Proceedings of the Twenty-Seventh International Conference on Automated Planning and Scheduling (ICAPS 2017)

Complexity of Timeline-Based Planning

Nicola Gigante, Angelo Montanari
Dept. of Math, Computer Science, and Physics
University of Udine, Italy
gigante.nicola@spes.uniud.it
angelo.montanari@uniud.it

Marta Cialdea Mayer
Dept. of Engineering
University of Roma Tre
Rome, Italy
cialdea@dia.uniroma3.it

Andrea Orlandini
Inst. of Cognitive Science and Technology
National Research Council
Rome, Italy
andrea.orlandini@istc.cnr.it

Abstract

Timeline-based planning is a paradigm that models temporal planning domains as sets of independent, but interacting, components. The behavior of the components can be described by means of a number of state variables whose evolution and interactions over time are governed by a set of temporal constraints. This paradigm is different from the one underlying the common action-based formalisms *à la* PDDL, where the focus is on what can be done by an executive agent. Although successfully used in many real-world applications, little work has been done on the expressiveness and complexity of the timeline-based formalism. The present paper provides a characterization of the complexity of non-flexible timeline-based planning, by proving that a general formulation of the problem is EXPSPACE-complete. Such a result extends a previous work where the same complexity bound was proved for a restricted fragment of timeline-based planning that was shown to be expressive enough to capture action-based temporal planning. In addition, we prove that requiring an upper bound to the solution horizon as part of the input decreases the complexity of the problem, that becomes NEXPTIME-complete.

1 Introduction

In the area of Automated Planning, most of the languages used to represent planning problems, like, for instance, PDDL (Fox and Long 2003; Gerevini et al. 2009), are *action-based* as they focus on which actions can be performed by an executive agent to act on its environment. The purpose of the reasoning process is then that of finding which actions are needed to obtain a given goal. Action-based planning formalisms have been the subject of an extensive theoretical study, from both the expressiveness and the complexity points of view. Classical Planning is known to be PSPACE-complete (Bylander 1994), with a number of tractable fragments (Bäckström et al. 2015), and a variety of extensions have been compared to it, such as planning with temporally extended goals (Bacchus and Kabanza 1998; De Giacomo and Vardi 1999), conditional planning (Rintanen 2004; 2012), and temporal planning (Rintanen 2007). The latter problem, in particular, extends classical planning by reasoning explicitly about time and the *time duration* of actions, and it has been shown to be EXPSPACE-complete.

A different paradigm, known as *Timeline-based Planning*, has been proposed in the literature (Muscettola 1994). It represents temporal planning domains as sets of independent, but interacting, *components*, which are modeled by *state variables* whose evolution over time (the *timelines*) is constrained by a set of temporal constraints. Timeline-based Planning has its origin in the space sector, where it has been successfully applied to many complex real-world cases (Muscettola 1994; Jónsson et al. 2000; Cesta et al. 2007). In these complex applications, timelines provide a suitable way to properly model temporal behaviors of systems composed of a high number of components.

A remarkable research effort has been dedicated to design, build, and deploy software environments for the synthesis of timeline-based planning and scheduling (P&S) applications, including EUROPA (Barreiro et al. 2012), ASPEN (Chien et al. 2000), and APSI-TRF (Cesta et al. 2009). Mostly related to these environments, several attempts have been made to characterize the concept of *timeline*. Cesta and Oddi (1996) propose a domain description language able to suitably represent physical domains in order to solve P&S problems. Frank and Jónsson (2003) present an interval planning paradigm for representing and reasoning about planning domains with time, resources, concurrent activities, and so on. Chien et al. (2012) develop a basic timeline representation to model a set of states, resources, timing, and transition constraints, aiming at generalizing previous efforts made in a number of P&S systems designed for space applications.

All the above contributions aim at properly describing the exploited concepts, languages, and tools, but they do not provide a formally-grounded definition for them. Despite its practical relevance, indeed, the theoretical properties of timeline-based planning have not been systematically investigated yet. In particular, a general picture of computational complexity and expressiveness is still missing, and little work had been done on this front until recently. A formalization of non-flexible timeline-based planning has been given in (Cimatti, Micheli, and Roveri 2013), while the flexible variant has been dealt with in (Cialdea Mayer, Orlandini, and Ubrico 2014) (*flexible* timelines provide a unified framework to reason on both planning and execution under uncertainty). A complete formalization of the planning problem, including flexible timelines and controllabil-

ity issues in timeline-based planning in the presence of uncertainty, can be found in (Cialdea Mayer, Orlandini, and Umbrico 2016). In (Gigante et al. 2016), we identify an EXPSPACE-complete restriction of non-flexible timeline-based planning and we show it to be expressive enough to capture the *action-based* temporal planning problem, providing a first expressiveness comparison between the different approaches. Besides the confinement to *non-flexible* timelines, interpreted over a *discrete* time domain, it forbids the use of *unbounded* interval relations in the definition of temporal constraints, and it does not allow the input to specify an upper bound on the solution *horizon*.

While these restrictions were important for this expressiveness comparison, here we pursue the more general goal of providing a computational complexity characterization of the problem. Thus, in this paper we relax some restrictions by allowing *unbounded interval relations*, showing that they can be included among the syntactic features of the language at no computational cost, as the problem remains EXPSPACE-complete. Subsequently, we allow the specification of an upper bound on the plan horizon as part of the problem input, a feature often used in practice, and we show that this additional constraint *decreases* the computational complexity of the problem, which becomes NEXPTIME-complete. Together, these results proceed towards the completion of the computational complexity picture of non-flexible timeline-based planning over discrete time.

The paper is organized as follows. Section 2 introduces some basic definitions of the considered timeline-based planning problem, and shows that some of its features, that are usually considered as primitives, can be expressed on top of unbounded interval relations. The EXPSPACE-completeness of the problem is then shown in Section 3. Section 4 proves the NEXPTIME-completeness of the problem when we admit the specification of a fixed horizon as part of the input. Finally, Section 5 summarizes the achieved results and discusses future research directions.

2 Timeline-based Planning Problems

In this section, we first introduce notation and terminology of timeline-based planning, mostly borrowed from (Gigante et al. 2016). Then, we show that the addition of unbounded interval relations makes it possible to simplify the formulation of the problem in various respects, thus easing the study of its computational complexity.

Definition 1 (State variable)**.** *A* state variable x *is a triple* (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 value* $v \in V_x$ *to the set of values that* x *can take immediately after* v*;*
- $D_x : V_x \to \mathbb{N} \times \mathbb{N}$ *is a function that maps each* $v \in V_x$ *to a pair* (d_{min}, d_{max})*, with* $d_{min} \leq d_{max}$*, where* d_{min} *and* d_{max} *are respectively the minimum and maximum duration of an interval over which* x *has value* v*.*

To specify the values that a state variable x actually takes over time and the duration of the validity intervals, we make use of the notion of token.

