Taming the Complexity of Timeline-Based Planning over Dense Temporal Domains

Laura Bozzelli
University of Napoli "Federico II", Napoli, Italy

Angelo Montanari
University of Udine, Udine, Italy

Adriano Peron
University of Napoli "Federico II", Napoli, Italy

Abstract

The problem of timeline-based planning (TP) over dense temporal domains is known to be undecidable. In this paper, we introduce two semantic variants of TP, called *strong minimal and weak minimal* semantics, which allow to express meaningful properties. Both semantics are based on the minimality in the time distances of the existentially-quantified time events from the universally-quantified reference event, but the weak minimal variant distinguishes minimality in the past from minimality in the future. Surprisingly, we show that, despite the (apparently) small difference in the two semantics, for the strong minimal one, the TP problem is still undecidable, while for the weak minimal one, the TP problem is just **PSPACE**-complete. Membership in **PSPACE** is determined by exploiting a strictly more expressive extension (ECA^{+}) of the well-known robust class of Event-Clock Automata (ECA) that allows to encode the weak minimal TP problem and to reduce it to non-emptiness of Timed Automata (TA). Finally, an extension of ECA^{+}(ECA^{++}) is considered, proving that its non-emptiness problem is undecidable. We believe that the two extensions of ECA (ECA^{+} and ECA^{++}), introduced for technical reasons, are actually valuable per sé in the field of TA.

2012 ACM Subject Classification Computing methodologies → Planning under uncertainty; Theory of computation → Quantitative automata

Keywords and phrases Timeline-based planning, timed automata, event-clock automata

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2019.34

1 Introduction

Timeline-based planning (TP for short) is a promising approach to real-time temporal planning and reasoning about executions under uncertainty [10, 11, 13, 14, 15, 16]. Compared to classical action-based temporal planning [17, 28], TP adopts a more declarative paradigm which focuses on the constraints that sequences of actions have to fulfil to reach a given goal. In TP, the planning domain is modeled as a set of independent, but interacting, components, each one identified by a *state variable*. The temporal behaviour of a single state variable (component) is described by a sequence of *tokens* (*timeline*) where each token specifies a value of the variable (state) and the period of time during which it takes that value. The overall temporal behaviour (set of timelines) is constrained by a set of *synchronization rules* that specify quantitative temporal requirements between the time events (start-time and end-time) of distinct tokens. Synchronization rules have a very simple format: either *trigger rules*, expressing invariants and response properties (for each token with a fixed state, called *trigger*, there exist some tokens satisfying some mutual temporal relations), or *trigger-less ones*, expressing goals (there exist some tokens satisfying some mutual temporal relations). Notice that the way in which requirements are specified by synchronization rules corresponds to the "freeze" mechanism in the well-known timed temporal logic TPTL [1], which uses the freeze quantifier to bind a variable to a specific temporal context (a token in the TP setting).

TP has been successfully exploited in a number of application domains, including space missions, constraint solving, and activity scheduling (see, e.g., [4, 9, 12, 18, 23, 25]). A systematic study of expressiveness and complexity of TP has been undertaken only very recently in both the discrete-time and the dense-time settings [5, 6, 20, 21].

In the discrete-time case, the TP problem is **EXPSPACE**-complete, and expressive enough to capture action-based temporal planning (see [20, 21]). Despite the simple format of synchronization rules, the shift to a dense-time domain dramatically increases expressiveness and complexity, depicting a scenario which resembles that of the well-known timed linear temporal logics MTL and TPTL, under a pointwise semantics, which are undecidable in the general setting [1, 26]. The TP problem in its full generality is indeed undecidable [6], and undecidability is caused by the high expressiveness of trigger rules (if only trigger-less rules are used, the TP problem is just **NP**-complete [8]). Decidability can be recovered by imposing suitable syntactic/semantic restrictions on the trigger rules. In particular, two restrictions are considered in [5]: (i) the first one limits the comparison to tokens whose start times follow the start time of the trigger (*future semantics of trigger rules*); (ii) the second one imposes that a non-trigger token can be referenced at most once in the timed constraints of a trigger rule (*simple trigger rules*). Under these two restrictions, the TP problem is decidable with a non-primitive recursive complexity [5] and can be solved by a reduction to model checking of Timed Automata (TA) [2] against MTL over *finite* timed words, the latter being a known decidable problem [27]. By removing either the future semantics or the simple trigger rule restrictions, the TP problem turns out to be undecidable [6, 7].

Our Contribution. In this paper, without imposing any syntactic restriction to the format of synchronization rules, we investigate an alternative semantics for the trigger rules in the dense-time setting, which turns out to be still quite expressive and relevant for practical applications, and has the main advantage of guaranteeing a reasonable computational complexity. In the standard semantics of trigger rules, if there are many occurrences of non-trigger tokens carrying the same specified value, say v, nothing forces the choice of a specific occurrence. For instance, if the trigger token represents a prompt and the v-valued token is a reaction to it, the chosen v-valued token is not guaranteed to be the first one after issuing the prompt. In a reactive context, one is in general interested in relating an issued prompt to the first response to it and not to an arbitrarily delayed one. A similar idea is exploited by Event-Clock Automata (ECA) [3], a well-known robust subclass of Timed Automata(TA) [2]. In ECA, each symbol a of the alphabet is associated with a *recorder* or *past clock*, recording (at the current time) the time elapsed since the last occurrence of a, and a *predicting* or *future clock*, measuring the time required for the next occurrence of a.

The alternative semantics of trigger rules is based on the minimality in the time distances of the start times of existentially quantified tokens in a trigger rule from the start time of the trigger token. In fact, the minimality constraint can be used to express two alternative semantics: the *weak minimal semantics*, which distinguishes minimality in the past, with respect to the trigger token, from minimality in the future, and the *strong minimal semantics*, which considers minimality over all the start times (both in the past and in the future). Surprisingly, this apparently small difference in the definitions of weak and strong minimal semantics leads to a dramatic difference in the complexity-theoretic characterization of the TP problem: while the TP problem under the *strong minimal semantics* is still undecidable, the TP problem under the *weak minimal semantics* turns out to be **PSPACE**-complete (which is the complexity of the emptiness problem for TA and ECA [2, 3]). **PSPACE** membership of the weak minimal TP problem is shown by a non-trivial exponential-time reduction to

non-emptiness of TA. To handle the trigger rules under the weak minimal semantics, we exploit, as an intermediate step in the reduction, a strictly more expressive extension of ECA, called ECA$^+$. This novel extension of ECA is obtained by allowing a larger class of atomic event-clock constraints, namely, diagonal constraints between clocks of the same polarity (past or future) and *sum constraints* between clocks of *opposite polarity*. In [19], these atomic constraints are used in *event-zones* to obtain symbolic forward and backward analysis semi-algorithms for ECA, which are not guaranteed to terminate. We show that, similar to ECA, ECA$^+$ are closed under language Boolean operations and can be translated in exponential time into equivalent TA with an exponential number of control states, but a linear number of clocks. We also investigate an extension of ECA$^+$, called ECA^{++}, where the polarity requirements in the diagonal and sum constraints are relaxed, and we show that the nonemptiness problem for such a class of automata is undecidable. We believe that these two original extensions of ECA, namely, ECA$^+$ and ECA^{++}, are interesting per sé, as they shed new light on the landscape of event-clock and timed automata.

The paper is organized as follows. In Section 2, we recall the TP framework and we introduce the strong and weak minimal semantics. In Section 3, we prove that the TP problem under the strong minimal semantics is still undecidable. In Section 4, we introduce and address complexity and expressiveness issues for ECA$^+$ and ECA^{++}. Moreover, by exploiting the results for ECA$^+$, we prove **PSPACE**-completeness of the weak minimal TP problem. Conclusions provide an assessment of the work done and outline future research themes.

