Model Checking Timeline-based Systems 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. In this paper, we introduce an automaton-theoretic approach to model checking linear time properties of timeline-based systems over dense temporal domains. The system under consideration is specified by means of (a decidable fragment of) *timeline* structures, timelines for short, which are a formal setting proposed in the literature to model planning problems in a declarative way. Timelines provide an interval-based description of the behavior of the system, instead of a more conventional point-based one. The relevant system properties are expressed by formulas of the logic MITL (a well-known timed extension of LTL) to be checked against timelines. In the paper, we prove that the model checking problem for MITL formulas (resp., its fragment $\mathsf{MITL}_{(0,\infty)}$) over timelines is **EXPSPACE**-complete (resp., **PSPACE**-complete).

Keywords: Model checking · Timelines · Timed Automata · Metric Interval Temporal Logic

1 Introduction

Model checking (MC) is a set of techniques that allow for the automatic verification of (temporal) properties of a system, where the model is usually represented by a (finite) Kripke structure and the properties by logics such as LTL or CTL. The representation of the system is inherently point-based, as it makes explicit how a system evolves *state-by-state* (i.e., how it can move from a state to another one, according to the transition function), and it describes which are the atomic properties that hold at every state. In the case of real-time systems, the behaviour of the system has to be checked also against quantitative timing properties and constraints. In this case, the model of the system has to be enriched with quantitative time information. For instance, a timed transition system provides information about the time elapsing when moving from a state of the system to the following one. To express properties of real-time systems, quantitative time

 The work by A. Montanari and A. Peron has been supported by the GNCS Project 2019 "Formal methods for combined verification techniques".

extensions of LTL (and CTL), such as, for instance, Metric Temporal Logic (MTL, [14]), have been introduced.

In this paper, we focus on the problem of model checking real-time systems, where the commonly adopted point-based representation has been replaced by an interval-based one. The basic idea is that the system can be decomposed into a number of components which evolve in parallel, possibly synchronising at some points. The behaviour of a component is described by a *timeline*, namely, a sequence of intervals during which the component lasts in a state.

Timelines have been fruitfully exploited in temporal planning for quite a long time. We borrow the formal definition of systems from *timeline-based planning* (TP) [10]. TP can be viewed as an alternative to the classic action-based approach to planning, and it has been successfully applied in several contexts (see, e.g., [3,8,9,15]). Action-based planning aims at determining a sequence of actions that, given the initial state of the world and a goal, transforms, step by step, the state of the world until a state satisfying the goal is reached. Compared to action-based planning, TP adopts a more declarative approach. It models the planning domain as a set of independent (but interacting) components, each one consisting of a number of *state variables*. The evolution of the values of state variables over time is described by means of a set of *timelines* (in turn, these are sequences of time intervals, called *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 (values of) state variables.

In [11,12] Gigante et al. proved that TP is **EXPSPACE**-complete over *discrete temporal domains*. Assuming the temporal domain to be dense allows us to avoid discretization in system descriptions, that is, it makes it possible to describe systems at a higher level of abstraction, enabling us to neglect unnecessary details and paving the way for a really general interval-based MC. Even though the TP problem over *dense* temporal domains is known to be *undecidable* in the general case [5], decidability can be recovered by imposing significant restrictions on the logical structure of synchronization rules ([7,4]). Computational complexities suitable for a practical exploitation of TP can be achieved by further constraining the form of the rules [4].

In this paper, by making use of known results on dense-time TP, we study timeline-based MC, where systems are modeled by timelines over dense temporal domains and properties are expressed by means of formulas of *Metric Interval Temporal Logic* (MITL) [2], a timed extension of LTL which is interpreted over timed words. To solve the MC problem, we exploit an automaton-theoretic approach. To specify the system, we use a fragment of TP, namely, Future TP with simple rules and non singular intervals (details in the following), which is known to be **EXPSPACE**-complete. Solving TP is a preliminary step for MC since properties are checked against timelines satisfying the synchronization rules (a sort of a "feasibility check" of the system description). The connection with the MITL logic is obtained by suitably encoding timelines into time words and defining the MC problem in terms of containment of timed languages. By exploiting *timed automata* [1] constructions to define the encoding of the timeline-based description of the system and the models of MITL properties, we prove that the MC problem for MITL (resp., its fragment $\mathsf{MITL}_{(0,\infty)}$ [2]) over timelines is **EXPSPACE**-complete (resp., **PSPACE**-complete).

The paper is organized as follows. In Section 2, we give some background knowledge about the TP framework. In Section 3, we introduce and solve the MC problem for systems described as timelines against MITL. In Section 4, we provide some conclusive remarks.

2 The TP Problem

Let $\mathbb{N}$ be the set of natural numbers and $\mathbb{R}_+$ be the set of non-negative real numbers; moreover, let *Intv* denote the set of intervals of $\mathbb{R}_+$ whose endpoints are in $\mathbb{N} \cup \{\infty\}$, and let $\mathit{Intv}_{(0,\infty)}$ denote the set of non-singular intervals $I \in \mathit{Intv}$ such that either I is unbounded, or I is left-closed with left endpoint 0. Intervals in $\mathit{Intv}_{(0,\infty)}$ can be represented by expressions of the form $\sim n$, for some $n \in \mathbb{N}$ and $\sim \in \{<, \leq, >, \geq\}$. We now introduce the basic notions of the TP framework [10,11]. The 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 state variable* x,
- $T_x : V_x \to 2^{V_x}$ *is the* value transition function, *which maps each* $v \in V_x$ *to the (possibly empty) set of successor values, and*
- $D_x : V_x \to \mathit{Intv}$ *is the* constraint (or duration) function *that maps each* $v \in V_x$ *to an interval of Intv.*

A *token* for a state 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. In order to clarify the variable to which a token refers, we shall often denote (v, d) as (x, v, d).

