(\tilde{x_i}^\top) = T(\tilde{x_i}^\bot) = \{\tilde{x_{i+1}}^\top, \tilde{x_{i+1}}^\bot\}$,
5. $T(x_n^\top) = T(x_n^\bot) = \{\tilde{\$}, stop\}$,
6. $T(\tilde{x_n}^\top) = T(\tilde{x_n}^\bot) = \{\$, stop\}$, and
7. for all $v \in V$, $D(v) = [2, +\infty[$.

Intuitively, we represent an assignment of variables x_i^j by means of a timeline for y: after every occurrence of the symbol \$, n tokens are present, one for each x_i, and the value $x_i^\top$ (resp., $x_i^\bot$) represents a positive (resp., negative) assignment of x_i^j, for some *odd* $j \geq 1$. Then, there is an occurrence of $\tilde{\$}$, after which n more tokens occur, again one for each x_i, and the value $\tilde{x_i}^\top$ (resp., $\tilde{x_i}^\bot$) represents a positive (resp., negative) assignment of x_i^j, for some *even* $j \geq 2$. See Figure 2 for an example.

We start with the next simple trigger rules, one for each $v \in V$: $o[y = v] \rightarrow o \leq^{\mathsf{s},\mathsf{e}}_{[0,2]} o$. Paired with the function D, they enforce all tokens' durations to be *exactly* 2: intuitively, since we exclude singular intervals, requiring, for instance, that a token o' starts t instants of time after the end of o, with $t \in [\ell, \ell+1]$ and $\ell \in \mathbb{N}$ is even, boils down to o' starting *exactly* ℓ instants after the end of o. We also observe that, given the constant token duration, the density of the time domain does not play any role in this proof.

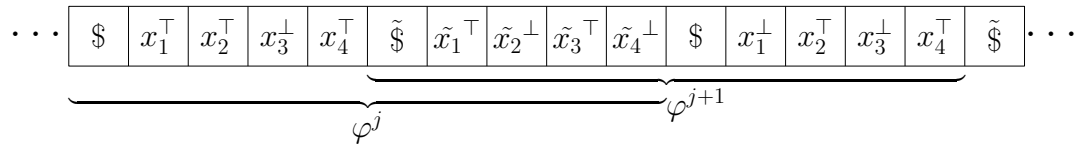

Figure 2: Let the formula φ be defined over two sets of variables, $\Gamma = \{x_1, x_2, x_3, x_4\}$ and $\Gamma^{+1} = \{x_1^{+1}, x_2^{+1}, x_3^{+1}, x_4^{+1}\}$. The j-th copy (we assume j is odd) of φ, i.e., φ^j, is satisfied by the assignment $x_1^j \mapsto \top, x_2^j \mapsto \top, x_3^j \mapsto \bot, x_4^j \mapsto \top, x_1^{j+1} \mapsto \top, x_2^{j+1} \mapsto \bot, x_3^{j+1} \mapsto \top, x_4^{j+1} \mapsto \bot$. The analogous for φ^{j+1}.

We now add the rules: (i) $\top \to \exists o[y = \$].o \geq^{\mathsf{s}}_{[0,1]} 0$; (ii) $\top \to \exists o[y = \tilde{\$}].o \geq^{\mathsf{s}}_{[0,1]} (2^{2n}+1)\cdot 2(n+1)$; (iii) $\top \to \exists o[y = stop].o \geq^{\mathsf{s}}_{[0,1]} (2^{2n}+2)\cdot 2(n+1)$. They respectively impose that (i) a token with value \$ starts exactly at $t = 0$ (recall that the duration of every token is 2); (ii) there exists a token with value $\tilde{\$}$ starting at $t = (2^{2n}+1)\cdot 2(n+1)$; (iii) a token with value $stop$ starts at $t = (2^{2n}+2)\cdot 2(n+1)$. We are forcing the timeline to encode truth assignments for variables $x_1^1, \ldots, x_n^1, \ldots, x_1^{2^{2n}+2}, \ldots, x_n^{2^{2n}+2}$: as a matter of fact, we will decide satisfiability of the finite formula $\Phi_f = \bigwedge_{j=1}^{2^{2n}+1} \varphi^j$, which is equivalent to Φ.