2 The TP Problem

In this section, we recall the TP framework as presented in [15, 20] and we introduce the strong and weak minimal semantics. In TP, domain knowledge is encoded by a set of state variables, whose behaviour over time is described by transition functions and constrained by synchronization rules. We fix the following notation. Let $\mathbb{N}$ be the set of natural numbers, $\mathbb{R}_+$ the set of non-negative real numbers, and *Intv* the set of intervals in $\mathbb{R}_+$ whose endpoints are in $\mathbb{N} \cup \{\infty\}$. Given a finite word w over some alphabet (or, equivalently, a finite sequence of symbols), $|w|$ denotes the length of w and for all $0 \leq i < |w|$, $w(i)$ is the i-th letter of w.

▶ **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 $d \in D_x(v)$. For a token $t = (v, d)$, *value*(t) denotes the first component v of t. 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 $0 \leq i < n$. We associate to the i-th token $(0 \leq i \leq n)$ of the timeline π two punctual events: (i) the *start point* whose timestamp (*start time*), denoted by $\mathsf{s}(\pi, i)$, is 0 if $i = 0$, and is given by $\sum_{h=0}^{i-1} d_h$ otherwise, and (ii) the *end point* whose timestamp (*end time*), denoted by $\mathsf{e}(\pi, i)$, is given by $\mathsf{e}(\pi, i) := \mathsf{s}(\pi, i) + d_i$.

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.

▶ **Example 2.** Assume to have transactions (e.g database transactions) accessing a common shared resource A for read/write operations. The resource A can be unlocked (un_A), read_locked (r_l_A) or write_locked (w_l_A). A state variable $x_A = (V_A, T_A, D_A)$ with

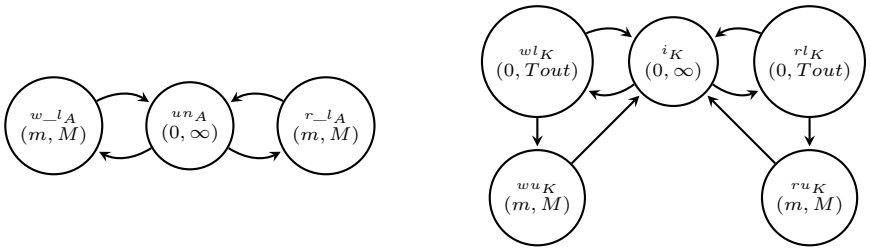


Figure 1 State variables x_A and x_K.

$V_A = \{un_A, r\_l_A, w\_l_A\}$ is used to describe the availability/locking of the resource during time. The value transition function T_A is represented as a graph in Figure 1 (left). Each node is labelled by a value v and by the constraint $D_A(v)$. The constants m and M are lower and upper bound, respectively, for the durations of read/write locking.
A state variable $x_K = (V_K, T_K, D_K)$, with K ranging over transaction names, describes the read/write locking requests issued by transaction K for the use of the resource A. A transaction can be idle (i_K), issuing a read or write lock for accessing the resource (rl_K or wl_K, resp.), or reading or writing the resource (ru_K or wu_K, resp.). Therefore we have $V_K = \{i_K, rl_K, wl_K, ru_K, wu_K\}$. An issued lock request can be accepted allowing the use of the resource or refused. There is a timeout $Tout$ for waiting the availability of the resource. The value transition and constraint functions T_K and D_K are depicted in Figure 1 (right).

Synchronization rules. Fix a finite set SV of state variables. 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 (point constraints) and on the difference between start/end-times of tokens (difference 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*.

An *atom* is either a clause of the form $ev(o) \in I$ (*point atom*), or of the form $ev(o) - ev'(o') \in I$ (*difference atom*), where $o, o' \in \Sigma$, $I \in Intv$, and $ev, ev' \in \{\mathsf{s}, \mathsf{e}\}$. Intuitively, an atom $ev(o) \in I$ asserts that the ev-time (i.e., the start-time if $ev = \mathsf{s}$, and the end-time otherwise) of the token referenced by o is in the interval I, while an atom $ev(o) - ev'(o') \in I$ requires that the difference between the ev-time and the ev'-time of the tokens referenced by o and o', respectively, is in I. Formally, an atom is evaluated with respect to a Σ-*assignment* λ_Π *for a given multi-timeline* Π *of* SV 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 $0 \leq i < |\pi|$ (intuitively, (π, i) represents the token of Π referenced by the name o). An atom $ev(o) \in I$ (resp., $ev(o) - ev'(o') \in I$) *is satisfied by* λ_Π if $ev(\lambda_\Pi(o)) \in I$ (resp., $ev(\lambda_\Pi(o)) - ev'(\lambda_\Pi(o')) \in I$).

An *existential statement* $\mathcal{E}$ 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*. Intuitively, the quantifier $o_i[x_i = v_i]$ binds the name o_i to some token in the timeline for variable x_i having value v_i. A Σ-assignment λ_Π for a multi-timeline Π of SV *satisfies* $\mathcal{E}$ if each atom in $\mathcal{C}$ is satisfied by λ_Π, and 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 which satisfies $\mathcal{E}$.

▶ **Definition 3.** *A* synchronization rule $\mathcal{R}$ *for the set* SV *of state variables has the forms* *(trigger rule)* $o_0[x_0 = v_0] \rightarrow \mathcal{E}_1 \vee \mathcal{E}_2 \vee \ldots \vee \mathcal{E}_k$, *(trigger-less rule)* $\top \rightarrow \mathcal{E}_1 \vee \mathcal{E}_2 \vee \ldots \vee \mathcal{E}_k$, *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 trigger rules, the quantifier* $o_0[x_0 = v_0]$ *is called* trigger, *and it is required that only* o_0 *may appear free in* $\mathcal{E}_i$ *(for* $i = 1, \ldots, k$*). For trigger-less rules, it is required that no token name appears free.*

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 having 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. Formally, the *standard semantics* of the synchronization rules is defined as follows. A multi-timeline Π of SV *satisfies* a *trigger-less rule* $\mathcal{R}$ of SV if Π satisfies some existential statement of $\mathcal{R}$. Π *satisfies* a *trigger rule* $\mathcal{R}$ of SV with trigger $o_0[x_0 = v_0]$ if for every position i of the timeline $\Pi(x_0)$ for x_0 such that $\Pi(x_0)(i) = (v_0, d)$, there is an existential statement $\mathcal{E}$ of $\mathcal{R}$ and a Σ-assignment λ_Π for Π such that $\lambda_\Pi(o_0) = (\Pi(x_0), i)$ and λ_Π satisfies $\mathcal{E}$. Trigger-less are usually exploited to express initial conditions or the goals of the problem. Trigger rules are useful to specify invariants and response requirements.

▶ **Example 4.** With reference to Example 2, we introduce synchronization rules to guarantee that the shared resource A is accessed in mutual exclusion during writing. We first define some shorthand for expressing conjunctions of atoms where o and $\overline{o}$ are token names:

- $during(o, \overline{o}) := \mathsf{s}(\overline{o}) - \mathsf{s}(o) \in [0, \infty) \wedge \mathsf{e}(o) - \mathsf{e}(\overline{o}) \in [0, \infty)$ requires that the token referenced by $\overline{o}$ occurs during the token referenced by o;
- $overlap(o, \overline{o}) := \mathsf{e}(o) - \mathsf{s}(\overline{o}) \in (0, \infty) \wedge \bigwedge_{ev \in \{\mathsf{s},\mathsf{e}\}} ev(\overline{o}) - ev(o) \in (0, \infty)$ asserts that the $\overline{o}$'s token does not start before the o's token and crosses the end point of the o's token.

