Undecidability of Future Timeline-based Planning over Dense Temporal Domains*

Laura Bozzelli[1], Alberto Molinari[2], Angelo Montanari[2], and Adriano Peron[1]

[1] University of Napoli "Federico II", Napoli, Italy
lr.bozzelli@gmail.com; adrperon@unina.it
[2] University of Udine, Udine, Italy
molinari.alberto@gmail.com; angelo.montanari@uniud.it

Abstract. The present work focuses on timeline-based planning over dense temporal domains. In automated planning, the temporal domain is commonly assumed to be discrete, the dense case being dealt with by resorting to some form of discretization. In the last years, the planning problem over dense temporal domains has been finally addressed both in the timeline-based setting and, very recently, in the action-based one. Dense timeline-based planning, in its full generality, has been shown to be undecidable. Decidability has been recovered by imposing suitable syntactic and/or semantic restrictions (the complexity of decidable fragments varies a lot, spanning from non-primitive recursive hardness to **NP**-completeness, passing through **EXPSPACE**- and **PSPACE**-completeness). In this paper, we proved that restricting to the future fragment is not enough to get decidability.

Keywords: Automated planning· Timeline-based planning · Dense time · Decidability.

1 Introduction

The present contribution adds an important piece to the general picture of timeline-based planning over dense temporal domains by showing that restricting to the future fragment is not enough to get decidability.

Inspired by classical control theory, timeline-based planning has emerged as a viable alternative to the more common 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, lead to a state where the goal is met. Timeline-based planning looks at the problem more abstractly, focusing on what has to happen to meet the goal instead of what an agent has to do to reach it.

In timeline-based planning, planning domains are described as collections of independent, but interacting, components, each one consisting of a set of state variables. The evolution of the values of state variables over time is modeled 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.

The temporal domain is commonly assumed to be discrete, the dense case being dealt with by forcing a more or less artificial discretization of the domain.

In [4], Gigante et al. showed that (discrete) timeline-based planning (TP for short) with bounded temporal relations and token durations, and no temporal horizon, is **EXPSPACE**-complete and expressive enough to capture action-based temporal planning. Later, Gigante et al. proved that TP with unbounded interval relations is still **EXPSPACE**-complete [5] (if an upper bound to the temporal horizon is added, the problem becomes **NEXPTIME**-complete), and that the same holds for TP with recurrent goals [3].

Even though the potentialities of automated planning over dense time, in terms of both naturalness and expressiveness, are commonly recognized, its systematic investigation has been undertaken only very recently.

The computational complexity of action-based temporal planning, as represented by PDDL 2.1, over dense time has been addressed in [7] (the problem is known to be **EXPSPACE**-complete over discrete time). The problem has been shown to be **PSPACE**-complete when self-overlap is forbidden (self-overlap means that actions are allowed to overlap already running instances of themselves), whereas, when allowed, it becomes **EXPSPACE**-complete with ϵ-*separation* (a minimum amount ϵ of separation between mutually exclusive events is guaranteed) and **undecidable** without ϵ-*separation* (separation is simply required to be non-zero).

TP over dense time has been studied in depth in [1]. The general 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 general, synchronization rules allow a universal quantification over the tokens of a timeline (*triggers*). By disallowing it and retaining only rules in purely existential form (*trigger-less rules*), the TP problem becomes **NP**-complete. In [1], various intermediate cases have been investigated.

A first restriction that can be imposed on *trigger rules* is that the name of a non-trigger token appears exactly once in the body (*interval atoms*) of the rule (*simple trigger rules*). Such a syntactical restriction avoids comparisons of multiple token time-events with a non-trigger reference time-event. A second restriction concerns future and past tokens. When a token is "selected" by a trigger, the synchronization rule allows one to compare tokens of the timelines both preceding (past) and following (future) the trigger token. One can restrict the comparison only to tokens in the future with respect to the trigger (*future semantics of trigger rules*). In [1], it has been shown that the TP problem restricted to simple trigger rules remains **undecidable**. Decidability can be recovered by adding the future semantics to simple trigger rules: future TP with simple trigger rules has been proved to be non-primitive recursive-hard. Better complexity results can be obtained by restricting also the type of intervals used

in the simple trigger rules to compare tokens. In particular, future TP with simple trigger rules without singular intervals (an interval is called *singular* if it has the form $[a, a]$, for $a \in \mathbb{N}$) is **EXPSPACE**-complete, **PSPACE**-complete if one only allows intervals of the forms $[0, a]$ and $[b, +\infty[$ are considered.