We now consider the next rules, that enforce the satisfaction of each φ^j or, equivalently, of φ over the assignments of $(x_1^j, \ldots, x_n^j, x_1^{j+1}, \ldots, x_n^{j+1})$. For the t-th conjunct of φ, we define the future simple rule:

$$o[y = \tilde{\$}] \to \Big(\bigvee_{x_i \in \Gamma \cap L_t^+} \exists o'[y = \tilde{x}_i^{\top}].o \leq^{\mathsf{e,s}}_{[0,4n]} o'\Big) \vee \Big(\bigvee_{x_i^{+1} \in \Gamma^{+1} \cap L_t^+} \exists o'[y = x_i^{\top}].o \leq^{\mathsf{e,s}}_{[0,4n]} o'\Big) \vee$$
$$\Big(\bigvee_{x_i \in \Gamma \cap L_t^-} \exists o'[y = \tilde{x}_i^{\bot}].o \leq^{\mathsf{e,s}}_{[0,4n]} o'\Big) \vee \Big(\bigvee_{x_i^{+1} \in \Gamma^{+1} \cap L_t^-} \exists o'[y = x_i^{\bot}].o \leq^{\mathsf{e,s}}_{[0,4n]} o'\Big) \vee$$
$$\exists o''[y = stop].o \leq^{\mathsf{e,s}}_{[0,2n]} o''.$$

Basically, this rule (the rule where the trigger has value \$ being analogous) states that, after every occurrence of $\tilde{\$}$, a token o', making true at least a (positive or negative) literal in the conjunct, must occur by $4n$ time instants (i.e., before the following occurrence of $\tilde{\$}$). The disjunct $\exists o''[y = stop].o \leq^{\mathsf{e,s}}_{[0,2n]} o''$ is present just to avoid evaluating φ on the n tokens before (the first occurrence of) $stop$.

The variable y and all synchronization rules can be generated in time polynomial in $|\varphi|$ (in particular, all interval bounds and time constants of time-point atoms have a value, encoded in binary, in $O(2^{2n})$). □

6 Conclusions and future work

In this paper, we investigated decidability and complexity issues for TP over dense temporal domains. Such a problem is known to be undecidable [7] even if restricted to simple trigger rules. Here, we have shown that decidability can be recovered by adding the future semantics to simple trigger rules. Moreover, future TP with simple trigger rules has been proved to be non-primitive recursive-hard (the same result holds in the case of future TP with all intervals being in $Intv_{(0,\infty)}$). Finally, if, additionally, singular intervals are avoided, it turns out to be **EXPSPACE**-complete, and **PSPACE**-complete if we consider only intervals in $Intv_{(0,\infty)}$.

Future work will focus on decidability of future TP with arbitrary trigger rules which remains open.

References

[1] R. Alur & D. L. Dill (1994): *A theory of timed automata*. Theoretical Computer Science 126(2), pp. 183–235, doi:10.1016/0304-3975(94)90010-8.

[2] R. Alur, T. Feder & T. A. Henzinger (1996): *The Benefits of Relaxing Punctuality*. Journal of the ACM 43(1), pp. 116–146, doi:10.1145/227595.227602.

[3] R. Alur & T. A. Henzinger (1993): *Real-Time Logics: Complexity and Expressiveness*. Information and Computation 104(1), pp. 35–77, doi:10.1006/inco.1993.1025.

[4] R. Alur & T. A. Henzinger (1994): *A Really Temporal Logic*. Journal of the ACM 41(1), pp. 181–204, doi:10.1145/174644.174651.