The following first pair of trigger-less rules fix the initial conditions: the resource A is initially unlocked and each transaction K is idle. The next trigger rule ensures that when a transaction K reads the resource A, the resource is locked for reading. The last trigger rule requires that when K writes A, there is a write locking token of A having the same temporal window as the K-token.

- $\top \rightarrow \exists o[x_A = un_A].\, \mathsf{s}(o) \in [0, 0]$ and $\top \rightarrow \exists o[x_K = i_K].\, \mathsf{s}(o) \in [0, 0]$;
- $o_0[x_K = ru_K] \rightarrow \exists o[x_A = r\_l_A].during(o, o_0)$;
- $o_0[x_K = wu_K] \rightarrow \exists o[x_A = w\_l_A].during(o, o_0) \wedge during(o_0, o)$.

The two trigger rules above ensure the mutual exclusion among reads and writes of the resource A by the same transaction K. The following rules are added to guarantee mutual exclusion when distinct transactions K and H write A, i.e. we have to ensure that K and H do not feature tokens of value wu_K and wu_H, respectively, with the same temporal window.

$$o_0[x_K = wu_K] \rightarrow \bigvee_{s \in \{i_H, wl_H, rl_H\}} \big(\exists o[x_H = s].during(o, o_0) \vee \exists o[x_H = s].during(o_0, o) \vee \exists o[x_H = s].overlap(o_0, o) \vee \exists o[x_H = s].overlap(o, o_0)\big).$$

Minimal semantics of trigger rules. In the following we define the variant of the semantics for trigger rules newly proposed and investigated in this paper. It is obtained from the standard semantics by additionally requiring that the given Σ-assignment λ_Π selects for each (existential) quantifier $o[x = v]$ a token for variable x with value v whose start point has a minimal time distance from the start point of the trigger. Actually, the constraint of minimality can be used to express two alternative semantics: the *weak minimal semantics* which distinguishes minimality in the past (w.r.t. the trigger token) from the minimality in the future, and the *strong minimal semantics* which considers minimality over all the start times (both in the past and in the future) of the tokens for a variable x and x-value v.

▶ **Definition 5.** *Let $o_0 \in \Sigma$. A Σ-assignment λ_Π for a multi-timeline Π of SV is* weakly minimal *w.r.t. o_0 if for each $o \in \Sigma$ with $\lambda_\Pi(o) = (\pi, i)$, the following holds:*

- minimality in the past: *if $\mathsf{s}(\pi, i) \leq \mathsf{s}(\lambda_\Pi(o_0))$ then there is no position ℓ along the timeline π such that $value(\pi(i)) = value(\pi(\ell))$ and $\mathsf{s}(\pi, i) < \mathsf{s}(\pi, \ell) \leq \mathsf{s}(\lambda_\Pi(o_0))$.*
- minimality in the future: *if $\mathsf{s}(\pi, i) \geq \mathsf{s}(\lambda_\Pi(o_0))$ then there is no position ℓ along the timeline π such that $value(\pi(i)) = value(\pi(\ell))$ and $\mathsf{s}(\pi, i) > \mathsf{s}(\pi, \ell) \geq \mathsf{s}(\lambda_\Pi(o_0))$.*

A Σ-assignment λ_Π for Π is strongly minimal *w.r.t. o_0 if for each $o \in \Sigma$ with $\lambda_\Pi(o) = (\pi, i)$, there is no position ℓ along the timeline π such that $value(\pi(i)) = value(\pi(\ell))$ and $|\mathsf{s}(\pi, \ell) - \mathsf{s}(\lambda_\Pi(o_0))| < |\mathsf{s}(\pi, i) - \mathsf{s}(\lambda_\Pi(o_0))|$.*

The weak minimal (resp., strong minimal) semantics of the trigger rules is obtained from the standard one by imposing that the considered Σ-assignment λ_Π is weakly minimal (resp., strongly minimal) w.r.t. the trigger token o_0.

Note that we consider start points of tokens for expressing minimality. Equivalent semantics can be obtained by considering end points of tokens instead.

With reference to Example 4, we observe that, due to the constraints *during* and *overlap*, the weak minimal semantics for the considered trigger rules corresponds to the standard one.

Domains and plans. A TP domain $\mathcal{D} = (SV, R)$ is specified by a finite set SV of state variables and a finite set R of synchronization rules modeling their admissible behaviors. A *weak* (resp., *strong*) *minimal plan* of $\mathcal{D}$ is a multi-timeline of SV satisfying all the rules in R under the weak (resp., strong) minimal semantics of trigger rules. The *weak* (resp. *strong*) *minimal TP problem* is checking given a domain $\mathcal{D}$, whether there is a weak (resp. strong) minimal plan of $\mathcal{D}$. We also consider the *discrete-time versions* of the previous problems, where the durations of the tokens in a plan are restricted to be natural numbers.

▶ Assumption 6 (*Strict time monotonicity*). In the following, for simplifying the technical presentation of some results, without loss of generality, we assume that given a state variable $x = (V_x, T_x, D_x)$, the duration of a token for x is never zero, i.e., for each $v \in V_x$, $0 \notin D_x(v)$.

3 Undecidability of the strong minimal TP problem

In this section, we establish the negative result for the strong minimal semantics by a polynomial-time reduction from the *halting problem for Minsky 2-counter machines* [24]. The key feature in the reduction is the possibility to express for a given value v, a temporal equidistance requirement w.r.t. the start point of the trigger token for the start points of the last token before the trigger with value v and the first token after the trigger with value v.

▶ **Theorem 7.** *The strong minimal TP problem is undecidable even in the discrete-time setting.*

Proof. A nondeterministic Minsky 2-counter 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 $L = \{\mathsf{inc}, \mathsf{dec}, \mathsf{zero\_test}\} \times \{1, 2\}$. For a transition $\delta = (q, op, q') \in \Delta$, we define $from(\delta) := q$, $op(\delta) := op$, and $to(\delta) := q'$. Without loss of generality we assume that:

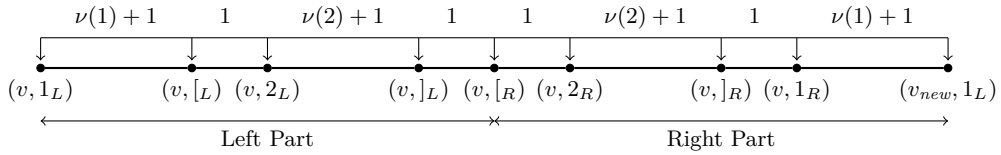
- for each transition $\delta \in \Delta$, $from(\delta) \neq q_{halt}$ and $to(\delta) \neq q_{init}$, and
- there is exactly one transition in Δ, denoted δ_{init}, having as source location q_{init}.

An M-configuration is a pair (q, ν) consisting of a location $q \in Q$ and a counter valuation $\nu : \{1, 2\} \to \mathbb{N}$. A computation of M is a non-empty *finite* sequence $(q_1, \nu_1), \ldots, (q_k, \nu_k)$ of configurations such that for all $1 \leq i < k$, there is some instruction $op_i = (tag_i, c_i) \in L$, so

that $(q_i, op_i, q_{i+1}) \in \Delta$ and: (i) $\nu_{i+1}(c) = \nu_i(c)$ if $c \neq c_i$; (ii) $\nu_{i+1}(c_i) = \nu_i(c_i) + 1$ if $tag_i = \mathsf{inc}$; (iii) $\nu_{i+1}(c_i) = \nu_i(c_i) - 1$ and $\nu_i(c_i) > 0$ if $tag_i = \mathsf{dec}$; and (iv) $\nu_{i+1}(c_i) = \nu_i(c_i) = 0$ if $tag_i = \mathsf{zero\_test}$. The halting problem is to decide whether for a machine M, 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) (it was proved to be undecidable in [24]). To prove Theorem 7 we construct a TP instance $\mathcal{D}_M = (SV_M, R_M)$ such that M halts *iff* there exists a strong minimal discrete-time plan for $\mathcal{D}_M$.