The relation	holds if
$a \leq^{\mathsf{s},\mathsf{s}}_{[l,u]} b$	$l \leq s_b - s_a \leq u$
$a \leq^{\mathsf{e},\mathsf{e}}_{[l,u]} b$	$l \leq e_b - e_a \leq u$
$a \leq^{\mathsf{s},\mathsf{e}}_{[l,u]} b$	$l \leq e_b - s_a \leq u$
$a \leq^{\mathsf{e},\mathsf{s}}_{[l,u]} b$	$l \leq s_b - e_a \leq u$
$a \leq^{\mathsf{s}}_{[l,u]} t$	$l \leq t - s_a \leq u$
$a \geq^{\mathsf{s}}_{[l,u]} t$	$l \leq s_a - t \leq u$
$a \leq^{\mathsf{e}}_{[l,u]} t$	$l \leq t - e_a \leq u$
$a \geq^{\mathsf{e}}_{[l,u]} t$	$l \leq e_a - t \leq u$

Table 1: Interval and time-point relations involving interval $a = (s_a, e_a)$, interval $b = (s_b, e_b)$, and time point $t \in \mathbb{N}$. Bounds $l \in \mathbb{N}$ and $u \in \mathbb{N} \cup \{+\infty\}$ are given to each relation.

Definition 2 (Token)**.** *Let* $x = (V_x, T_x, D_x)$ *be a state variable. A* token *for* x *is a pair* (v, d)*, where* $v \in V_x$*,* $D_x(v) = (d_{min}, d_{max})$*, and* $d_{min} \leq d \leq d_{max}$ *(*$d \in \mathbb{N}$ *is called the* duration *of the token).*

The time-varying behavior of a state variable is represented through a finite sequence of tokens called a *timeline*.

Definition 3 (Timeline)**.** *A* timeline *for a state variable* $x = (V_x, T_x, D_x)$ *is a non-empty finite sequence* $\mathsf{T} = \langle (v_1, d_1), \ldots, (v_k, d_k) \rangle$ *of tokens for* x*, where, for all* $i = 1, \ldots, k-1$*, it holds that* $v_{i+1} \in T(v_i)$*.*

It is worth noticing that the values of a variable x in two consecutive tokens do not need to be different.

Any token $\tau_i = (v_i, d_i)$ in a timeline $\mathsf{T} = \langle \tau_1, \ldots, \tau_k \rangle$ can be associated with a time interval by means of the pair of functions $\mathsf{start\_time}(\tau_i) = \sum_{j=1}^{i-1} d_j$ and $\mathsf{end\_time}(\tau_i) = \mathsf{start\_time}(\tau_i) + d_i$. The end time of the last token of a timeline is called the *horizon* of the timeline. In the following, when there is no ambiguity, we will interchangeably refer to a token and to the time interval associated with it.

The behavior of state variables is constrained by a set of *synchronization rules*, which relate tokens, possibly belonging to different timelines, through temporal relations among intervals or among intervals and time points. Interval and time-point relations that can be exploited in synchronization rules are those commonly used in timeline-based planning formulations, *e.g.*, (Cialdea Mayer, Orlandini, and Umbrico 2016). To express them, we adopt the compact notation proposed in (Gigante et al. 2016). For instance, given two tokens (intervals) a and b, we write $a \leq^{\mathsf{s},\mathsf{s}}_{[l,u]} b$ for a $\mathsf{starts\_before\_start}_{[l,u]}$ b. The set of possible relations between (the endpoints of) a pair of intervals and between (the endpoints of) an interval and a point are given in Table 1 where, for the sake of brevity, we write s_a and e_a instead of, respectively, $\mathsf{start\_time}(a)$ and $\mathsf{end\_time}(a)$.

Definition 4 (Synchronization rules)**.** *Let* Σ *be a finite set of* token names*. An* atom *is an expression of the form* $a\ \rho\ b$ *or* $a\ \rho\ t$*, where* a *and* b *are token names,* $t \in \mathbb{N}$*, and* ρ *is one of*

the temporal relations of Table 1. An existential statement *is a statement of the form:*

$$\exists a_1[x_1 = v_1] \dots a_n[x_n = v_n] \,.\, \mathcal{C}$$

where $n \geq 0$, $\mathcal{C}$ is a conjunction of atoms, $a_1, \dots, a_n$ are token names, $x_1, \dots, x_n$ are state variables, and $v_1, \dots, v_n$ are values from the domains of $x_1, \dots, x_n$, respectively.

A synchronization rule *is a clause of the form:*

$$a_0[x_0 = v_0] \longrightarrow \mathcal{E}_1 \vee \mathcal{E}_2 \vee \dots \vee \mathcal{E}_k, \quad \textit{or}$$
$$\top \longrightarrow \mathcal{E}_1 \vee \mathcal{E}_2 \vee \dots \vee \mathcal{E}_k$$

where a_0 is a token name, x_0 a state variable, v_0 a value from the domain of x_0, and $\mathcal{E}_1, \dots, \mathcal{E}_k$ are existential statements *where only a_0 may appear free.*

The $a_0[x_0 = v_0]$ part of rules of the first form is called the trigger*; rules of the second form are called* trigger-less*.*

A formal account of the semantics of synchronization rules can be found in (Cialdea Mayer, Orlandini, and Umbrico 2016). Intuitively, the left part (the *trigger*) is a universal quantifier, which says that *for all* the tokens a_0, where the variable x_0 takes the value v_0, at least one of the existential statements $\mathcal{E}_i$ must be true. The existential statements in turn assert the existence of tokens $a_1, \dots, a_n$, where the respective state variables take the specified values, that satisfy the interval relations given by $\mathcal{C}$. Trigger-less rules just assert the satisfaction of the existential statements.

A timeline-based planning domain is specified by the set of state variables and the set of synchronization rules representing their admissible behaviors. As shown in (Cialdea Mayer, Orlandini, and Umbrico 2016), the initial conditions and the goal of the problem can be expressed by means of a set of trigger-less rules, and thus we do not need to treat them differently from other kinds of rules, that is, we assume them to be included in the set of synchronization rules.

Definition 5 (Planning problem)**.** *A timeline-based planning* problem *is a pair $\mathcal{P} = (SV, S)$, where SV is a set of state variables and S is a set of synchronization rules involving variables in SV. A* solution plan *π for $\mathcal{P}$ is a set of timelines, one for each state variable in SV, such that the horizon of all the timelines is the same and all the synchronization rules in S are satisfied.*

Some useful relations can be defined on top of the synchronization rules of Definition 4. As an example, the thirteen Allen's ordering relations between pairs of intervals (Allen 1983) can be defined in terms of the basic interval relations of Table 1. In particular, the equality interval relation $a = b$ can be defined as $a \leq^{\mathsf{s},\mathsf{s}}_{[0,0]} b \wedge a \leq^{\mathsf{e},\mathsf{e}}_{[0,0]} b$. Moreover, in the following, any temporal relation devoid of explicit bounds will be provided with the bounds $[0, +\infty]$.