The decidability status of the TP problem with arbitrary trigger rules under the future semantics was left open in [1] (it was only shown that it is at least non-primitive recursive even under the assumption that the intervals in the rules have the forms $[0, a]$ and $[b, +\infty[$). In this paper, we negatively answer the open issue: future TP over dense time is undecidable.

The paper is organized as follows. In Section 2, we recall the distinctive features of the TP framework. Then, in Section 3, we prove that future TP is undecidable. Conclusions give a short assessment of the work and outline future research directions.

2 Preliminaries

In this section, we provide some notation and background knowledge about the TP problem. For a systematic account of expressiveness and complexity of timeline-based planning, including a careful analysis of the way in which temporal uncertainty and nondeterminism are dealt with, we refer the reader to [6] and follow-up publications.

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 $I \in Intv$ 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 the following, we recall the TP framework as presented in [2, 4]. 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* $D_x : V_x \to Intv$ *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 $\pi = (v_0, d_0) \dots (v_n, d_n)$ 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.

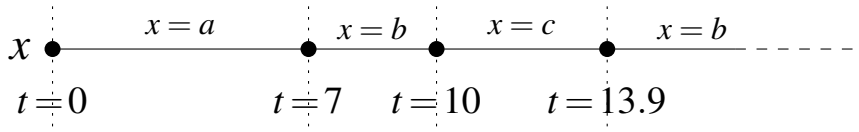

Fig. 1. An example of timeline $(a, 7)(b, 3)(c, 3.9)\cdots$ for the state variable $x = (V_x, T_x, D_x)$, where $V_x = \{a, b, c, \ldots\}$, $b \in T_x(a)$, $c \in T_x(b)$, $b \in T_x(c)\ldots$ and $D_x(a) = [5, 8]$, $D_x(b) = [1, 4]$, $D_x(c) = [2, \infty[\ldots$

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* $o_1, o_2 \in \Sigma$, $I \in Intv$, $n \in \mathbb{N}$, *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 *$\mathcal{E}$ for a finite set SV of state variables is a statement of the form:*

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

where $\mathcal{C}$ 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 $\mathcal{C}$, 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 $\mathcal{E}$ *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 $\mathcal{E}$ *if there exists a* Σ*-assignment* λ_Π *for* Π *consistent with* $\mathcal{E}$ *such that each atom in* $\mathcal{C}$ *is satisfied by* λ_Π.

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

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

where $o_0 \in \Sigma$, $x_0 \in SV$, $v_0 \in V_{x_0}$, *and* $\mathcal{E}_1, \ldots, \mathcal{E}_k$ *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* $\mathcal{E}_i$ *(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* $\mathcal{R}$ *is* simple *if for each existential statement* $\mathcal{E}$ *of* $\mathcal{R}$ *and each token name* o *distinct from the trigger, there is at most one* interval atom *of* $\mathcal{E}$ *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 $\mathcal{E}_i$ 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 $\mathcal{R}$ *of* SV, Π satisfies $\mathcal{R}$ *if* Π *satisfies some existential statement of* $\mathcal{R}$. *Given a* trigger rule $\mathcal{R}$ *of* SV *with trigger* $o_0[x_0 = v_0]$, Π satisfies $\mathcal{R}$ *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* $\mathcal{E}$ *of* $\mathcal{R}$ *and a* Σ*-assignment* λ_Π *for* Π *which is consistent with* $\mathcal{E}$ *such that* $\lambda_\Pi(o_0) = (\Pi(x_0), i)$ *and* λ_Π *satisfies all the atoms of* $\mathcal{E}$.

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* $\mathcal{R} = o_0[x_0 = v_0] \rightarrow \mathcal{E}_1 \vee \mathcal{E}_2 \vee \ldots \vee \mathcal{E}_k$ *if* Π *satisfies the trigger rule obtained from* $\mathcal{R}$ *by replacing each existential statement* $\mathcal{E}_i = \exists o_1[x_1 = v_1] \cdots \exists o_n[x_n = v_n].\mathcal{C}$ *with* $\exists o_1[x_1 = v_1] \cdots \exists o_n[x_n = v_n].\mathcal{C} \wedge \bigwedge_{i=1}^{n} o_0 \leq^{\mathbf{s},\mathbf{s}}_{[0,+\infty[} o_i$.