We exploit a state variable x_M for encoding the evolution of the machine M and additional state variables for checking that the values of counters in the timeline for x_M are correctly updated. The domain V_M of the state variable x_M is given by $V_M := V_\Delta \times \{1_L, 1_R, 2_L, 2_R, [_L,]_L, [_R,]_R\}$ where V_Δ is the set of pairs $(\delta'_\perp, \delta)$, where $\delta'_\perp \in \Delta \cup \{\perp\}$, $\delta \in \Delta$, and $to(\delta'_\perp) = from(\delta)$ if $\delta'_\perp \neq \perp$, and $\delta = \delta_{init}$ otherwise. Intuitively, in the pair $(\delta'_\perp, \delta)$, δ represents the transition currently taken by M from the current non-halting configuration C, while $\delta'_\perp$ is $\perp$ if C is the initial configuration, and δ' represents the transition exploited by M in the previous computational step otherwise.

A configuration $C = (q, \nu)$ of M is encoded by the timelines π_C (*configuration codes*) of length 9 for the state variable x_M illustrated in the following figure, where $v \in V_\Delta$ (called V_Δ*-value of* π_C) is of the form $(\delta'_\perp, \delta)$ such that $from(\delta) = q$. Note that the configuration



code π_C is subdivided in two parts. In the left part (resp., right part), the encoding of counter 1 (resp., 2) precedes the encoding of counter 2 (resp., 1). The value $\nu(1)$ of counter 1 is encoded by the duration, which is $\nu(1) + 1$, of the *counter token* with value marked by 1_L in the left part, and the *counter token* with value marked by 1_R in the right part, and similarly for counter 2. The four tokens with values marked by $[_L$, $]_L$, $[_R$, and $]_R$, respectively, are called *tagged* tokens and their duration is always 1: they are used to check by trigger rules (under the strong minimal semantics) that increment and decrement M-instructions are correctly encoded. Moreover, we require that the configuration code π_C satisfies the following additional requirement (V_Δ*-requirement*), with $v = (\delta'_\perp, \delta)$ and $\delta = (q, op, q')$:

- $v_{new} = v$ if $to(\delta) = q_{halt}$, and v_{new} is of the form (δ, δ'') otherwise (*consecution*);
- if $\delta = \delta_{init}$ then the counter tokens have duration 1;
- if $op = (\mathsf{dec}, c)$ (resp., $op = (\mathsf{zero\_test}, c)$), then the durations of the counter tokens with values (v, c_L) and (v, c_R) are greater than 1 (resp., are equal to 1).

A *pseudo-configuration code* is defined as a configuration code but the durations of the counter tokens are arbitrary with the restriction that the V_Δ-requirement is fulfilled.

By construction and the assumption on M, we can easily define a *trigger-less* rule $\mathcal{R}_{init,halt}$ and define the transition and constraint function of x_M in such a way that the timelines of x_M satisfying $\mathcal{R}_{init,halt}$ (called *pseudo-computation codes*) are the sequences of the form $\pi_0 \cdots \pi_n$ such that: (i) $\pi_i \cdot \pi_{i+1}(0)$ and π_n are *pseudo-configuration codes* for all $0 \leq i < n$, (ii) if $n > 0$ (resp., $n = 0$), the V_Δ-value of $\pi_0 \cdot \pi_1(0)$ (resp., π_0) is $(\perp, \delta_{init})$ (*initialization*), and (iii) the V_Δ-value of π_n is of the form $(\delta'_\perp, \delta)$ such that $to(\delta) = q_{halt}$ (*halting*).

We now consider the crucial part of the reduction which has to guarantee that along a pseudo-computation code the counters are correctly encoded (i.e., the durations of the left and right tokens for each counter in a pseudo-configuration code coincide) and are updated

accordingly to the M-instructions. Here, we focus on the increment instruction $(\mathsf{inc}, 1)$ for counter 1. For this, we exploit trigger rules in conjunction with an additional state variable $x_{(\mathsf{inc},1)}$ having domain $V_{check} := \{check_1, check_2, trigger, \bot\}$ and capturing the timelines π such that the duration of each token is at least 1 and the untimed part of π is an arbitrary non-empty word over V_{check}. Let π_M be a pseudo-computation code, π_C a non-initial pseudo-configuration code of π_M with V_Δ-value $(\delta'_\bot, \delta)$ such that $\delta'_\bot \neq \bot$ and $op(\delta'_\bot) = (\mathsf{inc}, 1)$ (we denote by $V_{(\mathsf{inc},1)}$ the set of such V_Δ-values), and π_{C_p} the pseudo-configuration code preceding π_C along π_M. We need to ensure that the duration of the token for counter 1 (resp., 2) in the left part of π_C is one plus the duration (resp., is the duration) of the token for counter 1 (resp., 2) in the right part of π_{C_p}. The proposed encoding ensures that the previous requirement holds *iff* for each token tk_{1_L} of π_M with value in $V_{(\mathsf{inc},1)} \times \{1_L\}$, the following holds ($(\mathsf{inc}, 1)$-*requirement*): for the last token marked by $]_R$ (resp., $[_R$) preceding tk_{1_L} and the first token marked by $[_L$ (resp., $]_L$) following tk_{1_L}, their start points have the same time distance from the start point of tk_{1_L}. Then, in order to enforce the $(\mathsf{inc}, 1)$-requirement, we first require that:

(*) the timelines π_M and $\pi_{(\mathsf{inc},1)}$ of variables x_M and $x_{(\mathsf{inc},1)}$, respectively, are synchronized, i.e., they have the same length and for each position i, the start-times of the ith tokens of π_M and $\pi_{(\mathsf{inc},1)}$ coincide;

(**) for the timeline $\pi_{(\mathsf{inc},1)}$ for variable $x_{(\mathsf{inc},1)}$ (synchronized with π_M), it holds that a token has value *trigger* (resp., has value $check_1$, resp., has value $check_2$) iff the associated token along π_M has a value in $V_{(\mathsf{inc},1)} \times \{1_L\}$ (resp., in $V_\Delta \times \{]_R, [_L\}$, resp., in $V_\Delta \times \{[_R,]_L\}$).

Since the duration of a token is not zero, the previous two requirements (*)–(**) can be easily expressed by trigger rules under the strong minimal semantics. Finally, we require that for each trigger-token $tk_{trigger}$ along the timeline $\pi_{(\mathsf{inc},1)}$ for $x_{(\mathsf{inc},1)}$ and for each $\ell = 1, 2$, the start points of the last $check_\ell$-token of $\pi_{(\mathsf{inc},1)}$ preceding $tk_{trigger}$ and the first $check_\ell$-token following $tk_{trigger}$ have the same time distance from the start point of $tk_{trigger}$. By the strong minimal semantics, this requirement can be expressed by the following two trigger rules, where $op = (\mathsf{inc}, 1)$, which ensure that for the $check_\ell$-tokens ($\ell = 1, 2$) whose start points have the smallest time distance from the start point of the trigger, there is one preceding the trigger and one following the trigger:

$$o[x_{op} = trigger] \rightarrow \exists o_1[x_{op} = check_1]\exists o_2[x_{op} = check_2].\ \textstyle\bigwedge_{\ell=1,2} \mathsf{s}(o) - \mathsf{s}(o_\ell) \in [0, \infty)$$
$$o[x_{op} = trigger] \rightarrow \exists o_1[x_{op} = check_1]\exists o_2[x_{op} = check_2].\ \textstyle\bigwedge_{\ell=1,2} \mathsf{s}(o_\ell) - \mathsf{s}(o) \in [0, \infty). \quad \blacktriangleleft$$

4 Decidability of the weak minimal TP problem

In this section, we show that the weak minimal TP problem is decidable and **PSPACE**-complete. The upper bound is obtained by an exponential-time reduction to nonemptiness of Timed Automata (TA) [2]. In order to handle the trigger rules under the weak minimal semantics, we exploit as an intermediate step an extension, denoted by ECA^+, of the known class of *Event Clock Automata* (ECA) [3]. The rest of the section is organized is follows. We first shortly recall the class of Timed Automata (TA) [2]. Then, in Subsection 4.1, we introduce and address complexity and expressiveness issues for the newly introduced class of ECA^+. Finally, in Subsection 4.2, we solve the weak minimal TP problem.