The behavior of the state variable x is specified by means of a *timeline*, which is a non-empty sequence of tokens $\pi = (v_0, d_0) \cdots (v_n, d_n)$ consistent with the value transition function T_x, namely, 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 of the timeline π are defined as: $\mathsf{s}(\pi, i) = 0$ if $i = 0$, and $\mathsf{s}(\pi, i) = \sum_{h=0}^{i-1} d_h$ otherwise; $\mathsf{e}(\pi, i) = \sum_{h=0}^{i} d_h$. See Figure 1 for an example of a timeline.

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

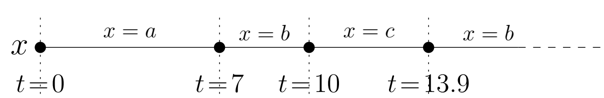

Fig. 1. An example of timeline $(a, 7)(b, 3)(c, 3.9)\cdots$ for the state variable $x = (V_x, T_x, D_x)$, where $V_x = \{a, b, c, \ldots\}$, $b \in T_x(a)$, $c \in T_x(b)$, $b \in T_x(c)\ldots$ and $D_x(a) = [5, 8]$, $D_x(b) = [1, 4]$, $D_x(c) = [2, \infty[\ldots$

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 $\Sigma = \{o, o_0, o_1, o_2, \ldots\}$ of *token names* to refer to the tokens along a multi-timeline, and are based on the notions of *atom* and *existential statement*.

Atom. An *atom* ρ 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 $o \leq_I^{e} n$ (respectively, $n \leq_I^{e} o$) *is satisfied by* λ_Π if $n - e(\lambda_\Pi(o)) \in I$ (respectively, $e(\lambda_\Pi(o)) - n \in I$).

*Existential statement*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 $1 \leq i \leq n$.

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 quantifier $o_i[x_i = v_i]$, we have $\lambda_\Pi(o_i) = (\pi, h)$, 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.

We can now introduce synchronization rules, which constrain tokens, possibly belonging to different timelines.

Definition 2. *A* synchronization rule $\mathcal{R}$ *for a finite set* SV *of state variables is a rule of one of the forms* $o_0[x_0 = v_0] \to \mathcal{E}_1 \vee \mathcal{E}_2 \vee \ldots \vee \mathcal{E}_k$, $\top \to \mathcal{E}_1 \vee \mathcal{E}_2 \vee \ldots \vee \mathcal{E}_k$, *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*; we require that only* o_0 *may occur free in* $\mathcal{E}_i$, *for all* $1 \leq i \leq n$. *In rules of the second form (*trigger-less rules*), no token name may occur 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, the trigger $o_0[x_0 = v_0]$ acts as a universal quantifier, which states that *for all* the tokens of the timeline for x_0, where x_0 takes the value v_0, at least one of the existential statements $\mathcal{E}_i$ must be satisfied. As an example, $o_0[x_0 = v_0] \to \exists o_1[x_1 = v_1].o_0 \leq_{[2,\infty[}^{\mathsf{e},\mathsf{s}} o_1$ states that after *every* token for x_0, with value v_0, there exists a token for x_1, with value v_1, *starting* at least 2 time instants after the *end* of the former. Trigger-less rules simply assert the satisfaction of some existential statement. The meaning of *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.

A variable describes the timed behaviour of a single module of a system. Trigger-less rules can be used to express either initial conditions of modules or goals (timed reachability properties). Synchronization rules are used either to synchronize the behaviour of modules or to express invariant timed properties.

Definition 3. *Let Π be a multi-timeline of a set SV of state variables. (i) Given a* trigger-less rule *$\mathcal{R}$ of SV, Π* satisfies *$\mathcal{R}$ if Π satisfies some existential statement of $\mathcal{R}$. (ii) 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 $\pi = \Pi(x_0)$ for x_0 such that $\pi(i) = (v_0, d)$, there exists an existential statement $\mathcal{E}$ of $\mathcal{R}$ and a Σ-assignment λ_Π for Π consistent with $\mathcal{E}$ such that $\lambda_\Pi(o_0) = (\pi, i)$ and λ_Π satisfies all the atoms of $\mathcal{E}$.*

In the following, we shall consider also a stronger notion of satisfaction of trigger rules, called *satisfaction under the future semantics*, which requires that all non-trigger tokens selected by some quantifier do not start *strictly before* the start time of the trigger token.

Definition 4. *A multi-timeline Π of SV satisfies a trigger rule $\mathcal{R} = o_0[x_0 = v_0] \to \mathcal{E}_1 \vee \mathcal{E}_2 \vee \ldots \vee \mathcal{E}_k$* under the future semantics *if Π satisfies the trigger rule obtained from $\mathcal{R}$ by replacing each $\mathcal{E}_i = \exists o_1[x_1 = v_1] \cdots \exists o_n[x_n = v_n].\mathcal{C}$ by $\mathcal{E}'_i = \exists o_1[x_1 = v_1] \cdots \exists o_n[x_n = v_n].\big(\mathcal{C} \wedge \bigwedge_{i=1}^{n} o_0 \leq^{\mathsf{s},\mathsf{s}}_{[0,+\infty[} o_i\big)$.*

Finally, a *TP domain* $P = (SV, R)$ is specified by a finite set SV of state variables and a finite set R of synchronization rules for SV modeling their admissible behaviors. As already pointed out, trigger-less rules can be used to express initial and intermediate conditions and as well as the goals of the problem, while trigger rules are much more powerful and useful, and can be exploited, for instance, to specify invariants and response requirements.

A *plan for* $P = (SV, R)$ is a multi-timeline of SV satisfying all the rules in R. A *future plan for* P is defined in a similar way, but it requires the satisfaction of *all* trigger rules under the future semantics.