We now show that the formalism can actually be simplified in various respects, without reducing its expressiveness. In (Gigante et al. 2016), we proved that, in the restricted case of timeline-based planning devoid of unbounded interval relations, any problem $\mathcal{P}$ can be rewritten using only binary state variables. It is easy to show that that proof immediately transfers to the general case.

Proposition 1. *Every timeline-based planning problem can be rewritten, with at most a polynomial increase in size, into an equivalent one that only uses binary state variables.*

Additionally, here we show that some syntactic features of synchronization rules, as formulated in Definition 4, can be treated as syntactic sugar.

Theorem 1. *Every timeline-based planning problem $\mathcal{P} = (SV, S)$ can be rewritten, with at most a polynomial increase in size, into an equivalent problem $\mathcal{P}' = (SV', S')$ such that: (i) it does not make use of time-point relations, and (ii) it does not make use of trigger-less rules.*

Proof. The thesis follows from Lemmata 1 and 2 below. □

Lemma 1. *Every timeline-based planning problem $\mathcal{P} = (SV, S)$ can be rewritten, with at most a polynomial increase in size, into an equivalent one that does not make use of time-point relations.*

Proof. To rewrite time-point relations, we add a fresh variable $\overline{x}$, with domain $V_{\overline{x}} = \{0, 1\}$, duration function $D_{\overline{x}}(0) = D_{\overline{x}}(1) = (1, 1)$ and transition function $T(0) = T(1) = \{1\}$. The timeline for the variable $\overline{x}$ can only contain a chain of unit-sized tokens all holding $\overline{x} = 1$, throughout the entire timeline, excepting the first token which can hold $\overline{x} = 0$. By asking for the existence of such a token, we can position other tokens relatively to the start of the timeline without using time-point relations. Thus, all existential statements of the following form:

$$\exists \dots a_i[\dots] \,.\, a_i \leq^{\mathsf{x}}_{[l,u]} t \wedge \dots, \qquad \text{or}$$
$$\exists \dots a_i[\dots] \,.\, a_i \geq^{\mathsf{x}}_{[l,u]} t \wedge \dots$$

can be replaced respectively as follows:

$$\exists \dots a_i[\dots]\overline{a}[\overline{x} = 0] \,.\, \overline{a} \leq^{\mathsf{s},\mathsf{x}}_{[t-\overline{u},t-l]} a_i \wedge \dots, \qquad \text{or}$$
$$\exists \dots a_i[\dots]\overline{a}[\overline{x} = 0] \,.\, \overline{a} \leq^{\mathsf{s},\mathsf{x}}_{[t+l,t+u]} a_i \wedge \dots$$

where $\overline{u} = \min(t, u)$ and $t + \infty = +\infty$. It is worth noticing that the existence of the anchoring token is not requested by a trigger-less rule, since we want to avoid them as well, as shown by Lemma 2 below. Rather, its existence is ensured by the existential statements of each translated rule. □

Lemma 2. *Every timeline-based planning problem $\mathcal{P} = (SV, S)$ can be rewritten, with at most a polynomial increase in size, into an equivalent one that does not make use of trigger-less synchronization rules.*

Proof. The purpose of trigger-less rules is to require the existence of some tokens regardless of any other one. To simulate a trigger-less rule with a triggered one, it is sufficient to use a trigger that is satisfied for sure at least once in the solution. Thus, let $\overline{x}$ be the auxiliary state variable from the proof of Lemma 1. A trigger-less rule of the form $\top \longrightarrow \mathcal{E}_1 \vee \dots \vee \mathcal{E}_k$ can be replaced by the equivalent rule:

$$\overline{a}[\overline{x} = 0] \longrightarrow \mathcal{E}_1 \vee \dots \vee \mathcal{E}_k$$
$$\overline{a}[\overline{x} = 1] \longrightarrow \mathcal{E}_1 \vee \dots \vee \mathcal{E}_k$$

Note that the rule can be possibly triggered multiple times, but this does not change the meaning of the rules, as they simply ask for the existence of the tokens described in the existential statements. □

3 Complexity of Timeline-based Planning with Unbounded Interval Relations

In this section we provide an exponential space algorithm to solve timeline-based planning problems as defined in Section 2, thus proving that the problem belongs to EXPSPACE. The hardness of the problem for the same class follows directly from (Gigante et al. 2016, Theorem 1), since the extension considered here is a direct generalization. Hence, the problem turns out to be EXPSPACE-complete.

The proposed algorithm is a nondeterministic guess-and-check procedure, running in nondeterministic exponential space, that gives us also a deterministic exponential space procedure thanks to the well known fact that EXPSPACE = NEXPSPACE. The procedure, outlined in Theorem 2, nondeterministically guesses a solution of at most doubly exponential size, and then is able to check its correctness using a singly exponential amount of space. This works thanks to Lemma 3, a *small model* result showing that any satisfiable problem has a solution at most doubly exponentially long. The exponential space algorithm shown in (Gigante et al. 2016) for the restricted problem works in a similar way, but the proof of the small model result cannot be directly adapted to the case of unbounded interval relations.

Let $\mathcal{P} = (SV, S)$ be a timeline-based planning problem and let $|\mathcal{P}|$ be the size of the input representation of $\mathcal{P}$. W.l.o.g., by Proposition 1 and Theorem 1, we can assume that $\mathcal{P}$ only makes use of binary variables and features neither trigger-less rules nor time-point relations. W.l.o.g., we also assume that all the token names used in synchronization rules in S are unique, and call $\mathcal{N}$ the set of such names.

A solution plan for $\mathcal{P}$, as a set of timelines, describes separately the evolution of each variable on its own. It will be useful, however, to reason about the solution as a whole, as a sequence of *events* which mark the time points when something is changing, *i.e.*, when some token ends and the following one starts.