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 untimed word $a_0 \cdots a_n$ and $\tau = \tau_0 \cdots \tau_n$. A *timed language* over Σ is a set of timed words over Σ.

A TA over Σ is a tuple $\mathcal{A} = (\Sigma, Q, Q_0, C, \Delta, F)$, where Q is a finite set of (control) states, $Q_0 \subseteq Q$ is the set of initial states, C is a finite set of *clocks*, $F \subseteq Q$ is the set of accepting states, and Δ is the finite set of transitions (q, a, θ, Res, q') such that $q, q' \in Q$, $a \in \Sigma$, $Res \subseteq C$ is a clock reset set, and θ is a *clock constraint* over C, that is a conjunction of atomic formulas of the form $c \sim n$ (*simple constraints*) with $c \in C$, $\sim \in \{<, \leq, \geq, >\}$, and $n \in \mathbb{N}$. We denote by $K_{\mathcal{A}}$ the maximal constant used in the clock constraints of $\mathcal{A}$. Intuitively, in a TA $\mathcal{A}$, 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 $\mathcal{A}$ is a pair (q, val), where $q \in Q$ and $val : C \to \mathbb{R}_+$ is a clock valuation for C assigning to each clock a non-negative real number. For $t \in \mathbb{R}_+$ and a reset set $Res \subseteq C$, the valuations $(val + t)$ and $val[Res]$ are defined as: 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. For a clock constraint θ, val satisfies θ, written $val \models \theta$, if for each conjunct $c \sim n$ of θ, $val(c) \sim n$.

A run r of $\mathcal{A}$ 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 an initial configuration (q_0, val_0), with $q_0 \in Q_0$ and $val_0(c) = 0$ for all $c \in C$, and such that for all $0 \leq i \leq n$ (we let $\tau_{-1} = 0$): $(q_i, a_i, \theta, Res, q_{i+1}) \in \Delta$ for some constraint θ 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* $\mathcal{L}_T(\mathcal{A})$ of $\mathcal{A}$ is the set of timed words w over Σ s.t. there is an accepting run of $\mathcal{A}$ over w.

4.1 Extended Event-clock Automata

In this section, we introduce an extension, denoted by ECA$^+$, of *Event Clock Automata* (ECA) [3]. In ECA, clocks have a predefined association with the input alphabet symbols and their values refer to the time distances from previous and next occurrences of input symbols. ECA$^+$ extend ECA by allowing a larger class of atomic event-clock constraints, namely diagonal constraints (alias difference constraints) between clocks of the same polarity and *sum constraints* between clocks of *opposite polarity*. Additionally, we consider the extension of ECA$^+$, denoted by ECA^{++}, where the polarity requirements in the diagonal and sum constraints are relaxed. We show that ECA$^+$ are more expressive than ECA and that they can be translated in exponential time into equivalent TA. Differently from ECA$^+$, ECA^{++} are a very powerful formalism having an undecidable nonemptiness problem.

Here, we adopt a propositional-based approach where the input alphabet is given by $2^{\mathcal{P}}$ for a given set of atomic propositions. The set $C_{\mathcal{P}}$ of event clocks associated with $\mathcal{P}$ is given by $C_{\mathcal{P}} := \bigcup_{p \in \mathcal{P}} \{\overleftarrow{c}_p, \overrightarrow{c}_p\}$. Thus, for each proposition $p \in \mathcal{P}$, there are two event clocks: the *event-recording or past clock* $\overleftarrow{c}_p$ which records the time elapsed since the last occurrence of p in the input word (if any), and the *event-predicting or future clock* $\overrightarrow{c}_p$ which provides the time required to the next occurrence of p (if any). A special value $\perp$ is exploited to denote the absence of a past (resp., future) occurrence of proposition p. Formally, the values of the event clocks at a position i of a timed word w can be deterministically determined as follows.

▶ **Definition 8** (Deterministic clock valuations). *An* event-clock valuation *is a mapping* $val : C_{\mathcal{P}} \mapsto \mathbb{R}_+ \cup \{\perp\}$*, assigning to each event clock a value in* $\mathbb{R}_+ \cup \{\perp\}$*. For a timed word* $w = (\sigma, \tau)$ *over* $2^{\mathcal{P}}$ *and a position* $0 \leq i < |w|$*, the* event-clock valuation val_i^w*, specifying the values of the event clocks at position* i *along* w*, is defined as follows for each* $p \in \mathcal{P}$*:*

$$val_i^w(\overleftarrow{c_p}) = \begin{cases} \tau_i - \tau_\ell & \text{if there exists the unique } 0 \leq \ell < i : p \in \sigma(\ell) \text{ and} \\ & \quad \forall k : \ell < k < i \Rightarrow p \notin \sigma(k) \\ \bot & \text{otherwise} \end{cases}$$

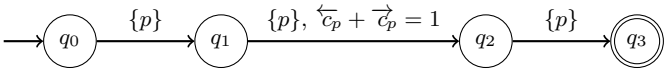
$$val_i^w(\overrightarrow{c_p}) = \begin{cases} \tau_\ell - \tau_i & \text{if there exists the unique } i < \ell < |\sigma| : p \in \sigma(\ell) \text{ and} \\ & \quad \forall k : i < k < \ell \Rightarrow p \notin \sigma(k) \\ \bot & \text{otherwise} \end{cases}$$

An ECA^+ over $2^{\mathcal{P}}$ is a tuple $\mathcal{A} = (2^{\mathcal{P}}, Q, Q_0, C_{\mathcal{P}}, \Delta, F)$, where Q is a finite set of states, $Q_0 \subseteq Q$ is a set of initial states, $F \subseteq Q$ is a set of accepting states, and Δ is a finite set of transitions (q, a, θ, q'), where $q, q' \in Q$, $a \in 2^{\mathcal{P}}$, and θ is an ECA^+ *event-clock constraint* that is a conjunction of atomic formulas of the following forms, where $p, p' \in \mathcal{P}$, $\sim \in \{<, \leq, \geq, >\}$, and $n_\perp \in \mathbb{N} \cup \{\bot\}$: (i) $\overleftarrow{c_p} \sim n_\perp$ or $\overrightarrow{c_p} \sim n_\perp$ (*simple constraints*); or (ii) $\overleftarrow{c_p} - \overleftarrow{c_{p'}} \sim n_\perp$ or $\overrightarrow{c_p} - \overrightarrow{c_{p'}} \sim n_\perp$ (*diagonal constraints* between event clocks of the same polarity); or (iii) $\overleftarrow{c_p} + \overrightarrow{c_{p'}} \sim n_\perp$ (*sum constraints* between event clocks of opposite polarity). We denote by $K_{\mathcal{A}}$ the maximal constant used in the event-clock constraints of $\mathcal{A}$. An ECA [3] is an ECA^+ which does *not* use diagonal and sum constraints. We also consider the extension of ECA^+, denoted by ECA^{++}, where the transition guards also exploit as conjuncts diagonal (resp., sum) constraints over event clocks of opposite polarity (resp., of the same polarity).