The *TP problem* (resp., *Future TP problem*) is a decision problem formulated as follows: given a TP domain $P = (SV, R)$, is there a plan (resp., a *future* plan) for P? Table 1 summarizes all the known decidability and complexity results for TP and restrictions of TP involving trigger rules with future semantics, simple trigger rules, and intervals in atoms (of trigger rules) which are non-singular or in $\mathit{Intv}_{(0,\infty)}$ [4,5,6]. In particular, both the general TP problem and the future TP problem over dense temporal domains are undecidable [5,6]. In [4], it is proved that decidability of the TP problem can be recovered if we use only *simple* trigger rules under the *future semantics*. If we further assume intervals in trigger rules to be non-singular (resp., to be in $\mathit{Intv}_{(0,\infty)}$), the problem becomes **EXPSPACE**-complete (resp., **PSPACE**-complete). We conclude the section by showing how to specify a simple timed system in a TP domain.

Example 1. Let us consider a system consisting of three components (*temperature sensor*, *processing unit*, and *data transmission unit*) respectively modelled by the state variables, $x_{\texttt{temp}} = (V_{\texttt{temp}}, T_{\texttt{temp}}, D_{\texttt{temp}})$, $x_{\texttt{proc}} = (V_{\texttt{proc}}, T_{\texttt{proc}}, D_{\texttt{proc}})$, and $x_{\texttt{transm}} = (V_{\texttt{transm}}, T_{\texttt{transm}}, D_{\texttt{transm}})$, where

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

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

- $V_{\texttt{temp}} = \{\texttt{ready}, \texttt{not\_ready}\}$, $T_{\texttt{temp}}(\texttt{ready}) = \{\texttt{not\_ready}\}$, $T_{\texttt{temp}}(\texttt{not\_ready}) = \{\texttt{ready}\}$, $D_{\texttt{temp}}(\texttt{ready}) = [1, 2]$, $D_{\texttt{temp}}(\texttt{not\_ready}) = [2, 3]$;
- $V_{\texttt{proc}} = \{\texttt{reading}_1, \texttt{reading}_2, \texttt{read}_0, \texttt{read}_1, \texttt{read}_2\}$, $T_{\texttt{proc}}(\texttt{reading}_1) = \{\texttt{read}_0, \texttt{read}_1\}$, $T_{\texttt{proc}}(\texttt{reading}_2) = \{\texttt{read}_1, \texttt{read}_2\}$, $T_{\texttt{proc}}(\texttt{read}_0) = \{\texttt{reading}_1\}$, $T_{\texttt{proc}}(\texttt{read}_1) = \{\texttt{reading}_2\}$, $T_{\texttt{proc}}(\texttt{read}_2) = \{\texttt{read}_2\}$, $D_{\texttt{proc}}(\texttt{reading}_1) = D_{\texttt{proc}}(\texttt{reading}_2) = [1, 2]$, $D_{\texttt{proc}}(\texttt{read}_0) = D_{\texttt{proc}}(\texttt{read}_1) = D_{\texttt{proc}}(\texttt{read}_2) = [2, 3]$;
- $V_{\texttt{transm}} = \{\texttt{send}\}$, $T_{\texttt{transm}}(\texttt{send}) = \{\texttt{send}\}$, $D_{\texttt{transm}}(\texttt{send}) = [2, 5]$.

The temperature sensor swaps between the states `ready` and `not_ready`. In the former, it senses the temperature of the environment and *possibly* sends the temperature value to the processing unit. The processing unit receives *two* temperature samples from the sensor, and sends the average value to the data transmission unit. When in state $\texttt{read}_i$, for $i = 0, 1, 2$, i samples have been read, while when in $\texttt{reading}_j$, for $j = 1, 2$, it is attempting to read the j-th sample. A successful reading requires the occurrence of a `ready` token of the sensor *within* a reading token of the processing unit (conversely, such an occurrence does not guarantee the success of reading). Analogously, the processing unit can send data to the transmitter only if a token with value $\texttt{read}_2$ occurs within a token `send`.

The sensor starts in state `not_ready` and the processing unit starts in state $\texttt{reading}_1$, as required by the trigger-less rules $\top \to \exists o[x_{\texttt{temp}} = \texttt{not\_ready}].o \leq^{\mathsf{s}}_{[0,0]} 0$ and $\top \to \exists o[x_{\texttt{proc}} = \texttt{reading}_1].o \leq^{\mathsf{s}}_{[0,0]} 0$. (Recall that trigger-less rules may also contain singular intervals.) The goal of the system is encoded by the rule $\top \to \exists o_1[x_{\texttt{proc}} = \texttt{read}_2]\exists o_2[x_{\texttt{transm}} = \texttt{send}].(o_2 \leq^{\mathsf{s},\mathsf{s}}_{[0,+\infty[} o_1 \wedge o_1 \leq^{\mathsf{e},\mathsf{e}}_{[0,+\infty[} o_2)$.

The synchronization between the sensor and the processing unit for reading is given by the following simple trigger rule (*under the future semantics*).

$$o[x_{\texttt{proc}} = \texttt{reading}_1] \to (\exists o_1[x_{\texttt{proc}} = \texttt{read}_0].o \leq^{\mathsf{e},\mathsf{s}}_{[0,1]} o_1) \vee \\ (\exists o_2[x_{\texttt{proc}} = \texttt{read}_1]\exists o_3[x_{\texttt{temp}} = \texttt{ready}].o \leq^{\mathsf{e},\mathsf{s}}_{[0,1]} o_2 \wedge o_3 \leq^{\mathsf{e},\mathsf{e}}_{[0,+\infty[} o). \quad (1)$$

Each reading attempt (token $\texttt{reading}_1$) can be either unsuccessful being followed by $\texttt{read}_0$ (first existential statement) or successful including a `ready` token and being followed by a $\texttt{read}_1$ token (second existential statement). Due to the future semantics, the token o starts no later than o_3. A similar rule is given for the second temperature sample. In Figure 2, we show an example of a plan/computation for the system described by $P = (\{x_{\texttt{temp}}, x_{\texttt{proc}}, x_{\texttt{transm}}\}, R)$.