Definition 6 (Events)**.** *Let $<$ be an arbitrary ordering over state variables. An* event sequence *σ associated with a solution plan π is a sequence of* events *$e_1, \ldots, e_m$, where each event is a tuple of the form $e_i = (t_i, f_i, x_i, v_i)$, where $t_i \in \mathbb{N}$, $f_i \in \{\mathsf{start}, \mathsf{end}\}$, $x_i \in SV$, and $v_i \in V_x$. The events in the sequence represents the starting and ending endpoints of all the tokens in π, as follows:*

- *for each token τ in a timeline of π, there are two events e_i and e_j, with $i < j$, $f_i = \mathsf{start}$, and $f_j = \mathsf{end}$;*
- *$t_0 = 0$ and for each $1 \leq i \leq m$:*

$$\mathsf{time}(e_i) = \sum_{j=0}^{i} t_j = \mathsf{start\_time}(\tau_j) \quad \text{if } f_i = \mathsf{start},$$

$$\mathsf{time}(e_i) = \sum_{j=0}^{i} t_j = \mathsf{end\_time}(\tau_j) \quad \text{if } f_i = \mathsf{end},$$

  where τ_j is the token associated with the event e_j;
- *for each pair of events e_i and e_j, if $\mathsf{time}(e_i) = \mathsf{time}(e_j)$ and $x_i < x_j$, then $i < j$.*

An event sequence is a flattened representation of the timelines in π, with the endpoints of each token placed on the same line and timestamped by the *difference* in time with regards to the *previous* event. The third condition in Definition 6 ensures that there is only one canonical sequence to represent a given solution. Moreover, using relative rather than absolute timestamps is needed to keep the memory footprint of an event sequence under control in Theorem 2.

Definition 7. *Given a set of events $E = \{e_{i_1}, \ldots, e_{i_k}\}$ from an event sequence σ, the* span *of E is the maximum distance in time between two events of E in σ.*

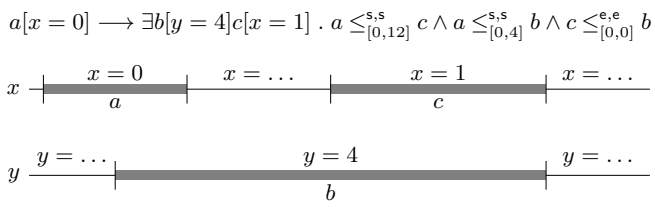
Let π be a solution plan for $\mathcal{P}$ and σ be its associated event sequence. The main tool in the proof of Lemma 3 will be the *witness set* of π (see Definition 8), namely, a set of graphs that describes the way in which the solution satisfies the constraints of the problem. This additional structure, which is built on top of the event sequence, will be useful to look at the problem from a more abstract perspective. In particular, the witness set will allow us to keep under control the combinatorial complexity of the problem. We will indeed show that, among all the witness graphs of π, we can focus on a very small number of non-isomorphic ones. Such a number is bounded above by the number and the size of the synchronization rules of $\mathcal{P}$.

Let τ be a token from a timeline in π, let $\mathcal{R}_\tau$ be the set of all the synchronization rules triggered by the existence of τ, and let ξ_τ be the set of existential statements belonging to rules in $\mathcal{R}_\tau$ and *satisfied* by π.

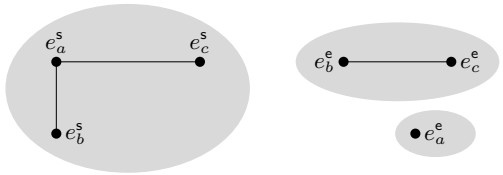
Definition 8 (Witness graph)**.** *Given an existential statement $\mathcal{E} \in \xi_\tau$, the* witness graph *of $\mathcal{E}$ over τ is a* labeled undirected *graph $G_{\mathcal{E},\tau} = (V, E, \lambda)$, where $V \subseteq \sigma$ is a set of events, and $\lambda : V \to \mathcal{N}$ is a labeling function assigning a token name to each event. The graph is built as follows.*

1. *Given the events e^{s}_τ, e^{e}_τ representing the start and the end of τ, $e^{\mathsf{s}}_\tau, e^{\mathsf{e}}_\tau \in V$.*
2. *Let $\mathcal{C}$ be the conjunction of atoms of $\mathcal{E}$. Since $\mathcal{E}$ is satisfied, there must be a mapping between the atoms in $\mathcal{C}$ and the events in σ that witnesses this satisfaction, mapping each atom $a \leq^{\mathsf{x},\mathsf{y}}_{[l,u]} b$ to two events representing the x and y endpoints of the tokens that interpret a and b. For each pair of events e_i and e_j belonging to the image of such mapping, let e'_i and e'_j be the events representing the opposite endpoints of the tokens of e_i and e_j. Then $e_i, e_j, e'_i, e'_j \in V$, and the labeling function λ assigns each event to the corresponding token names,* i.e., *$\lambda(e_i) = \lambda(e'_i) = a$ and $\lambda(e_j) = \lambda(e'_j) = b$.*
3. *Given the events e_i and e_j of the previous point, there is an edge between them if and only if either they are the endpoints of the same token, or the atom satisfied by the two events uses a bounded interval relation,* i.e., *of the form $a \leq^{\mathsf{x},\mathsf{y}}_{[l,u]} b$ where $u \neq +\infty$.*

Definition 9 (Witness Components)**.** *Given a witness graph G built on a token from π, consider the decomposition of G into* connected components *$g_1, \ldots, g_m$. Each g_i is called a* witness component *of π.*



(a) Set of timelines satisfying the example rule.



(b) The witness graph $G_{\mathcal{E},a}$ from token a of Figure 1a. Highlighted in gray are its three witness components.

Figure 1: An example of a *witness graph*.

The *witness set* of π, denoted by $\mathcal{G}_\pi$, is the set of all the witness graphs that can be built from the tokens of π. An example of set of timelines and a corresponding witness graph is shown in Figure 1. The single witness components represent the patterns of events that can appear in the solution, and are bound together by bounded interval relations. By grouping events in this way we abstract from the high number of possible ways in which a single group of bounded atoms can be satisfied, and we can handle the satisfaction of unbounded atoms by viewing them as unbounded temporal constraint between the whole witness components.

Definition 10 (Related events)**.** *If e_i and e_j are two events from a witness graph $G_{\mathcal{E},\tau}$, then e_i and e_j are said to be* related *if the two events are the endpoints of the same token, or if they satisfy an* unbounded *atom,* i.e.*, one of the form $a \leq^{x,y}_{[l,+\infty]} b$, contained in $\mathcal{E}$. If e_i must happen before (after) e_j, then e_i is* left-related *(right-related) to e_j. A witness component is (left-/right-) related to another if any of its events are (left-/right-) related to any of the events of the other.*

The structure of a given witness graph gives us useful information about the *locality* of the problem, *i.e.*, how many different pieces of a solution can interfere with the satisfaction of a single rule, and which is the size of such pieces.