Let us fix an event-clock valuation *val*. We extend in the natural way the valuation *val* to differences (resp., sums) of event clocks: for all $c, c' \in C_{\mathcal{P}}$, $val(c - c') = val(c) - val(c')$ and $val(c + c') = val(c) + val(c')$ where each sum or difference involving $\bot$ evaluates to $\bot$. Given an event-clock constraint θ, *val* satisfies θ, written $val \models \theta$, if for each conjunct $t \sim n_\perp$ of θ, either (i) $val(t) \neq \bot$, $n_\perp \neq \bot$, and $val(t) \sim n_\perp$, or (ii) $val(t) = \bot$, $n_\perp = \bot$, and $\sim \in \{\leq, \geq\}$.

A run π of an ECA^+ (resp., ECA^{++}) $\mathcal{A}$ over a timed word $w = (\sigma, \tau)$ is a sequence of states $\pi = q_0, \ldots, q_{|w|}$ such that $q_0 \in Q_0$ and for all $0 \leq i < |w|$, $(q_i, \sigma(i), \theta, q_{i+1}) \in \Delta$ for some constraint θ such that $val_i^w \models \theta$. The run π is *accepting* if $q_{|w|} \in F$. The *timed language* $\mathcal{L}_T(\mathcal{A})$ of $\mathcal{A}$ is the set of timed words w over $2^{\mathcal{P}}$ s.t. there is an accepting run of $\mathcal{A}$ on w.

As an example, let us consider the ECA^+ $\mathcal{A}_p$, depicted below, whose set $\mathcal{P}$ of atomic propositions consists of a unique proposition p. Evidently, $\mathcal{A}_p$ accepts the set of timed words



w of length 3 of the form $(\{p\}, \tau_0), (\{p\}, \tau_1), (\{p\}, \tau_2)$ such that the time difference between the first and last symbol is 1, i.e. $\tau_2 - \tau_0 = 1$. One can easily show that there is no ECA accepting $\mathcal{L}_T(\mathcal{A}_p)$. Hence, we obtain the following result.

▶ **Theorem 9.** *For a proposition p, there is a timed language over $2^{\{p\}}$ which is definable by ECA^+ but is* not *definable by ECA. Hence, ECA^+ are strictly more expressive than ECA.*

Like ECA [3], we show that the class of timed languages accepted by ECA^+ (resp., ECA^{++}) is closed under Boolean operations. The closure under complementation is crucially based on the fact that event-clock values are determined solely by the input word.

▶ **Theorem 10** (Closure properties)**.** *Given two ECA^+ (resp., ECA^{++}) $\mathcal{A}$ and $\mathcal{A}'$ over $2^{\mathcal{P}}$ with n and n' states, respectively, one can construct ECA^+ (resp., ECA^{++}) $\mathcal{A}_\cup$, $\mathcal{A}_\cap$, and $\mathcal{A}_c$ such that: (i) $\mathcal{A}_\cup$ (resp., $\mathcal{A}_\cap$) accepts $\mathcal{L}_T(\mathcal{A}) \cup \mathcal{L}_T(\mathcal{A}')$ (resp., $\mathcal{L}_T(\mathcal{A}) \cap \mathcal{L}_T(\mathcal{A}')$) and has $n + n'$ (resp., nn') states and greatest constant $\max(K_{\mathcal{A}}, K_{\mathcal{A}'})$; and (ii) $\mathcal{A}_c$ accepts the complement of $\mathcal{L}_T(\mathcal{A})$ and has $2^{O(n)}$ states and greatest constant $K_{\mathcal{A}}$.*

It is known that ECA can be translated in singly exponential time into equivalent TA [3]. We generalize this result to the class of ECA^+.

▶ **Theorem 11** (From ECA$^+$ to TA). *Given an ECA$^+$ $\mathcal{A}$ over $2^{\mathcal{P}}$, one can construct in exponential time a TA $\mathcal{A}'$ over $2^{\mathcal{P}}$ such that $\mathcal{L}_T(\mathcal{A}') = \mathcal{L}_T(\mathcal{A})$ and $K_{\mathcal{A}'} = K_{\mathcal{A}}$. Moreover, $\mathcal{A}'$ has $n \cdot 2^{O(p)}$ states and $O(p)$ clocks, where n is the number of $\mathcal{A}$-states and p is the number of event-clock atomic constraints used by $\mathcal{A}$.*

Sketched proof. Let $\mathcal{A} = (2^{\mathcal{P}}, Q, Q_0, C_{\mathcal{P}}, \Delta, F)$ be an ECA$^+$ over $2^{\mathcal{P}}$. The TA $\mathcal{A}'$ accepting $\mathcal{L}_T(\mathcal{A})$ is essentially obtained from $\mathcal{A}$ by replacing each atomic event-clock constraint of $\mathcal{A}$ with a set of standard clocks together with associated reset operations and clock constraints. To remove simple event-clock constraints of $\mathcal{A}$, we proceed as in [3]. Here, we focus on the removal of diagonal constraints over event-predicting clocks. Let us consider a diagonal predicting clock constraint $\eta : \overrightarrow{c}_p - \overrightarrow{c}_{p'} \sim n_\perp$ of $\mathcal{A}$ where $n_\perp \in \mathbb{N} \cup \{\perp\}$. We consider the case $n_\perp \neq \perp$ (the other case being simpler). For handling the constraint η, the TA $\mathcal{A}'$ exploits the fresh standard clock c_η and in case $n_\perp = 0$ and $\sim$ is $\geq$, the additional fresh standard clock $\hat{c}_\eta$. The first (resp., second) clock is reset *only if* proposition p' (resp., p) occurs in the current input symbol. Assume that the prediction η is done by $\mathcal{A}$ at position i of the input word for the first time. Then, the simulating TA $\mathcal{A}'$ carries the obligation η in its control state in order to check that there are next positions where p and p' occur and $\tau_p - \tau_{p'} \sim n_\perp$ holds, where τ_p (resp., $\tau_{p'}$) is the timestamp associated with the first next position $i_p > i$ (resp., $i_{p'} > i$) where p (resp., p') occurs. Note that all the predictions η done by $\mathcal{A}$ before positions i_p and $i_{p'}$ correspond to the same obligation. First, assume that the first next position $i_{p'} > i$ where p' occurs strictly precedes position i_p. In this case, on reading position $i_{p'}$, $\mathcal{A}'$ resets the clock c_η and replaces the old obligation η with the updated obligation (η, p') in order to check that the constraint $c_\eta \sim n_\perp$ holds when the next p occurs (i.e., at position i_p). If a new prediction η is done at a position $j_{new} \geq i_{p'}$ strictly preceding i_p, the fresh obligation η is carried in the control state together with the obligation (η, p'). We distinguish two cases:

- p' occurs in some position strictly following j_{new} and strictly preceding i_p. Let j' be the smallest of such positions. On reading position j', $\mathcal{A}'$ replaces the old obligations η and (η, p') with (η, p') and resets the clock c_η *iff* η is a lower bound constraint, i.e., $\sim \in \{>, \geq\}$. This is safe since if η is a lower bound, then the fulfillment of prediction η at j_{new} guarantees the fulfillment of prediction η at position i. Vice versa, if η is an upper bound, then the fulfillment of prediction η at i guarantees the fulfillment of prediction η' at position j_{new}. Thus, when η is a lower bound, new obligations (η, p') rewrite the old ones, while when η is an upper bound, new obligations (η, p') are ignored.
- there is no position strictly following j_{new} and strictly preceding i_p, where p' occurs. In this case, when i_p is read, the old obligation η is replaced with the obligation (η, p) unless p' occurs at position i_p (in the latter case, $\mathcal{A}'$ simply checks that $0 \sim n_\perp$).