[5] J. Barreiro, M. Boyce, M. Do, J. Frank, M. Iatauro, T. Kichkaylo, P. Morris, J. Ong, E. Remolina, T. Smith & D. Smith (2012): *EUROPA: A Platform for AI Planning, Scheduling, Constraint Programming, and Optimization*. In: Proceedings of ICKEPS.

[6] L. Bozzelli, A. Molinari, A. Montanari & A. Peron (2018): *Complexity of timeline-based planning over dense temporal domains: exploring the middle ground*. Technical Report 2/2018, University of Udine, Italy. Available at `https://www.dimi.uniud.it/assets/preprints/2-2018-molinari.pdf`.

[7] L. Bozzelli, A. Molinari, A. Montanari & A. Peron (2018): *Decidability and Complexity of Timeline-based Planning over Dense Temporal Domains*. In: Proceedings of KR. Available at `https://www.uniud.it/it/ateneo-uniud/ateneo-uniud-organizzazione/dipartimenti/dmif/assets/preprints/1-2018-molinari`.

[8] L. Bozzelli, A. Molinari, A. Montanari, A. Peron & G. Woeginger (2018): *Timeline-Based Planning over Dense Temporal Domains with Trigger-less Rules is NP-Complete*. In: Proceedings of ICTCS.

[9] A. Cesta, G. Cortellessa, S. Fratini, A. Oddi & N. Policella (2007): *An Innovative Product for Space Mission Planning: An A Posteriori Evaluation*. In: Proceedings of ICAPS, pp. 57–64.

[10] S. Chien, D. Tran, G. Rabideau, S.R. Schaffer, D. Mandl & S. Frye (2010): *Timeline-Based Space Operations Scheduling with External Constraints*. In: Proceedings of ICAPS, pp. 34–41.

[11] M. Cialdea Mayer, A. Orlandini & A. Umbrico (2016): *Planning and Execution with Flexible Timelines: a Formal Account*. Acta Informatica 53(6–8), pp. 649–680, doi:10.1007/s00236-015-0252-z.

[12] S. Demri & R. Lazic (2009): *LTL with the freeze quantifier and register automata*. ACM Transactions on Computational Logic 10(3), pp. 16:1–16:30, doi:10.1145/1507244.1507246.

[13] J. Frank & A. Jónsson (2003): *Constraint-based Attribute and Interval Planning*. Constraints 8(4), pp. 339–364, doi:10.1023/A:1025842019552.

[14] N. Gigante, A. Montanari, M. Cialdea Mayer & A. Orlandini (2016): *Timelines are Expressive Enough to Capture Action-Based Temporal Planning*. In: Proceedings of TIME, pp. 100–109, doi:10.1109/TIME.2016.18.

[15] N. Gigante, A. Montanari, M. Cialdea Mayer & A. Orlandini (2017): *Complexity of Timeline-Based Planning*. In: Proceedings of ICAPS, pp. 116–124.

[16] D. Harel (1992): *Algorithmics: The spirit of computing*, 2nd edition. Wesley.

[17] A. K. Jónsson, P. H. Morris, N. Muscettola, K. Rajan & B. D. Smith (2000): *Planning in Interplanetary Space: Theory and Practice*. In: Proceedings of ICAPS, pp. 177–186.

[18] R. Koymans (1990): *Specifying Real-Time Properties with Metric Temporal Logic*. Real-Time Systems 2(4), pp. 255–299, doi:10.1007/BF01995674.

[19] N. Muscettola (1994): *HSTS: Integrating Planning and Scheduling*. In: Intelligent Scheduling, Morgan Kaufmann, pp. 169–212.

[20] J. Ouaknine & J. Worrell (2007): *On the decidability and complexity of Metric Temporal Logic over finite words*. Logical Methods in Computer Science 3(1), doi:10.2168/LMCS-3(1:8)2007.

[21] C. M. Papadimitriou (1994): *Computational complexity*. Addison-Wesley, Reading, Massachusetts.