Definition 11 (Window)**.** *Given a problem $\mathcal{P} = (SV, S)$, the* window *of $\mathcal{P}$, denoted $w(\mathcal{P})$, is the product of all the finite coefficients greater than zero that appear as bounds in the interval relations of synchronization rules in S.*

Proposition 2. *The span of the events belonging to a witness component is at most $w(\mathcal{P})$.*

Proof. Let g_i be a witness component of π, from a witness graph G. For each pair of adjacent nodes (events) e_i and e_j, the distance in time between e_i and e_j is bounded by some coefficient $u_{i,j}$, *i.e.*, the upper bound appearing in the corresponding satisfied atom. So the distance between any two non adjacent events cannot be greater than the sum of all the bounds $u_{k,l}$ for each adjacent e_k and e_l, which is less than the *product* of such bounds, *i.e.*, less than $w(\mathcal{P})$. □

The grouping of events into the connected components of the witness graphs allows us to reason about the combinatorial aspects of the problem from a coarser point of view.

Definition 12 (Witness class)**.** *Let W be a set of witness components. By $W^\sim$ we denote the quotient of W over the labeled graph isomorphism relation. Any element $[g] \in W^\sim$ is called a* witness class *of W.*

Proposition 3. *Let W be a set of witness components. The number of witness classes of W is less than the size of the input problem. In other words:*

$$|W^\sim| < |\mathcal{P}|$$

Proof. Given how witness graphs are constructed in Definition 8, it can be seen that the structure of the graph is a direct consequence of the syntactic structure of the rules. In particular, all the witness graphs built from the same existential statement are isomorphic to each other, hence the different connected components in each are also determined. At most, we can have two witness components for each token name, and each token name is used at least twice in describing $\mathcal{P}$, so the total number must be less than $|\mathcal{P}|$. □

We can say that a problem is *connected* if all the witness graphs in any possible solution plan are always *connected*, *i.e.*, with only one connected component each. This property was defined in (Gigante et al. 2016) in a different, more syntactic way by looking at the structure of the synchronization rules, but the two definitions coincide in the restricted case with only bounded relations. Then, only connected problems were admitted, while here the general case is considered.

The size of the window $w(\mathcal{P})$ is the fundamental building block to reason about the length of a solution, and forms the basis of our argument. As a first step, it is useful to count how many different combinations of different subsegments of a solution can fit into a time span of $w(\mathcal{P})$ time steps.

Proposition 4. *Let π be a solution for the problem $\mathcal{P}$, and σ be its associated event sequence. The number of possible subsequences of σ of span $k > 0$ is at most:*

$$2^{2k|\mathcal{P}|}$$

Proof. Consider $\mathcal{P} = (SV, S)$ and let σ_w be a subsequence of σ of span k. Such sequence of events represents the evolution of the state variables in SV in a time interval of k time steps. In addition to the values of the variables, we need to also keep track of the start and the end of the tokens, even when a token begins with the same variable value as the preceding one, as this is a different situation than a single longer token. Thus, to represent the state of the variables at a single time step we need a *state word* of $2|SV|$ bits, two for each variable: one for the value of the variable and one used to

mark the boundaries of the tokens, by flipping between zero and one each time a token ends and the subsequent one begins. There are $2^{2|SV|}$ possible state words, and the number of a possible sequences of k state words is thus:

$$\left(2^{2|SV|}\right)^k = 2^{2k|SV|} < 2^{2k|\mathcal{P}|}$$

Observe that the number of possible valid subsequences of an event sequence spanning k time steps cannot be greater than the possible combinations of k state words. □

After proving a couple of simple facts that will be used later, Lemma 3 will show the main fact that will allow us to devise an exponential-space algorithm for the problem, *i.e.*, a doubly exponential bound on the span of the solution.

Proposition 5. *Consider a finite non-empty set Γ, a word $w \in \Gamma^*$, and an integer $m > 0$. If $|w| \geq (|\Gamma| + 1)^m$, then there is an element $\gamma \in \Gamma$ repeating at least 2^m times in w.*

Proof. Proceeding by induction on m, in the base case where $m = 0$, if $|w| > 1$ we are trivially sure to find an element which appears a single time. If $m > 0$, and $|w| \geq (|\Gamma| + 1)^m$, then w is formed by the juxtaposition of $k = |\Gamma| + 1$ blocks $w_1, \ldots, w_k$ of length $(|\Gamma| + 1)^{m-1}$. In each w_i, by the inductive hypothesis, we find an element σ_i which repeats 2^{m-1} times in w_i. But having $|\Gamma| + 1$ such blocks, there must be at least two i and j such that $\gamma_i = \gamma_j$, which thus globally repeats $2 \cdot 2^{m-1} = 2^m$ times in w. □

Proposition 6. *Let Σ be a finite set, and let $\langle \Sigma_1, \ldots, \Sigma_m \rangle$ be a sequence of subsets of Σ,* i.e., *$\Sigma_i \subseteq \Sigma$ for all $1 \leq i \leq m$. If $m \geq (2^{|\Sigma|}+1)^2$, there exist three integers $0 \leq i, j, k \leq m$ such that $\Sigma_j \subseteq \bigcup_{\ell=i}^{j-1} \Sigma_\ell$ and $\Sigma_j \subseteq \bigcup_{\ell=j+1}^{k} \Sigma_\ell$.*

Proof. By Proposition 5, after $(2^{|\Sigma|} + 1)^2$ elements in the sequence we are sure to find four repetitions of the same subset $\Sigma_i \subseteq \Sigma$. Chosing the middle one of any three of them is sufficient to find the needed subset. □

Lemma 3. *Let $\mathcal{P} = (SV, S)$ be a timeline-based planning problem. If the problem admits a solution, then there exists a solution such that the span of its associated event sequence is at most doubly exponentially long.*

Proof. Denote $w(\mathcal{P})$ as w, let $n = |\mathcal{P}|$. Let π be a solution with associated event sequence σ, W the set of witness components of $\mathcal{G}_\pi$, and Γ the set of all the possible subsequences of span $k = 2w$. Now, observe that $w < 2^n$ and $k < 2^{n+1}$, and that $|\Gamma| \leq 2^{2kn}$, by Proposition 4. Now, let $m = (2^{|W^\sim|} + 1)^2 + 1$, and suppose that σ spans more than $\beta(\mathcal{P}) = k(|\Gamma| + 1)^{\lceil \log_2 m \rceil}$ time steps. It can be verified that $\beta(\mathcal{P}) \in \mathcal{O}(2^{2^n})$, since $|\Gamma| \in \mathcal{O}(2^{2^n})$ and $m \in \mathcal{O}(2^n)$.

Then, by Proposition 5 we know there must be at least m repeated subsequences $\sigma_1, \ldots, \sigma_m$, each of span k, centered over m time points $t_1, \ldots, t_m$. Now, let $W_{i,j}$ be the set of all the witness components of any witness graph from $\mathcal{G}_\pi$ completely included between $t_i - w$ and $t_j + w$. Then, Proposition 6 implies that the number m of such repetitions is enough to ensure that we can find four repeated subsequences $\sigma_i, \sigma_j, \sigma_k$ and σ_h, centered at the corresponding time points $t_i < t_j < t_k < t_h$, such that $W^\sim_{j,k} \subseteq W^\sim_{i,j}$ and $W^\sim_{j,k} \subseteq W^\sim_{k,h}$, *i.e.*, for any witness component between σ_j and σ_k, there are other two isomorphic witness components, one between σ_i and σ_j, and one between σ_k and σ_h.