In both the cases on reading position i_p, the constraint $c_\eta \sim n_\perp$ is checked and the obligation (η, p') is discarded. The case where $i_{p'} = i_p$ is trivial (on reading position i, $\mathcal{A}'$ checks that $0 \sim n_\perp$ holds). Finally, assume that i_p strictly precedes i'_p. The cases where either $c \neq 0$ or $\sim$ is distinct from $\geq$ are easy to handle, since in these cases if η is a lower bound (resp., upper bound), then the prediction η done at position i is not satisfied (resp., is satisfied). Thus, we focus on the case where $c = 0$ and $\sim$ is $\geq$. On reading position i_p, the clock $\hat{c}_\eta$ is reset and the old obligation η is replaced with the updated obligation (η, p) in order to check that the constraint $\hat{c}_\eta = 0$ holds when the next p' occurs (i.e., at position $i_{p'}$). In this case, new obligations (η, p) occurring before position $i_{p'}$ are ignored, i.e., the clock $\hat{c}_\eta$ is not reset at such positions. Finally, in order to ensure that raised obligations about η are eventually checked, the accepting states of $\mathcal{A}'$ do not contain such obligations. ◀

Theorem 11 cannot be extended to the class of ECA^{++}. In fact we show that for these automata, the nonemptiness problem is undecidable. The undecidability proof is similar to the one for the strong minimal TP problem.

▶ **Theorem 12.** *The nonemptiness problem of ECA^{++} is undecidable even for the subclass of ECA^{++} which use only simple atomic event-clock constraints and diagonal constraints over event clocks of opposite polarity of the form $\overleftarrow{c_p} - \overrightarrow{c_{p'}} = 0$.*

4.2 Solving the weak minimal TP problem

In this section, by exploiting the results of Section 4.1, we establish the following result, where for a TP domain $\mathcal{D} = (SV, R)$, the *maximal constant* $K_{\mathcal{D}}$ of $\mathcal{D}$ is the greatest integer occurring in the atoms of R and in the constraint functions of the variables in SV.

▶ **Theorem 13.** *Given a TP domain $\mathcal{D} = (SV, R)$, one can build in exponential time a TA $\mathcal{A}_{\mathcal{D}}$ with $2^{O(N+\sum_{x\in SV}|V_x|)}$ states, $O(N+|SV|)$ clocks, and maximal constant $O(K_{\mathcal{D}})$, where N is the overall number of quantifiers and atoms in the rules of R, such that $\mathcal{L}_T(\mathcal{A}_{\mathcal{D}}) \neq \emptyset$ iff there is a* weak minimal plan *of $\mathcal{D}$. Moreover, the weak minimal TP problem is* **PSPACE***-complete.*

Sketched proof. For each $x \in SV$, let $x = (V_x, T_x, D_x)$. In order to prove the first part of Theorem 13, we first define an encoding of the multi-timelines of SV by means of timed words over $2^{\mathcal{P}}$ for the set $\mathcal{P}$ of propositions given by $\{\mathsf{init}\} \cup \bigcup_{x\in SV} \mathcal{P}_x$ where for each $x \in SV$, $\mathcal{P}_x = \{x\} \times V_x \times \{\mathsf{s}, \mathsf{e}\} \times \{0, 1\}$. We use the propositions in $\mathcal{P}_x$ to encode the tokens tk along a timeline for x: the start point and end point of tk are specified by propositions (x, v, s, b) and (x, v, e, b), respectively, where $b \in \{0, 1\}$ and v is the value of tk. The meaning of the bit $b \in \{0, 1\}$ is explained below. The additional proposition $\mathsf{init} \in \mathcal{P}$ is used to mark the first point of a multi-timeline code in order to check point atoms of trigger rules by ECA^{+} event-clock constraints. A *code for a timeline for x* is a timed word w over $2^{\mathcal{P}_x}$ of the form

$$w = (\{(x, v_0, \mathsf{s}, b_0)\}, \tau_0), (\{(x, v_0, \mathsf{e}, b_0)\}, \tau_1) \cdots (\{(x, v_n, \mathsf{s}, b_n)\}, \tau_n), (\{(x, v_n, \mathsf{e}, b_n)\}, \tau_{n+1})$$

such that for all $0 \leq i \leq n$: (i) $v_{i+1} \in T_x(v_i)$ if $i < n$; (ii) $\tau_0 = 0$ and $\tau_{i+1} - \tau_i \in D_x(v_i)$; (iii) let ℓ_i be the greatest index $0 \leq j < i$ such that $v_j = v_i$ if such an index exists, and let $\ell_i := \perp$ otherwise. Then, $b_i = (b_{\ell_i} + 1) \mod 2$ if $\ell_i \neq \perp$, and $b_i = 0$ otherwise. Intuitively, for each value $v \in V_x$ occurring along w, the associated bit acts as a modulo 2 counter which is incremented at each visit of v along w (in the handling of the trigger rules under the weak minimal semantics, it is used by ECA^{+} event-clock constraints to reference the end-event of a token whose start-event (x, v, s) is the first occurrence of (x, v, s, b) for some $b \in \{0, 1\}$ after the current input position). The timed word w encodes the timeline for x of length $n + 1$ given by $\pi = (v_0, \tau_1 - \tau_0) \dots (v_n, \tau_{n+1} - \tau_n)$. Note that since the duration of a token is not zero, we have that $\tau_{i+1} > \tau_i$ for all $0 \leq i \leq n$. A *code for a multi-timeline for SV* is obtained by merging different timelines (one for each variable $x \in SV$), i.e., it is a non-empty timed word w over $2^{\mathcal{P}}$ of the form $w = (P_0, \tau_0) \cdots (P_n, \tau_n)$ such that: (i) for all $x \in SV$, the timed word obtained from $(P_0 \cap \mathcal{P}_x, \tau_0) \cdots (P_n \cap \mathcal{P}_x, \tau_n)$ by removing the pairs $(\emptyset, \tau_i)$ is a code of a timeline for x; (ii) $\mathsf{init} \in P_0$, $\mathsf{init} \notin P_i$ for all $1 \leq i \leq n$, and $P_0 \cap \mathcal{P}_x \neq \emptyset$ for all $x \in SV$.

The trigger rules in R under the weak minimal semantics can be handled by ECA^{+} over $2^{\mathcal{P}}$: the start and end points of the chosen non-trigger tokens are mapped to last and next occurrences of propositions in $\mathcal{P}$ w.r.t. the current input position (trigger) of a multi-timeline encoding, while the atoms in the rules are mapped to ECA^{+} event-clock constraints. Note that ECA^{+} cannot express trigger-less rules since the semantics of these rules does not constraint the chosen punctual events to be closest as possible to a reference event.

▷ Claim 1. One can construct in exponential time an ECA^+ $\mathcal{A}_\forall$ over $2^{\mathcal{P}}$ such that for each multi-timeline Π of SV and encoding w_Π of Π, w_Π is accepted by $\mathcal{L}_T(\mathcal{A}_\forall)$ iff Π satisfies the trigger rules in R under the weak minimal semantics. Moreover, $\mathcal{A}_\forall$ has a unique state, $O(N_a)$ atomic event-clock constraints, and maximal constant $O(K_\mathcal{D})$, where N_a is the overall number of atoms in the trigger rules in R.

For the trigger-less rules in R, the following result (Claim 2) has been established in [5] for a slightly different encoding of the multi-timelines. The result can be easily adapted to the encoding proposed here.

▷ Claim 2. One can construct in exponential time a TA $\mathcal{A}_\exists$ over $2^{\mathcal{P}}$ accepting the codes of the multi-timelines of SV which satisfy the *trigger-less* rules in R. Moreover, $\mathcal{A}_\exists$ has $2^{O(N_q+\sum_{x\in SV}|V_x|)}$ states, $O(|SV|+N_q)$ clocks, and maximal constant $O(K_\mathcal{D})$, where N_q is the overall number of quantifiers in the trigger-less rules of R.