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. Trigger-less rules can be used to express initial conditions and the goals of the

problem, while trigger rules are useful to specify invariants and response requirements. 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.

3 Undecidability of the future TP problem

In this section, we establish the following result.

Theorem 1. *Future TP with* one state variable *is undecidable even if the intervals are in* $Intv_{(0,\infty)}$.

Theorem 1 is proved by a polynomial-time reduction from the *halting problem for Minsky* 2*-counter machines* [8]. Such a machine is a tuple $M = (Q, q_{init}, q_{halt}, \Delta)$, where Q is a finite set of (control) locations, $q_{init} \in Q$ is the initial location, $q_{halt} \in Q$ is the halting location, and $\Delta \subseteq Q \times L \times Q$ is a transition relation over the instruction set $L = \{\mathsf{inc}, \mathsf{dec}, \mathsf{zero}\} \times \{1, 2\}$.

We adopt the following notational conventions. For an instruction $op = (\_, c) \in L$, let $c(op) := c \in \{1, 2\}$ be the *counter* associated with op. For a transition $\delta \in \Delta$ of the form $\delta = (q, op, q')$, we define $from(\delta) := q$, $op(\delta) := op$, $c(\delta) := c(op)$, and $to(\delta) := q'$. Without loss of generality, we make these assumptions:

- for each transition $\delta \in \Delta$, $from(\delta) \neq q_{halt}$ and $to(\delta) \neq q_{init}$, and
- 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 $\nu : \{1, 2\} \to \mathbb{N}$. M induces a transition relation, denoted by $\longrightarrow$, over pairs of M-configurations defined as follows. For configurations (q, ν) and (q', ν'), $(q, \nu) \longrightarrow (q', \nu')$ if for some instruction $op \in L$, $(q, op, q') \in \Delta$ and the following holds, where $c \in \{1, 2\}$ is the counter associated with the instruction op: (i) $\nu'(c') = \nu(c')$ if $c' \neq c$; (ii) $\nu'(c) = \nu(c) + 1$ if $op = (\mathsf{inc}, c)$; (iii) $\nu'(c) = \nu(c) - 1$ if $op = (\mathsf{dec}, c)$ (in particular, it has to be $\nu(c) > 0$); and (iv) $\nu'(c) = \nu(c) = 0$ if $op = (\mathsf{zero}, c)$.

A computation of M is a non-empty *finite* sequence $C_1, \ldots, C_k$ of configurations such that $C_i \longrightarrow C_{i+1}$ for all $1 \leq i < k$. M *halts* if there is a computation starting at the *initial* configuration (q_{init}, ν_{init}), where $\nu_{init}(1) = \nu_{init}(2) = 0$, and leading to some halting configuration (q_{halt}, ν). The halting problem is to decide whether a given machine M halts, and it is was proved to be undecidable [8]. We prove the following result, from which Theorem 1 directly follows.

Proposition 1. *One can construct (in polynomial time) a TP instance (domain)* $P = (\{x_M\}, R_M)$ *where the intervals in* P *are in* $Intv_{(0,\infty)}$ *such that* M *halts* iff *there exists a future plan for* P.

Proof. First, we define a suitable encoding of a computation of M as the untimed part of a timeline (i.e., neglecting tokens' durations and accounting only for their values) for x_M. For this, we exploit the finite set of symbols $V := V_{main} \cup V_{sec}$ corresponding to the finite domain of the state variable x_M. The set of *main* values V_{main} is the set of M-transitions, i.e. $V_{main} = \Delta$. The set of *secondary* values V_{sec} is defined as $V_{sec} := \Delta \times \{1, 2\} \times \{\#, beg, end\}$, where $\#$, *beg*, and *end* are three special symbols used as markers. Intuitively, in the encoding of an M-computation a main value keeps track of the transition used in the current step of the computation, while the set V_{sec} is used for encoding counter values.

For $c \in \{1, 2\}$, a *c-code for the main value* $\delta \in \Delta$ is a finite word w_c over V_{sec} of the form $(\delta, c, beg) \cdot (\delta, c, \#)^h \cdot (\delta, c, end)$ for some $h \geq 0$ such that $h = 0$ if $op(\delta) = (\mathsf{zero}, c)$. The c-code w_c encodes the value for counter c given by h (or equivalently $|w_c| - 2$). Note that only the occurrences of the symbols $(\delta, c, \#)$ encode units in the value of counter c, while the symbol (δ, c, beg) (resp., (δ, c, end)) is only used as left (resp., right) marker in the encoding.