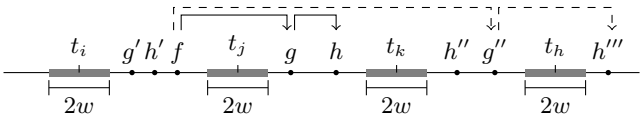


Figure 2: Situation described in the proof of Lemma 3. Gray segments are the four repeated subsequences. The contraction cuts the events between t_j and t_k. Some relation between components is broken (solid arrows), but can be recomposed (dashed arrows).

Being this the case, we can now claim that removing from σ every event included between j and k would lead to a shorter event sequence σ' associated to a shorter solution plan π'. To show this, we have to show that any synchronization rule, triggered by anything outside t_j and t_k, that was satisfied by something inside the cut segment, will still be satisfied after the cut. The situation is pictured in Figure 2. First of all, observe that the cut respects the structure of the single witness components, *i.e.*, the cut cannot result into partially built witness components. To see this, consider a witness component g that contains at least an event between t_j and t_k. Either *all* its events are contained in the cut, and thus it is going to disappear completely, or it spans across t_j (or t_k). In the latter case, we know from Proposition 2 that the events of g cannot span outside σ_j (or σ_k). But σ_j and σ_k are equal, so all the events on the left (right) of t_j can be substituted by those on the left (right) of t_k. In other words, any bounded atom satisfied before the cut will be satisfied after it, and we only have to reason about which witness components are removed from the solution, as a whole, without care about the single events. Note that this also implies that the cut respects the structure of single tokens, *i.e.*, the parenthesis structure of start/end events is not disrupted, and the duration function of the tokens, as the relation between the endpoints of a token is represented by an edge in the witness graph exactly as another kind of bounded relation. With a similar reasoning, it can be seen that the cut cannot result into violations of the transition functions.

Now, consider a witness graph $G_{\mathcal{E},\tau}$ with some witness component $g \in W_{j,k}$, with $\mathcal{E}$ its original existential statement. We have to ensure that, after the cut, $\mathcal{E}$ is still satisfied. Observe that there are not one but two isomorphic copies of g, *i.e.*, $g' \in W_{i,j}$ and $g'' \in W_{k,h}$, hence the simple *existence* of the events required by $\mathcal{E}$, and the satisfaction of bounded atoms between them, are ensured. However, some of the components outside the cut might be left- or right-related with g, and we must ensure that some other component can be used, instead, to satisfy $\mathcal{E}$. Thus, consider any other component f of $G_{\mathcal{E},\tau}$ that was related to g. After removing g, either g'' or g' can satisfy the unbounded atoms that were satisfied by f and g. Note that the presence of two copies of g is important. If we only had one copy, the cut

may not preserve some unbounded relation if g' was placed between g and its only copy (like f which is between g and g' in Figure 2). Looking at the relation between only two components at the time might not be sufficient, though, since the whole $G_{\mathcal{E},\tau}$ must be reconstructed. In particular, relations between two removed components are not necessarily satisfied by their surrounding copies alone (*e.g.*, the relation between g and h in Figure 2). However, since the copies are part of their own complete witness graphs isomorphic to $G_{\mathcal{E},\tau}$, it can be shown that there always exist other copies suitable to reconstruct $G_{\mathcal{E},\tau}$ (*e.g.*, h''' in Figure 2).

Thus, provided σ is longer than the doubly exponential upper bound shown above, we can remove everything from t_i to t_j and still obtain a valid solution event sequence. □

It is now possible to prove the final result, by giving an exponential-space algorithm to solve the problem, thus proving it to be EXPSPACE-complete.

Theorem 2. *Let $\mathcal{P}$ be a timeline-based planning problem. The problem of finding whether there exists a solution plan for $\mathcal{P}$ belongs to* EXPSPACE.

Proof. We will sketch an algorithm that can be executed by a *nondeterministic* Turing machine using exponential space. Then the thesis will follow from the well-known fact that NEXPSPACE = EXPSPACE. The algorithm at first nondeterministically guesses and keeps in memory a sequence of events δ_k of span $k = 2w$, with $\mathsf{time}(e_1) = 0$, that will be used as a scrolling window over the solution. A counter t is initialized to w and will count up to the doubly exponential upper bound of Lemma 3. If t reaches its maximum value and no solution has been accepted yet, the algorithm returns a negative answer. At each step, an internal check is done to ensure that the current sequence δ_w does not contain violations to the synchronization rules of the problem. To perform this check, we keep track of all the possible witness components that were found in the solution before the current window, including which left- or right-relations were satisfied among them, and of which classes of witness components are required to be found in the future.

Thus, the algorithm keeps a set $P \in 2^{W^\sim}$ of subsets of $W^\sim$, such that a set of witness classes J belongs to P if and only if an instance of each witness class has been found, satisfying all the left- and right-relations between them. Other two similar sets F and C are kept to track which sets of components have been found in the current window and which are still pending. Then, at each step, the algorithm proceeds as follows. For each subset J of $W^\sim$, a mapping is nondeterministically guessed between the token names used in the corresponding existential statements and the events in the current window δ_k. If the guessed events correctly compose the components in J while satisfying all the left- and right-relations among them, then J is added to C, otherwise the computation branch is rejected. After this stage of collection of components, all the elements of C are removed from F.

Then, each token τ contained in δ_w is considered. For each rule triggered by τ, an existential statement $\mathcal{E}$ is guessed, and a similar procedure as before is used to look for witness components of $G_{\mathcal{E},\tau}$ in the current window. If the witness component which mentions τ (if any) is not correctly present in the current window, the branch is rejected. Then, a set J is guessed of other components contained in the current window preserving all the required left- and right-relations among them. Among the missing witness components, if all of them are not either *only* left- or only right-related to the components in J, the branch is rejected, since they should have been found in the current window. Otherwise, if all of them are left-related to the components in J, then they should have appeared before: if $J \notin P$, the branch is rejected. Finally, if all of them are right-related to the components in J, J is added to F for the next step.

When all the tokens of the current window have been checked, elements from C are added to P, t is incremented, and a new event is guessed and added to the end of the sequence, while the first event of the sequence is discarded if the span of δ_w become greater than k. If, after any step, the set F is found empty, the computation is accepted. It can be verified that this procedure answers correctly using exponential space, as the set F, P, C are exponential in size. □

4 Complexity of Timeline-based Planning with Constrained Horizon

In this section, we consider the timeline-based planning problem where an upper bound on the size of the solution, usually called *horizon*, is included in the input. This is a common component in usual formulations of timeline-based planning problems (Cialdea Mayer, Orlandini, and Umbrico 2016, for example), because real-world applications often need control over the time needed to fulfill the goal (unconstrained planning is often unrealistic).