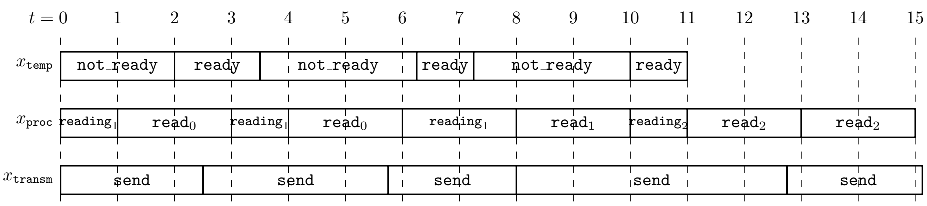

Fig. 2. An example of a computation of the considered system (plan).

3 Timeline-based MC for MITL specifications

In this section, we define the problem of model checking systems specified by means of timelines against properties expressed in terms of the logic MITL. The logic MITL is a fragment of the Metric Temporal logic (MTL) which extends LTL with time constraints on the until modality and is interpreted over *timed words*. The key idea is that multi-timelines can be naturally encoded into timed words so that MITL can be used to express properties of (encoded) timelines.

We start by introducing timed words and MITL. Then, we define the timeline-based MC problem for MITL, and we devise a procedure to solve it.

Timed words. 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 $\Sigma \times \mathbb{R}_+$ (the *timestamp* τ_i is the time at which the "event" a_i occurs) such that $\tau_i \leq \tau_{i+1}$ for all $0 \leq i < n$ (*monotonicity* requirement). We often denote the timed word w by (σ, τ), where σ is the finite (untimed) word $a_0 \cdots a_n$ and τ is the sequence of timestamps $\tau_0, \ldots, \tau_n$. A *timed language* over Σ is a set of timed words over Σ.

The logic MTL. Metric Temporal Logic (MTL) extends LTL with time constraints on the until modality [14]. Let $\mathcal{AP}$ be a finite set of proposition letters. MTL formulas φ over $\mathcal{AP}$ are defined by the grammar $\varphi ::= \top \mid p \mid \varphi \vee \varphi \mid \neg\varphi \mid \varphi \mathsf{U}_I \varphi$, where $p \in \mathcal{AP}$, $I \in \mathit{Intv}$, and U_I is the *strict timed until* MTL modality.

MTL formulas over $\mathcal{AP}$ are interpreted on timed words over $2^{\mathcal{AP}}$. Given an MTL formula φ, a timed word $w = (\sigma, \tau)$ over $2^{\mathcal{AP}}$, and a position $0 \leq i < |w|$, the satisfaction relation $(w, i) \models \varphi$—meaning that φ holds at position i of w—is defined as follows (we omit the clauses for Boolean connectives):

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

A *model of* φ is a timed word w over $2^{\mathcal{AP}}$ such that $(w, 0) \models \varphi$. The *timed language* $\mathcal{L}_T(\varphi)$ is the set of models of φ.

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

Encoding multi-timelines into timed words. Given $P = (SV, R)$, let us define an encoding of the multi-timelines of SV by means of timed words over $2^{\mathcal{AP}}$ for a suitable finite set $\mathcal{AP}$ of proposition letters. For each $x \in SV$, we let $x = (V_x, T_x, D_x)$. Given an interval $I \in \mathit{Intv}$ and some $n \in \mathbb{N}$, let $n + I$ (resp., $n - I$) denote the set of non-negative real numbers $\tau \in \mathbb{R}_+$ such that $\tau - n \in I$ (resp., $n - \tau \in I$). For an atom ρ in R involving a time constant (time-point atom), let $I(\rho)$ be the interval in Intv defined as follows: if ρ has the form $o \leq^{e}_{I} n$ (resp., $n \leq^{e}_{I} o$), then $I(\rho) = n - I$ (resp., $I(\rho) = n + I$). Finally, let Intv_R be the set of intervals $J \in \mathit{Intv}$ such that $J = I(\rho)$ for some time-point atom ρ occurring in a trigger rule of R.

For any pair of distinct state variables x and x', we assume the sets V_x and $V_{x'}$ to be disjoint. To encode multi-timelines of SV, we use the set $\mathcal{AP} = (\bigcup_{x \in SV} \mathit{Main}_x) \cup \mathit{Deriv}$ of proposition letters, where $\mathit{Main}_x = ((\{\mathit{beg}_x\} \cup V_x) \times V_x) \cup (V_x \times \{\mathit{end}_x\})$ and $\mathit{Deriv} = \mathit{Intv}_R \cup \{p_>\} \cup \bigcup_{x \in SV} \bigcup_{v \in V_x} \{\mathit{past}^{\mathsf{s}}_v, \mathit{past}^{\mathsf{e}}_v\}$. Intuitively, we use proposition letters in Main_x to encode a token along a timeline for x. Proposition letters in Deriv enrich the encoding in order to translate simple trigger rules in MTL formulas under the future semantics (see below). The tags beg_x and end_x in Main_x are used to mark the start and the end of a timeline for x. A token tk with value v along a timeline for x is encoded by two events: the *start-event* (occurring at the start time of tk) and the *end-event* (occurring at the end time of tk). The start-event of tk is specified by a main proposition letter of the form (v_p, v), where either $v_p = \mathit{beg}_x$ (tk is the first token of the timeline) or v_p is the value of the token for x preceding tk. The end-event of tk is instead specified by a main proposition letter of the form (v, v_s), where either $v_s = \mathit{end}_x$ (tk is the last token of the timeline) or v_s is the value of the token for x following tk. An example of encoding is given in Figure 3. Let us consider now the proposition letters in Deriv. The elements in Intv_R reflect the semantics of the time-point atoms in the trigger rules of R: for each $I \in \mathit{Intv}_R$, I holds at the current position if the current timestamp τ satisfies $\tau \in I$. The proposition $p_>$ is used to mark a timestamp if it is strictly greater than the previous one. Finally, a proposition $\mathit{past}^{\mathsf{s}}_v$ (resp., $\mathit{past}^{\mathsf{e}}_v$) is used to mark a timestamp τ if it is preceded by a token of value v starting (resp., ending) at the same time τ.