A *configuration-code* w *for a main value* $\delta \in \Delta$ is a finite word over V of the form $w = \delta \cdot w_1 \cdot w_2$ such that for each counter $c \in \{1, 2\}$, w_c is a c-code for the main value δ. The configuration-code w encodes the M-configuration $(from(\delta), \nu)$, where $\nu(c) = |w_c| - 2$ for all $c \in \{1, 2\}$. Note that if $op(\delta) = (\mathsf{zero}, c)$, then $\nu(c) = 0$.

A *computation*-code is a non-empty sequence of configuration-codes $\pi = w_{\delta_1} \cdots w_{\delta_k}$, where for all $1 \leq i \leq k$, w_{δ_i} is a configuration-code with main value δ_i, and whenever $i < k$, it holds that $to(\delta_i) = from(\delta_{i+1})$. Note that by our assumptions $to(\delta_i) \neq q_{halt}$ for all $1 \leq i < k$, and $\delta_j \neq \delta_{init}$ for all $1 < j \leq k$. The computation-code π is *initial* if the first configuration-code w_{δ_1} has the main value δ_{init} and encodes the initial configuration, and it is *halting* if for the last configuration-code w_{δ_k} in π, it holds that $to(\delta_k) = q_{halt}$. For all $1 \leq i \leq k$, let (q_i, ν_i) be the M-configuration encoded by the configuration-code w_{δ_i} and $c_i = c(\delta_i)$. The computation-code π is *well-formed* if, additionally, for all $1 \leq j < k$, the following holds:

- $\nu_{j+1}(c) = \nu_j(c)$ if either $c \neq c_j$ or $op(\delta_j) = (\mathsf{zero}, c_j)$ (*equality requirement*);
- $\nu_{j+1}(c_j) = \nu_j(c_j) + 1$ if $op(\delta_j) = (\mathsf{inc}, c_j)$ (*increment requirement*);
- $\nu_{j+1}(c_j) = \nu_j(c_j) - 1$ if $op(\delta_j) = (\mathsf{dec}, c_j)$ (*decrement requirement*).

Clearly, M halts *iff* there exists 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 if and only if there is a future plan of P.

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

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

By construction, it is a trivial task to define T so that the previous requirement is fulfilled.

Let $V_{halt} = \{\delta \in \Delta \mid to(\delta) = q_{halt}\}$. By Claim 3 and the assumption that $from(\delta) \neq q_{halt}$ for each transition $\delta \in \Delta$, in order to enforce the initialization and halting requirements, it suffices to ensure that a timeline has a token with value δ_{init} and a token with value in V_{halt}. This is captured by the trigger-less rules $\top \rightarrow \exists o[x_M = \delta_{init}].\top$ and $\top \rightarrow \bigvee_{v \in V_{halt}} \exists o[x_M = v].\top$.

Finally, the crucial well-formedness requirement is captured by the trigger rules in R_M which express punctual time constraints[3]. We refer the reader to Figure 2, that gives an intuition on the properties enforced by the rules we are about to define. In particular, we essentially take advantage of the dense temporal domain to allow for the encoding of arbitrarily large values of counters in two time units.