Definition 13 (Timeline-based problems with horizon)**.** *A Timeline-based planning problem* with horizon *is a pair $\mathcal{H} = (\mathcal{P}, h)$ where $\mathcal{P}$ is a timeline-based problem and $h \in \mathbb{N}$. A solution plan π for $\mathcal{H}$ is a solution plan for $\mathcal{P}$ whose horizon is at most h.*

Expressing a maximum length for the solution as part of the input significantly changes the expressive power and computational complexity of the problem. In fact, as shown in (Gigante et al. 2016) and in the previous section, timeline-based planning problems can encode problems that admit doubly-exponentially long plans. However, if we restrict the problem to those solutions of length at most h, the problem loses all the solutions more than exponentially long with regards to $|\mathcal{P}|$. This implies the following two facts.

Proposition 7. *Any timeline-based problem with horizon $\mathcal{H} = (\mathcal{P}, h)$ can be rewritten into another equivalent one, with at most a polynomial increase in size, which does not make use of* unbounded *interval relations.*

Proof. Any atom made of an unbounded interval relation, *i.e.*, of the form $a \leq^{\mathsf{x},\mathsf{y}}_{[l,+\infty]} b$ can be replaced by one of the form $a \leq^{\mathsf{x},\mathsf{y}}_{[l,h]} b$, without loosing any solution. □

Proposition 8. *Given a timeline-based planning problem with horizon $\mathcal{H}$, the problem of finding whether there exists a solution plan for $\mathcal{H}$ belongs to* NEXPTIME.

Proof. The same nondeterministic algorithm proposed in the proof of Theorem 2 can be used to solve horizon-constrained problems. Note, in fact, that each single step of the algorithm runs in nondeterministic exponential time. Thus, by looking for solutions of length at most h instead of the doubly exponential bound of Lemma 3, we obtain an algorithm of nondeterministic exponential running time, as only an exponential number of steps is done. □

Proposition 7 essentially tells us that unbounded interval relations are trivial syntactic sugar in the horizon-constrained case, on the contrary of what we saw earlier for the unconstrained case. Proposition 8 shows the unsurprising fact that adding an horizon causes the computational complexity to decrease. We will now show that this upper bound is tight by showing that the problem is hard for NEXPTIME with a reduction from a classic tiling problem.

Definition 14 (Exponentially bounded tiling problem)**.** *Consider the tuple $\mathcal{T} = (T, t_0, H, V, n)$, consisting in a set T of* tiles*, an* initial *tile $t_0 \in T$, two binary relations $H, V \in T \times T$, specifying the horizontal and vertical tiling constraints, and a positive $n \in \mathbb{N}^+$, encoded in binary. Let $[n] = \{1, \ldots, n\}$. Then, the* exponentially bounded tiling problem *is the problem of finding whether there exists a tiling $f : [n] \times [n] \to T$ of the square of size $n \times n$, with t_0 in the origin, that respects the tiling constraints,* i.e.*:*

- $f(0, 0) = t_0$
- *for all $x \in [n-1]$ and $y \in [n]$, $f(x, y)$ H $f(x+1, y)$*
- *for all $x \in [n]$ and $y \in [n-1]$, $f(x, y)$ V $f(x, y+1)$*

Theorem 3. *Given a timeline-based planning problem with horizon $\mathcal{H}$, the problem of finding whether there exists a solution plan for $\mathcal{H}$ is* NEXPTIME*-complete.*

Proof. The proof consists in a reduction from the exponentially bounded tiling problem of Definition 14, which is known to be NEXPTIME-complete (Johnson 1990). Let $\mathcal{T} = (T, t_0, H, V, n)$ be an instance of the problem. We build a suitable timeline-based planning problem with horizon $\mathcal{H} = (\mathcal{P}, h)$, with $\mathcal{P} = (SV, S)$, such that $\mathcal{T}$ admits a tiling if and only if $\mathcal{H}$ admits a solution plan.

The timeline-based problem has a single state variable x with domain $V_x = T$, *i.e.*, one possible value for each tile type in T. The transition function is $T_x(v) = V_x$, for each $v \in V_x$, and the duration function constrains every token to have unitary duration, that is, $D_x(v) = (1, 1)$ for each $v \in V_x$. By setting the horizon $h = n^2$, the values of t over time represent the tilings of the $n \times n$ square in a row-major layout. The synchronization rules can encode the tiling constraints as follows. First of all, the initial tile is put into place:

$$\top \longrightarrow \exists a[x = t_0] \,.\, a \leq^{\mathsf{s}}_{[0,0]} 0$$

The horizontal tiling relation H is represented by the following rules. For each tile $t \in T$, there is a rule for the consistency of the H relation on the right:

$$a[x = t] \longrightarrow a \leq^{\mathsf{s}}_{[0,0]} n^2 \vee \bigvee_{tHt'}^{t' \in T} \exists b[x = t'] \,.\, a \leq^{\mathsf{s,s}}_{[1,1]} b$$

and one on the left:

$$a[x = t] \longrightarrow a \leq^{\mathsf{s}}_{[0,0]} 0 \vee \bigvee_{t'Ht}^{t' \in T} \exists b[x = t'] \,.\, b \leq^{\mathsf{s,s}}_{[1,1]} a$$

These rules handle the satisfaction of the horizontal constraints in both directions. Both handle the special case for respectively the first/last tile, which cannot have anything on the right/left side. The encoding of vertical tiling constraints is similar. For each $t \in T$, the following rules are added:

$$a[x = t] \longrightarrow a \leq^{\mathsf{s}}_{[0,n]} n \vee \bigvee_{tVt'}^{t' \in T} \exists b[x = t'] \,.\, a \leq^{\mathsf{s,s}}_{[n,n]} b$$

$$a[x = t] \longrightarrow a \leq^{\mathsf{s}}_{[0,n]} n^2 \vee \bigvee_{t'Vt}^{t' \in T} \exists b[x = t'] \,.\, b \leq^{\mathsf{s,s}}_{[n,n]} a$$

It can be verified that this timeline-based problem with horizon correctly encodes the original tiling problem. Moreover, the encoding can be produced in polynomial time, since it produces a polynomial-sized problem and it only involves trivial loops over the elements of the tiling set T and the relations H and V. □

5 Conclusions and future work

In this paper, we showed that the computational complexity of a general formulation of the non-flexible timeline-based planning problem over discrete time is EXPSPACE-complete. This result extends the one given in (Gigante et al. 2016) for a restricted variant of the problem, excluding unbounded interval relations, which was shown to be already EXPSPACE-complete. The good news is that the addition of unbounded interval relations does not increase the computational complexity of the problem, despite the greater modeling flexibility it provides. A more general formulation, that removes the specification of bounds on the *length* of tokens, can also be considered. However, it has not been treated here because it can be shown that this additional feature can be used to force a solution of non-elementary length.