By Theorem 11 and Claim 1–2, the first part of Theorem 13 concerning the construction of the TA $\mathcal{A}_\mathcal{D}$ for the TP domain $\mathcal{D}$, directly follows. For the second part of Theorem 13, we recall that non-emptiness of a TA $\mathcal{A}$ can be solved by an **NPSPACE** search algorithm in the *region graph* of $\mathcal{A}$ which uses space logarithmic in the number of states of $\mathcal{A}$ and polynomial in the number of clocks and in the length of the encoding of the maximal constant of $\mathcal{A}$ [2]. Thus, since $\mathcal{A}_\mathcal{D}$ can be built on the fly, and the search in the region graph of $\mathcal{A}_\mathcal{D}$ can be done without explicitly constructing $\mathcal{A}_\mathcal{D}$, membership in **PSPACE** of the weak minimal TP problem follows. **PSPACE**-hardness is proved by a polynomial time reduction from a domino-tiling problem for grids with rows of linear length [22]. ◀

5 Conclusions

We have addressed the TP problem in the dense-time setting under two novel semantics of the trigger rules: the weak and strong minimal ones. Surprisingly, we have shown that, despite the apparently small difference in the two semantics, the strong minimal one leads to an undecidable TP problem, while the weak minimal one leads to a **PSPACE**-complete TP problem. In order to solve the weak minimal TP problem, we have investigated two novel and strictly more expressive extensions of ECA which we believe to be interesting per sé in the field of TA. As for future work, we shall study the strong minimal TP problem when just one or two state variables are used, whose decidability remains an open issue. Moreover, we aim at investigating the TP problem in the controllability setting, where the values of some variables are not under the system control, but depend on the environment.

References

1. R. Alur and T. A. Henzinger. A Really Temporal Logic. *Journal of the ACM*, 41(1):181–204, 1994.
2. Rajeev Alur and David L. Dill. A Theory of Timed Automata. *Theoretical Computer Science*, 126(2):183–235, 1994.
3. Rajeev Alur, Limor Fix, and Thomas A. Henzinger. Event-Clock Automata: A Determinizable Class of Timed Automata. *Theoretical Computer Science*, 211(1-2):253–273, 1999.
4. J. Barreiro, M. Boyce, M. Do, J. Frank, M. Iatauro, T. Kichkaylo, P. Morris, J. Ong, E. Remolina, T. Smith, and D. Smith. EUROPA: A Platform for AI Planning, Scheduling, Constraint Programming, and Optimization. In *Proc. of the 4th ICKEPS*, 2012.
5. L. Bozzelli, A. Molinari, A. Montanari, and A. Peron. Complexity of Timeline-Based Planning over Dense Temporal Domains: Exploring the Middle Ground. In *Proc. of the 9th GandALF 2018*, EPTCS 277, pages 191–205, 2018.

6 L. Bozzelli, A. Molinari, A. Montanari, and A. Peron. Decidability and Complexity of Timeline-Based Planning over Dense Temporal Domains. In *Proc. of the 16th KR*, pages 627–628. AAAI Press, 2018.

7 L. Bozzelli, A. Molinari, A. Montanari, and A. Peron. Undecidability of future timeline-based planning over dense temporal domains. *arxiv.org/abs/1904.09184*, 2019. `arXiv:1904.09184`.

8 L. Bozzelli, A. Molinari, A. Montanari, A. Peron, and G. J. Woeginger. Timeline-Based Planning over Dense Temporal Domains with Trigger-less Rules is NP-Complete. In *Proc. of the 19th ICTCS*, volume 2243 of *CEUR Workshop Proceedings*, pages 116–127, 2018.

9 A. Cesta, G. Cortellessa, S. Fratini, A. Oddi, and N. Policella. An Innovative Product for Space Mission Planning: An A Posteriori Evaluation. In *Proc. of the 17th ICAPS*, pages 57–64, 2007.

10 A. Cesta, A. Finzi, S. Fratini, A. Orlandini, and E. Tronci. Flexible Timeline-Based Plan Verification. In *Proc. of the 32nd KI*, LNCS 5803, pages 49–56. Springer, 2009.

11 A. Cesta, A. Finzi, S. Fratini, A. Orlandini, and E. Tronci. Analyzing Flexible Timeline-Based Plans. In *Proc. of the 19th ECAI*, volume 215 of *Frontiers in Artificial Intelligence and Applications*, pages 471–476. IOS Press, 2010.

12 S. Chien, D. Tran, G. Rabideau, S.R. Schaffer, D. Mandl, and S. Frye. Timeline-Based Space Operations Scheduling with External Constraints. In *Proc. of the 20th ICAPS*, pages 34–41. AAAI, 2010.

13 M. Cialdea Mayer and A. Orlandini. An Executable Semantics of Flexible Plans in Terms of Timed Game Automata. In *Proc. of the 22nd TIME*, pages 160–169. IEEE Computer Society, 2015.

14 M. Cialdea Mayer, A. Orlandini, and A. Ubrico. A Formal Account of Planning with Flexible Timelines. In *Proc. of the 21st TIME*, pages 37–46. IEEE Computer Society, 2014.

15 M. Cialdea Mayer, A. Orlandini, and A. Umbrico. Planning and Execution with Flexible Timelines: a Formal Account. *Acta Informatica*, 53(6–8):649–680, 2016.

16 A. Cimatti, A. Micheli, and M. Roveri. Timelines with Temporal Uncertainty. In *Proc. of the 27th AAAI*, 2013.

17 M. Fox and D. Long. PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains. *Journal of Artificial Intelligence Research*, 20:61–124, 2003.

18 J. Frank and A. Jónsson. Constraint-based Attribute and Interval Planning. *Constraints*, 8(4):339–364, 2003.

19 G. Geeraerts, J.F. Raskin, and N. Sznajder. Event Clock Automata: From Theory to Practice. In *Proc. of the 9th FORMATS*, LNCS 6919, pages 209–224. Springer, 2011.

20 N. Gigante, A. Montanari, M. Cialdea Mayer, and A. Orlandini. Timelines are Expressive Enough to Capture Action-based Temporal Planning. In *Proc. of the 23rd TIME*, pages 100–109. IEEE Computer Society, 2016.

21 N. Gigante, A. Montanari, M. Cialdea Mayer, and A. Orlandini. Complexity of Timeline-Based Planning. In *Proc. of the 27th ICAPS*, pages 116–124. AAAI Press, 2017.

22 D. Harel. *Algorithmics: The spirit of computing*. Wesley, 2nd edition, 1992.

23 A. K. Jónsson, P. H. Morris, N. Muscettola, K. Rajan, and B. D. Smith. Planning in Interplanetary Space: Theory and Practice. In *Proc. of the 5th AIPS*, pages 177–186. AAAI, 2000.

24 M. L. Minsky. *Computation: Finite and Infinite Machines*. Prentice-Hall, Inc., 1967.

25 N. Muscettola. HSTS: Integrating Planning and Scheduling. In *Intelligent Scheduling*, pages 169–212. Morgan Kaufmann, 1994.

26 J. Ouaknine and J. Worrell. On Metric Temporal Logic and Faulty Turing Machines. In *Proc. of the 9th FOSSACS*, LNCS 3921, pages 217–230. Springer, 2006.

27 J. Ouaknine and J. Worrell. On the decidability and complexity of Metric Temporal Logic over finite words. *Logical Methods in Computer Science*, 3(1), 2007.

28 J. Rintanen. Complexity of Concurrent Temporal Planning. In *Proc. of the 17th ICAPS*, pages 280–287. AAAI, 2007.