An *encoding of a timeline for* x is a timed word w over $2^{\mathit{Main}_x \cup \mathit{Deriv}}$ of the form $w = (\{(\mathit{beg}_x, v_0)\} \cup S_0, \tau_0)(\{(v_0, v_1)\} \cup S_1, \tau_1) \cdots (\{(v_n, \mathit{end}_x)\} \cup S_{n+1}, \tau_{n+1})$ where, for all $0 \leq i \leq n+1$, $S_i \subseteq \mathit{Deriv}$, and (i) $v_{i+1} \in T_x(v_i)$ for $i < n$; (ii) $\tau_0 = 0$ and $\tau_{i+1} - \tau_i \in D_x(v_i)$ for $i \leq n$; (iii) $S_i \cap \mathit{Intv}_R$ is the set of intervals $I \in \mathit{Intv}_R$ such that $\tau_i \in I$; (iv) $p_> \in S_i$ if and only iff either $i = 0$ or $\tau_i > \tau_{i-1}$; (v) for all $v \in V_x$, $\mathit{past}^{\mathsf{s}}_v \in S_i$ (resp., $\mathit{past}^{\mathsf{e}}_v \in S_i$) if and only if there is $0 \leq h < i$ such that $\tau_h = \tau_i$ and $v = v_h$ (resp., $\tau_h = \tau_i$, $v = v_{h-1}$ and $h > 0$). Note that the length of w is at least 2. The given timed word w encodes the timeline for x of length $n+1$ given by $\pi = (v_0, \tau_1)(v_1, \tau_2 - \tau_1) \cdots (v_n, \tau_{n+1} - \tau_n)$. The timestamps τ_i and τ_{i+1} represent the start and the end time of the i-th token of the timeline π $(0 \leq i \leq n)$. See again Figure 3 for an example.

Next, we define the encoding of a multi-timeline Π of SV. For $P \subseteq \mathcal{AP}$ and $x \in SV$, let $P[x] = P \setminus \bigcup_{y \in SV \setminus \{x\}} \mathit{Main}_y$. An *encoding of a multi-timeline* Π *of*

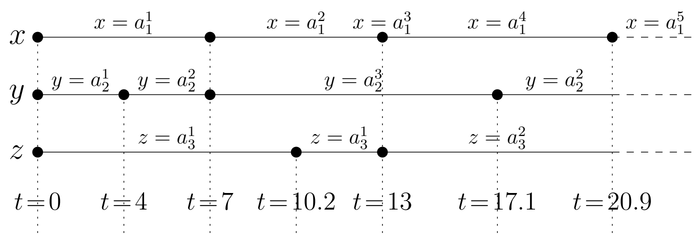

Fig. 3. Example of multi-timeline of $SV = \{x, y, z\}$. The timeline for x is $(a_1^1, 7), (a_1^2, 5), (a_1^3, 0), (a_1^4, 7.9), (a_1^5, \ldots)$. Note that the third token has null duration. The encoding of the timeline for x is $\big(\{(beg_x, a_1^1), p_>\}, 0\big)\big(\{(a_1^1, a_1^2), p_>\}, 7\big)\big(\{(a_1^2, a_1^3), p_>\}, 13\big)\big(\{(a_1^3, a_1^4), past^{\mathsf{s}}_{a_1^3}, past^{\mathsf{e}}_{a_1^2}\}, 13\big)$ $\big(\{(a_1^4, a_1^5), p_>\}, 20.9\big)\cdots$ The encoding of the multi-timeline is $\big(\{(beg_x, a_1^1), (beg_y, a_2^1), (beg_z, a_3^1), p_>\}, 0\big)\big(\{(a_2^1, a_2^2), p_>\}, 4\big)\big(\{(a_1^1, a_1^2), (a_2^2, a_2^3), p_>\}, 7\big)$ $\big(\{(a_3^1, a_3^1), p_>\}, 10.2\big)\big(\{(a_1^2, a_1^3), (a_3^1, a_3^2), p_>\}, 13\big)\big(\{(a_1^3, a_1^4), past^{\mathsf{s}}_{a_1^3}, past^{\mathsf{e}}_{a_1^2}\}, 13\big)$ $\big(\{(a_2^3, a_2^2), p_>\}, 17.1\big)\cdots$

SV, written w_Π, is a timed word w over $2^{\mathcal{AP}}$ 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 $(P_0[x], \tau_0) \cdots (P_n[x], \tau_n)$ by removing the pairs $(P_i[x], \tau_i)$ such that $P_i[x] \cap Main_x = \emptyset$ is an encoding of a timeline for x, and (ii) $P_0[x] \cap Main_x \neq \emptyset$ for all $x \in SV$ (initialization). For a system model $P_{\mathrm{sys}} = (SV, R)$, $\mathcal{L}_T(P_{\mathrm{sys}}) = \{w_\Pi : \Pi \text{ is a plan for } P_{\mathrm{sys}}\}$ is the set of timed words over $2^{\mathcal{AP}}$ encoding plans of P_{sys}.