In Section 4, we also proved that the problem becomes NEXPTIME-complete when we require the input to specify an upper bound on the solution horizon, as it happens in many application contexts, closing another question that was left open by the aforementioned work.

Future work should try to complete the complexity picture for timeline-based planning, *e.g.*, by studying the problem with the addition of *flexible* timelines (timelines with temporal uncertainty over token endpoints) and/or by looking at *dense* or *continuous* time domains. Other open theoretical questions about the timeline-based planning paradigm concern the expressiveness of the formalism, both in comparison to other planning paradigms and from a logical perspective. In (Gigante et al. 2016), we showed that non-flexible timelines can capture action-based temporal planning, but no comparison has been done with other extensions of classical planning, *e.g.*, adding uncertainty. Moreover, the formalism lacks a temporal logic counterpart, unlike *e.g.*, classical planning, which has been shown to be captured by Linear Temporal Logic in (Cialdea Mayer et al. 2007).

Acknowledgments

The work has been partially supported by the iNdAM GNCS project "Logic and Automata for Interval Model Checking". Andrea Orlandini is also partially supported by the European Commission within the FOURBYTHREE project, GA No. 637095.

References

Allen, J. F. 1983. Maintaining Knowledge about Temporal Intervals. *Communications of the ACM* 26(11):832–843.

Bacchus, F., and Kabanza, F. 1998. Planning for Temporally Extended Goals. *Annals of Mathematics in Artificial Intelligence* 22(1-2):5–27.

Bäckström, C.; Jonsson, P.; Ordyniak, S.; and Szeider, S. 2015. A Complete Parameterized Complexity Analysis of Bounded Planning. *Journal of Computer and System Sciences* 81(7):1311–1332.

Barreiro, J.; Boyce, M.; Do, M.; Frank, J.; Iatauro, M.; Kichkaylo, T.; Morris, P.; Ong, J.; Remolina, E.; Smith, T.; and Smith, D. 2012. EUROPA: A Platform for AI Planning, Scheduling, Constraint Programming, and Optimization. In *Prof. of the 4th International Competition on Knowledge Engineering for Planning and Scheduling*.

Bylander, T. 1994. The Computational Complexity of Propositional STRIPS Planning. *Artificial Intelligence* 69(1-2):165–204.

Cesta, A., and Oddi, A. 1996. DDL.1: A Formal Description of a Constraint Representation Language for Physical Domains. In Ghallab, M., and Milani, A., eds., *New Directions in AI Planning*. IOS Press. 341–352.

Cesta, A.; Cortellessa, G.; Fratini, S.; Oddi, A.; and Policella, N. 2007. An Innovative Product for Space Mission Planning: An A Posteriori Evaluation. In *Proc. of the 17th International Conference on Automated Planning and Scheduling*, 57–64.

Cesta, A.; Cortellessa, G.; Fratini, S.; and Oddi, A. 2009. Developing an End-to-End Planning Application from a Timeline Representation Framework. In *Proc. of the 21st Conference on Innovative Applications of Artificial Intelligence (IAAI-09)*, 66–71.

Chien, S.; Rabideau, G.; Knight, R.; Sherwood, R.; Engelhardt, B.; Mutz, D.; Estlin, T.; Smith, B.; Fisher, F.; Barrett, T.; Stebbins, G.; and Tran, D. 2000. ASPEN - Automated Planning and Scheduling for Space Mission Operations. In *Proc. of the 8th International Conference on Space Operations*.

Chien, S. A.; Johnston, M.; Frank, J.; Giuliano, M.; Kavelaars, A.; Lenzen, C.; and Policella, N. 2012. A Generalized Timeline Representation, Services, and Interface for Automating Space Mission Operations. In *Proc. of the 12th International Conference on Space Operations*.

Cialdea Mayer, M.; Limongelli, C.; Orlandini, A.; and Poggioni, V. 2007. Linear Temporal Logic as an Executable Semantics for Planning Languages. *Journal of Logic, Language and Information* 16(1):63–89.

Cialdea Mayer, M.; Orlandini, A.; and Ubrico, A. 2014. A Formal Account of Planning with Flexible Timelines. In *Proc. of the 21st International Symposium on Temporal Representation and Reasoning*, 37–46.

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

Cimatti, A.; Micheli, A.; and Roveri, M. 2013. Timelines with Temporal Uncertainty. In *Proc. the 27th AAAI Conference on Artificial Intelligence*.

De Giacomo, G., and Vardi, M. Y. 1999. Automata-Theoretic Approach to Planning for Temporally Extended Goals. In *Proc. of the 5th European Conference on Planning*, 226–238.

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

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

Gerevini, A.; Haslum, P.; Long, D.; Saetti, A.; and Dimopoulos, Y. 2009. Deterministic Planning in the Fifth International Planning Competition: PDDL3 and Experimental Evaluation of the Planners. *Artificial Intelligence* 173(5-6):619–668.

Gigante, N.; Montanari, A.; Cialdea Mayer, M.; and Orlandini, A. 2016. Timelines are Expressive Enough to Capture Action-based Temporal Planning. In *Proc. of the 23rd International Symposium on Temporal Representation and Reasoning*, 100–109.

Johnson, D. S. 1990. A Catalog of Complexity Classes. In *Handbook of Theoretical Computer Science, Volume A: Algorithms and Complexity*. 67–161.

Jónsson, A. K.; Morris, P. H.; Muscettola, N.; Rajan, K.; and Smith, B. D. 2000. Planning in Interplanetary Space: Theory and Practice. In *Proc. of 5th International Conference on Artificial Intelligence Planning and Scheduling*, 177–186.

Muscettola, N. 1994. HSTS: Integrating Planning and Scheduling. In Zweben, M., and Fox, M. S., eds., *Intelligent Scheduling*. Morgan Kaufmann. chapter 6, 169–212.

Rintanen, J. 2004. Complexity of Planning with Partial Observability. In *Proc. of the the 14th International Conference on Automated Planning and Scheduling*, 345–354.

Rintanen, J. 2007. Complexity of Concurrent Temporal Planning. In *Proc. of the 17th International Conference on Automated Planning and Scheduling*, 280–287.

Rintanen, J. 2012. Complexity of Conditional Planning under Partial Observability and Infinite Executions. In *Proc. of the 20th European Conference on Artificial Intelligence.*, 678–683.