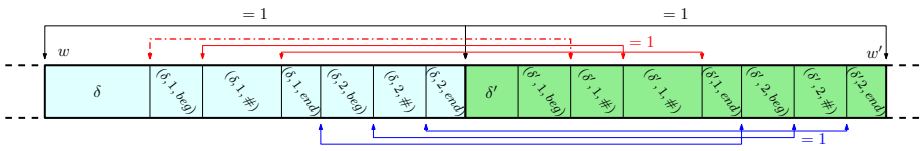

Fig. 2. The figure shows two adjacent configuration-codes, w (highlighted in cyan) and w' (in green), the former for $\delta = (q, (\mathsf{inc}, 1), q') \in \Delta$ and the latter for $\delta' = (q', \dots) \in \Delta$; w encodes the M-configuration (q, ν) where $\nu(1) = \nu(2) = 1$, and w' the M-configuration (q', ν') where $\nu'(1) = 2$ and $\nu'(1) = 1$.
The "1-Time distance between consecutive main values requirement" (represented by black lines with arrows) forces a token with a main value to be followed, after exactly one time instant, by another token with a main value.
Since $op(\delta) = (\mathsf{inc}, 1)$, the value of counter 2 does not change in this computation step, and thus the values for counter 2 encoded by w and w' must be equal. To this aim the "equality requirement" (represented by blue lines with arrows) sets a one-to-one correspondence between pairs of tokens associated with counter 2 in w and w' (more precisely, a token tk with value $(\delta, 2, \_)$ in w is followed by a token tk' with value $(\delta', 2, \_)$ in w' such that $\mathsf{s}(tk') - \mathsf{s}(tk) = 1$ and $\mathsf{e}(tk') - \mathsf{e}(tk) = 1$).
Finally, the "increment requirement" (red lines) performs the increment of counter 1 by doing something analogous to the previous case, but with a difference: the token tk' with value $(\delta', 1, \#)$ is in w' in the place where the token tk with value $(\delta, 1, beg)$ was in w (i.e., $\mathsf{s}(tk') - \mathsf{s}(tk) = 1$ and $\mathsf{e}(tk') - \mathsf{e}(tk) = 1$). The token tk'' with value $(\delta', 1, beg)$ is "anticipated", in such a way that $\mathsf{e}(tk'') - \mathsf{s}(tk) = 1$ (this is denoted by the dashed red line): the token with main value δ' in w' has a shorter duration than that with value δ in w, leaving space for tk'', so as to represent the unit added by δ to counter 1. Clearly density of the time domain plays a fundamental role here.

[3] Such punctual constrains are expressed by pairs of conjoined atoms whose intervals are in $Intv_{(0,\infty)}$.

Trigger rules for 1-Time distance between consecutive main values. We define non-simple trigger rules requiring that the overall duration of the sequence of tokens corresponding to a configuration-code amounts exactly to two time units. By Claim 3, 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 $\mathsf{s}(tk') - \mathsf{s}(tk) = 1$ (this denotes—with a little abuse of notation—that the difference of start times is exactly 1). To this aim, for each $v \in V_{main} \setminus V_{halt}$, we write the non-simple trigger rule with intervals in $Intv_{(0,\infty)}$:

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

Trigger rules for the equality requirement. In order to ensure the equality requirement, we exploit the fact that the end time of a token along a timeline corresponds to the start time of the next token (if any). Let $V^{=}_{sec}$ be the set of secondary states $(\delta, c, t) \in V_{sec}$ such that $to(\delta) \neq q_{halt}$, and either $c \neq c(\delta)$ or $op(\delta) = (\mathsf{zero}, c)$. Moreover, for a counter $c \in \{1, 2\}$ and a tag $t \in \{beg, \#, end\}$, let $V^t_c \subseteq V_{sec}$ be the set of secondary states given by $\Delta \times \{c\} \times \{t\}$. We require the following:

(*) each token tk with a $(V^t_c \cap V^{=}_{sec})$-value is eventually followed by a token tk' with a V^t_c-value such that $\mathsf{s}(tk') - \mathsf{s}(tk) = 1$ (i.e., the difference of start times is exactly 1). Moreover, if $t \neq end$, then $\mathsf{e}(tk') - \mathsf{e}(tk) = 1$ (i.e., the difference of end times is exactly 1).

Condition (*) is captured by the following non-simple trigger rules with intervals in $Intv_{(0,\infty)}$:

- for each $v \in V^t_c \cap V^{=}_{sec}$ and $t \neq end$,

$$o[x_M = v] \rightarrow$$

$$\textstyle\bigvee_{u \in V^t_c} \exists o'[x_M = u].\, o \leq^{\mathsf{s},\mathsf{s}}_{[1,+\infty[} o' \wedge o \leq^{\mathsf{s},\mathsf{s}}_{[0,1]} o' \wedge o \leq^{\mathsf{e},\mathsf{e}}_{[1,+\infty[} o' \wedge o \leq^{\mathsf{e},\mathsf{e}}_{[0,1]} o';$$

- for each $v \in V^{end}_c \cap V^{=}_{sec}$,

$$o[x_M = v] \rightarrow \bigvee_{u \in V^{end}_c} \exists o'[x_M = u].\, o \leq^{\mathsf{s},\mathsf{s}}_{[1,+\infty[} o' \wedge o \leq^{\mathsf{s},\mathsf{s}}_{[0,1]} o'.$$