Example 2. We list some $\mathsf{MITL}_{(0,\infty)}$ properties for the timed systems of Example 1. We first introduce some auxiliary formulas. Let $x \in SV$ and $v \in V_x$; $\psi(\mathsf{s}, v)$ and $\psi(\mathsf{e}, v)$ are two propositional formulas over $Main_x$ defined as: $\psi(\mathsf{s}, v) = (beg_x, v) \vee \bigvee_{u \in V_x} (u, v)$ and $\psi(\mathsf{e}, v) = (v, end_x) \vee \bigvee_{u \in V_x} (v, u)$. Intuitively, $\psi(\mathsf{s}, v)$ (resp., $\psi(\mathsf{e}, v)$) states that a start-event (resp., end-event) for a token for x with value v occurs at the current time. Finally, given an MTL formula θ, we define the MTL formula $EqTime(\theta) = \theta \vee [\neg p_> \mathsf{U}_{\geq 0} (\neg p_> \wedge \theta)]$, which is satisfied by an encoding of a multi-timeline at time τ if θ eventually holds in future position having the same timestamp τ.

- $\mathsf{G}_{<2}\ \neg\psi(\mathsf{s}, \mathtt{ready})$, which holds true in any system computation, as the sensor does not ever get ready by 2 seconds;
- $\mathsf{F}_{\leq 8}\ \psi(\mathsf{s}, \mathtt{read}_1)$ which does not hold true in all the computations (but it holds in the computation in Figure 2), since the sensor and the processing unit may synchronize for the first time after 8 seconds;
- $\mathsf{F}_{\geq 0}\big(\psi(\mathsf{s}, \mathtt{ready}) \wedge (\top\, \mathsf{U}_{>0}\ \psi(\mathsf{s}, \mathtt{ready}))\big)$ which holds true in any system computation, since the system fulfills the goal of eventually sending the data and, consequently, the sensor must become ready (at least) twice.
- $\mathsf{G}_{\geq 0}\big(\psi(\mathsf{s}, \mathtt{read}_1) \rightarrow \mathsf{F}_{\leq 3}\ \psi(\mathsf{s}, \mathtt{read}_2)\big)$, which is not true in all computations as the processing unit, after reading the first sample, may not be able to read the second one by 3 time units due to a delayed synchronization.

– $\mathsf{G}_{\geq 0}\big(\psi(\mathsf{s}, \mathtt{reading_1}) \wedge (EqTime(\psi(\mathsf{s}, \mathtt{ready})) \vee past^{\mathsf{s}}_{\mathtt{ready}}) \rightarrow \mathsf{F}_{\leq 2}\, \psi(\mathsf{s}, \mathtt{read_1})\big)$. We recall that the proposition letter $past^{\mathsf{s}}_{\mathtt{ready}}$ is true at the time of interpretation if there is a preceding token for $x_{\mathtt{temp}}$ with value `ready` starting at the same time. It is globally required that, whenever a token $\mathtt{reading_1}$ starts together with a token `ready`, a token for $\mathtt{read_1}$ starts within 2 times units. The invariant does not generally hold, as either (i) the token $\mathtt{reading_1}$ may not contain the token `ready`, hence $x_{\mathtt{proc}}$ will not move to the state $\mathtt{read_1}$ by 2 time units, or (ii) the token $\mathtt{reading_1}$ may be followed by a token $\mathtt{read_0}$ (when the synchronization required to move from $\mathtt{reading_1}$ to $\mathtt{read_0}$ fails).

Let us now formally define the timeline-based MC problem for MITL formulas.

Definition 5 (Model checking). *Given a system model $P_{sys} = (SV, R)$ and a MITL formula φ over $\mathcal{AP}$, the timeline-based MC problem for MITL formulas is to decide whether $\mathcal{L}_T(P_{sys}) \subseteq \mathcal{L}_T(\varphi)$.*

To solve the timeline-based MC problem for MITL we adopt an automaton theoretic approach which exploits Timed Automata (TA) as a reference model.

Timed Automata (TA). Let C be a finite set of clocks. A clock valuation is a function $val : C \rightarrow \mathbb{R}_+$ for C that assigns a non-negative real value to each clock in C. Given $t \in \mathbb{R}_+$ and a set $Res \subseteq C$ (called *reset set*), $(val + t)$ and $val[Res]$ denote the valuations for C 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.

A *clock constraint* θ over C is a Boolean combination of atomic formulas of the form $c \in I$ or $c - c' \in I$, where $c, c' \in C$ and $I \in Intv$. Given a clock valuation val and a clock constraint θ, val is said to satisfy θ, written $val \models \theta$, if θ evaluates to true after replacing each occurrence of a clock c in θ by $val(c)$, and interpreting Boolean connectives and membership to intervals in the standard way. We denote by $\Phi(C)$ the set of all possible clock constraints over C.

Definition 6. *A timed automaton (TA) over Σ is a tuple $\mathcal{A} = (\Sigma, Q, q_0, C, \Delta, F)$, where Q is a finite set of (control) states, $q_0 \in Q$ is the initial state, C is a finite set of clocks, $F \subseteq Q$ is the set of accepting states, and $\Delta \subseteq Q \times \Sigma \times \Phi(C) \times 2^C \times Q$ is the transition relation.*

The maximal constant of $\mathcal{A}$ *is the greatest integer occurring as an endpoint of some interval in the clock constraints of the transitions of $\mathcal{A}$.*

Intuitively, while transitions of a TA are performed instantaneously, 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. Clock constraints are used as guards of transitions to restrict the behavior of the automaton.

A configuration of $\mathcal{A}$ is a pair (q, val), where $q \in Q$ and val is a clock valuation for C. 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 the initial configuration (q_0, val_0), with $val_0(c) = 0$ for all $c \in C$ (*initiation requirement*) and such that, for $0 \leq i \leq n$, (i) $(q_i, a_i, \theta, Res, q_{i+1}) \in \Delta$ for some $\theta \in \Phi(C)$ and reset set Res, (ii) $(val_i + \tau_i - \tau_{i-1}) \models \theta$, and (iii) $val_{i+1} = (val_i + \tau_i - \tau_{i-1})[Res]$, where $\tau_{-1} = 0$ (*consecution requirement*).

The behavior of a TA $\mathcal{A}$ can be described as follows. Assume that $\mathcal{A}$ is on state $q \in Q$ after reading the symbol (a', τ_i) at time τ_i and, at that time, the clock valuation is val. Upon reading (a, τ_{i+1}), $\mathcal{A}$ chooses a transition of the form $\delta = (q, a, \theta, Res, q') \in \Delta$ such that the constraint θ is fulfilled by $(val + t)$, with $t = \tau_{i+1} - \tau_i$. The control then changes from q to q' and val is updated in such a way as to record the amount of elapsed time t in the clock valuation, and to reset the clocks in Res, namely, val is updated to $(val + t)[Res]$.

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

As shown in [1], given two TA $\mathcal{A}_1$, with s_1 states and k_1 clocks, and $\mathcal{A}_2$, with s_2 states and k_2 clocks, the union (resp., intersection) automaton $\mathcal{A}_\vee$ (resp., $\mathcal{A}_\wedge$) such that $\mathcal{L}_T(\mathcal{A}_\vee) = \mathcal{L}_T(\mathcal{A}_1) \cup \mathcal{L}_T(\mathcal{A}_2)$ (resp., $\mathcal{L}_T(\mathcal{A}_\wedge) = \mathcal{L}_T(\mathcal{A}_1) \cap \mathcal{L}_T(\mathcal{A}_2)$) can be effectively computed, and has $s_1 + s_2$ states (resp., $s_1 \cdot s_2$ states) and $k_1 + k_2$ clocks (resp., $k_1 + k_2$ clocks).

The automaton theoretic construction for the MC problem. Let $P = (SV, R)$ be an instance of the problem where the trigger rules in R are simple. The *maximal constant* of P, denoted by K_P, is the greatest integer occurring in the atoms of the rules in R and in the constraint functions of the state variables in SV.

The proposed approach exploits some automata constructions proposed in [4] to solve the Future TP problem with simple rules as follows:

1. It is possible to construct a TA $\mathcal{A}_{SV}$ over $2^{\mathcal{AP}}$ accepting the encodings of the multi-timelenes for SV;
2. It is possible to define an MTL formula $\varphi_\forall$ over $\mathcal{AP}$ such that for each multi-timeline Π of SV and encoding w_Π of Π, w_Π is a model of $\varphi_\forall$ if and only if Π satisfies all the trigger rules in R under the future semantics. Mostly relevant for our pourposes, if the intervals in the trigger rules are non-singular the formula $\varphi_\forall$ is a MITL formula;
3. It is possible to construct a TA $\mathcal{A}_\exists$ over $2^{\mathcal{AP}}$ such that for each multi-timeline Π of SV and encoding w_Π of Π, w_Π is accepted by $\mathcal{A}_\exists$ if and only if Π satisfies all the trigger-less rules in R;
4. In summary, there is a future plan for $P = (SV, R)$ if and only if $\mathcal{L}_T(\mathcal{A}_{SV}) \cap \mathcal{L}_T(\mathcal{A}_\exists) \cap \mathcal{L}_T(\varphi_\forall) \neq \emptyset$.

We list the precise results in [4] used for the construction:

Theorem 1. *For $P = (SV, R)$ with maximal constant K_P we build:*
1. *a TA $\mathcal{A}_{SV}$ over $2^{\mathcal{AP}}$, with $2^{O(\sum_{x \in SV} |V_x|)}$ states, $|SV| + 2$ clocks, and maximal constant $O(K_P)$, such that $\mathcal{L}_T(\mathcal{A}_{SV})$ is the set of encodings of the multi-timelines of SV (it is built in exponential time);*
2. *an MTL formula $\varphi_\forall$, with maximal constant $O(K_P)$, such that for each multi-timeline Π of SV and encoding w_Π of Π, w_Π is a model of $\varphi_\forall$ if and only if Π satisfies all the simple trigger rules in R, under the future semantics (it is built in linear time).*

The formula $\varphi_\forall$ *has* $O(|R| \cdot N_A \cdot N_{\mathcal{E}} \cdot \big(|\mathit{Intv}_R| + (\sum_{x \in SV} |V_x|)^2\big))$ *distinct subformulas, where* N_A *(resp.,* $N_{\mathcal{E}}$*) is the maximum number of atoms (resp., existential statements) in a trigger rule of* R*.*
The formula $\varphi_\forall$ *is an* MITL *(resp.,* $\mathsf{MITL}_{(0,\infty)}$*) formula if the intervals in the trigger rules are non-singular (resp., belong to* $\mathit{Intv}_{(0,\infty)}$*);*

3. *a* TA $\mathcal{A}_\exists$ *over* $2^{\mathcal{AP}}$ *such that, for each multi-timeline* Π *of* SV *and encoding* w_Π *of* Π*,* w_Π *is accepted by* $\mathcal{A}_\exists$ *if and only if* Π *satisfies all the trigger-less rules in* R *(it is built in exponential time).*
   $\mathcal{A}_\exists$ *has* $2^{O(N_q)}$ *states,* $O(N_q)$ *clocks, and maximal constant* $O(K_P)$*, where* N_q *is the overall number of quantifiers in the trigger-less rules of* R*.*

In summary, we can construct a TA $\mathcal{A}_{\text{sys}}$ such that $\mathcal{L}_T(P_{\text{sys}}) = \mathcal{L}_T(\mathcal{A}_{\text{sys}})$ and check it for emptyness to solve the Future TP problem with simple trigger rules.