We now show that Condition (*) together with strict time monotonicity and 1-Time distance between consecutive main values ensure the equality requirement. Let π be a timeline of x_M satisfying all the rules defined so far, w_δ and $w_{\delta'}$ two *adjacent* configuration-codes along π with w_δ preceding $w_{\delta'}$ (note that $to(\delta) \neq q_{halt}$), and $c \in \{1, 2\}$ a counter such that either $c \neq c(\delta)$ or $op(\delta) = (\mathsf{zero}, c)$. Let $tk_0 \cdots tk_{\ell+1}$ (resp., $tk'_0 \cdots tk'_{\ell'+1}$) be the sequence of tokens associated with the c-code of w_δ (resp., $w_{\delta'}$). We need to show that $\ell = \ell'$. By construction tk_0 and tk'_0 have value in V^{beg}_c, $tk_{\ell+1}$ and $tk'_{\ell'+1}$ have value in V^{end}_c, and for all $1 \leq i \leq \ell$ (resp., $1 \leq i' \leq \ell'$), tk_i has value in $V^{\#}_c$ (resp.,

$tk'_{i'}$ has value in $V_c^{\#}$). Then strict time monotonicity, 1-Time distance between consecutive main values, and Condition (*) guarantee the existence of an *injective* mapping $g : \{tk_0, \ldots, tk_{\ell+1}\} \to \{tk'_0, \ldots, tk'_{\ell'+1}\}$ such that $g(tk_0) = tk'_0$, $g(tk_{\ell+1}) = tk'_{\ell'+1}$, and for all $0 \leq i \leq \ell$, if $g(tk_i) = tk'_j$ (note that $j < \ell' + 1$), then $g(tk_{i+1}) = tk'_{j+1}$ (we recall that the end time of a token is equal to the start time of the next token along a timeline, if any). These properties ensure that g is *surjective* as well. Hence, g is a bijection and $\ell' = \ell$.

Trigger rules for the increment requirement. Let V_{sec}^{inc} be the set of secondary states $(\delta, c, t) \in V_{sec}$ such that $to(\delta) \neq q_{halt}$ and $op(\delta) = (\mathsf{inc}, c)$. By reasoning like in the case of the rules ensuring the equality requirement, in order to express the increment requirement, it suffices to enforce the following conditions for each counter $c \in \{1, 2\}$:

(i) each token tk with a $(V_c^{beg} \cap V_{sec}^{\mathsf{inc}})$-value is eventually followed by a token tk' with a V_c^{beg}-value such that $\mathsf{e}(tk') - \mathsf{s}(tk) = 1$ (i.e., the difference between the end time of token tk' and the start time of token tk is exactly 1);

(ii) for each $t \in \{beg, \#\}$, each token tk with a $(V_c^t \cap V_{sec}^{\mathsf{inc}})$-value is eventually followed by a token tk' with a $V_c^{\#}$-value such that $\mathsf{s}(tk') - \mathsf{s}(tk) = 1$ and $\mathsf{e}(tk') - \mathsf{e}(tk) = 1$ (i.e., the difference of start times and end times is exactly 1). Observe that the token with a $(V_c^{beg} \cap V_{sec}^{\mathsf{inc}})$-value is associated with a token with $V_c^{\#}$-value anyway;

(iii) each token tk with a $(V_c^{end} \cap V_{sec}^{\mathsf{inc}})$-value is eventually followed by a token tk' with a V_c^{end}-value such that $\mathsf{s}(tk') - \mathsf{s}(tk) = 1$ (i.e., the difference of start times is exactly 1);

Intuitively, if w and w' are two *adjacent* configuration-codes along a timeline of x_M, with w preceding w', (i) and (ii) force a token tk' with a $V_c^{\#}$-value in w' to "take the place" of the token tk with $(V_c^{beg} \cap V_{sec}^{\mathsf{inc}})$-value in w (i.e., they have the same start and end times). Moreover a token with V_c^{beg}-value must immediately precede tk' in w'.

These requirements can be expressed by non-simple trigger rules with intervals in $Intv_{(0,\infty)}$ similar to the ones defined for the equality requirement.

Trigger rules for the decrement requirement. For capturing the decrement requirement, it suffices to enforce the following conditions for each counter $c \in \{1, 2\}$, where V_{sec}^{dec} denotes the set of secondary states $(\delta, c, t) \in V_{sec}$ such that $to(\delta) \neq q_{halt}$ and $op(\delta) = (\mathsf{dec}, c)$:

(i) each token tk with a $(V_c^{beg} \cap V_{sec}^{\mathsf{dec}})$-value is eventually followed by a token tk' with a V_c^{beg}-value such that $\mathsf{s}(tk') - \mathsf{e}(tk) = 1$ (i.e., the difference between the start time of token tk' and the end time of token tk is exactly 1);

(ii) each token tk with a $(V_c^{\#} \cap V_{sec}^{\mathsf{dec}})$-value is eventually followed by a token tk' with a V_c^t-value where $t \in \{beg, \#\}$ such that $\mathsf{s}(tk') - \mathsf{s}(tk) = 1$ and $\mathsf{e}(tk') - \mathsf{e}(tk) = 1$ (i.e., the difference of start times and end times is exactly 1).

(iii) each token tk with a $(V_c^{end} \cap V_{sec}^{\mathsf{dec}})$-value is eventually followed by a token tk' with a V_c^{end}-value such that $\mathsf{s}(tk') - \mathsf{s}(tk) = 1$ (i.e., the difference of start times is exactly 1);

Analogously, (i) and (ii) produce an effect which is symmetric w.r.t. the case of increment.

Again, these requirements can be easily expressed by non-simple trigger rules with intervals in $Intv_{(0,\infty)}$ as done before for expressing the equality requirement.

By construction, the untimed part of a future plan of $P = (\{x_M\}, R_M)$ is an initial and halting well-formed computation-code. Vice versa, by exploiting denseness of the temporal domain, the existence of an initial and halting well-formed computation-code implies the existence of a future plan of P. This concludes the proof of Proposition 1.

4 Conclusion and future work

In this paper, we solved a problem left open in [1] by showing that future timeline-based planning with arbitrary trigger rules is undecidable over dense temporal domains.

We glimpse two directions for future research. On the one hand, we would like to compare expressive power and complexity of action- and timeline-based planning over dense time in a systematic way. On the other hand, we would like to study the effects of applying to the discrete case the same restrictions we imposed to timeline-based planning over dense time.

Acknowledgements

We would like to acknowledge the support from the GNCS project *Strategic Reasoning and Automatic Synthesis of Multi-Agent Systems.*

References

1. Bozzelli, L., Molinari, A., Montanari, A., Peron, A., Woeginger, G.J.: Timeline-based planning over dense temporal domains. Theor. Comput. Sci. **813**, 305–326 (2020). https://doi.org/10.1016/j.tcs.2019.12.030, https://doi.org/10.1016/j.tcs.2019.12.030
2. Cialdea Mayer, M., Orlandini, A., Umbrico, A.: Planning and Execution with Flexible Timelines: a Formal Account. Acta Informatica **53**(6–8), 649–680 (2016). https://doi.org/10.1007/s00236-015-0252-z
3. Della Monica, D., Gigante, N., Montanari, A., Sala, P.: A novel automata-theoretic approach to timeline-based planning. In: Thielscher, M., Toni, F., Wolter, F. (eds.) Principles of Knowledge Representation and Reasoning: Proceedings of the Sixteenth International Conference, KR 2018, Tempe, Arizona, 30 October - 2 November 2018. pp. 541–550. AAAI Press (2018), https://aaai.org/ocs/index.php/KR/KR18/paper/view/18024
4. Gigante, N., Montanari, A., Cialdea Mayer, M., Orlandini, A.: Timelines are Expressive Enough to Capture Action-based Temporal Planning. In: Proc. of TIME. pp. 100–109. IEEE Computer Society (2016). https://doi.org/10.1109/TIME.2016.18
5. Gigante, N., Montanari, A., Cialdea Mayer, M., Orlandini, A.: Complexity of timeline-based planning. In: Proc. of ICAPS. pp. 116–124. AAAI (2017)

6. Gigante, N.: Timeline-based Planning: Expressiveness and Complexity. Ph.D. thesis, University of Udine, Italy (2019), available on *arXiv* at: https://arxiv.org/abs/1902.06123
7. Gigante, N., Michieli, A., Montanari, A., Scala, E.: Decidability and complexity of action-based temporal planning over dense time. In: Proceedings of the 34th AAAI Conference on Artificial Intelligence. AAAI Press (2020)
8. Minsky, M.L.: Computation: Finite and Infinite Machines. Automatic Computation, Prentice-Hall, Inc. (1967)