Theorem 2. *([4]) The Future TP problem with simple trigger rules which uses only non-singular intervals in their atoms (resp., intervals in* $\mathit{Intv}_{(0,\infty)}$*) is decidable in* ***EXPSPACE*** *(resp., in* ***PSPACE****).*

The last fundamental fact is the well known result in [2] stating that given a MITL (resp., $\mathsf{MITL}_{(0,\infty)}$) formula ψ having N distinct subformulas and K the largest occurring integer, we can build a TA $\mathcal{A}_\psi$ accepting the models of ψ having $O(2^{N \cdot K})$ (resp., $O(2^N)$) states, $O(N \cdot K)$ (resp., $O(N)$) clocks, and maximum constant $O(K)$. Deciding the emptiness of $\mathcal{A}_\psi$ requires space *logarithmic* in the number of states of $\mathcal{A}_\psi$ and *polynomial* in the number of clocks and in the length of the encoding of K, hence exponential (resp., polynomial) space.

To decide if $\mathcal{L}_T(\mathcal{A}_{\text{sys}}) \subseteq \mathcal{L}_T(\varphi)$, we check whether $\mathcal{L}_T(\mathcal{A}_{\text{sys}}) \cap \mathcal{L}_T(\mathcal{A}_{\neg\varphi}) = \emptyset$ by defining the intersection $\mathcal{A}_\wedge$ of $\mathcal{A}_{\text{sys}}$ and $\mathcal{A}_{\neg\varphi}$, and checking for emptiness of its timed language. The size of $\mathcal{A}_\wedge$ is polynomial in those of $\mathcal{A}_{\text{sys}}$ and $\mathcal{A}_{\neg\varphi}$. Moreover $\mathcal{A}_{\text{sys}}$, $\mathcal{A}_{\neg\varphi}$ and $\mathcal{A}_\wedge$ can be built on the fly, and the emptiness test can be done without explicitly constructing them as well. The next result follows by the observations above and by Theorem 2. The *hardness* of the MC problems derives from the corresponding underlying Future TP problems.

Theorem 3. *The timeline-based MC problem for* MITL *formulas, with simple future trigger rules and non-singular intervals, is* ***EXPSPACE****-complete.*

The timeline-based MC problem for $\mathsf{MITL}_{(0,\infty)}$ *formulas, with simple future trigger rules and intervals in* $\mathit{Intv}_{(0,\infty)}$*, is* ***PSPACE****-complete.*

4 Conclusions

In this paper we have considered timed systems modelled by multi-timelines and studied the problems of model checking against properties expressed by the logics MITL and $\mathsf{MITL}_{(0,\infty)}$, respectively. To solve them, we have exploited an automaton-theoretic construction (using TAs) proving that they are **EXPSPACE**-complete and **PSPACE**-complete, respectively.

In future work we shall investigate model checking combining the interval-based representation of systems given by timelines with interval-based logics (e.g., Halpern-Shoham logic HS, see [13]) for expressing properties.

References

1. Alur, R., Dill, D.L.: A theory of timed automata. Theoretical Computer Science **126**(2), 183–235 (1994). https://doi.org/10.1016/0304-3975(94)90010-8
2. Alur, R., Feder, T., Henzinger, T.A.: The benefits of relaxing punctuality. Journal of the ACM **43**(1), 116–146 (1996). https://doi.org/10.1145/227595.227602
3. Barreiro, J., Boyce, M., Do, M., Frank, J., Iatauro, M., Kichkaylo, T., Morris, P., Ong, J., Remolina, E., Smith, T., Smith, D.: EUROPA: A Platform for AI Planning, Scheduling, Constraint Programming, and Optimization. In: Proc. of ICKEPS (2012)
4. Bozzelli, L., Molinari, A., Montanari, A., Peron, A.: Complexity of timeline-based planning over dense temporal domains: exploring the middle ground. In: Proc. of GandALF. pp. 191–205. EPTCS (2018). https://doi.org/10.4204/EPTCS.277.14
5. Bozzelli, L., Molinari, A., Montanari, A., Peron, A.: Decidability and Complexity of Timeline-based Planning over Dense Temporal Domains. In: Proc. of the Sixteenth International Conference on Principles of Knowledge Representation and Reasoning (KR 2018). AAAI (2018), full version at `https://www.dimi.uniud.it/la-ricerca/pubblicazioni/preprints/1.2018/`
6. Bozzelli, L., Molinari, A., Montanari, A., Peron, A.: Undecidability of future timeline-based planning over dense temporal domains. arxiv.org/abs/1904.09184 (2019)
7. Bozzelli, L., Molinari, A., Montanari, A., Peron, A., Woeginger, G.: Timeline-based planning over dense temporal domains with trigger-less rules is NP-complete. In: Proc. of ICTCS. pp. 116–127. CEUR WP (2018)
8. Cesta, A., Cortellessa, G., Fratini, S., Oddi, A., Policella, N.: An Innovative Product for Space Mission Planning: An A Posteriori Evaluation. In: Proc. of ICAPS. pp. 57–64. AAAI (2007)
9. Chien, S., Tran, D., Rabideau, G., Schaffer, S., Mandl, D., Frye, S.: Timeline-based space operations scheduling with external constraints. In: Proc. of ICAPS. pp. 34–41. AAAI (2010)
10. 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
11. 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
12. Gigante, N., Montanari, A., Cialdea Mayer, M., Orlandini, A.: Complexity of timeline-based planning. In: Proc. of ICAPS. pp. 116–124. AAAI (2017)
13. Halpern, J.Y., Shoham, Y.: A propositional modal logic of time intervals. Journal of the ACM **38**(4), 935–962 (1991). https://doi.org/10.1145/115234.115351
14. Koymans, R.: Specifying real-time properties with metric temporal logic. Real-Time Systems **2**(4), 255–299 (1990). https://doi.org/10.1007/BF01995674
15. Muscettola, N.: HSTS: Integrating Planning and Scheduling. In: Intelligent Scheduling, pp. 169–212. Morgan Kaufmann (1994)