Complexity of Qualitative Timeline-Based Planning

Dario Della Monica
University of Udine, Italy
https://users.dimi.uniud.it/~dario.dellamonica/
dario.dellamonica@uniud.it

Nicola Gigante
University of Udine, Italy
https://users.dimi.uniud.it/~nicola.gigante/
nicola.gigante@uniud.it

Salvatore La Torre
University of Salerno, Italy
https://docenti.unisa.it/salvatore.latorre
slatorre@unisa.it

Angelo Montanari
University of Udine, Italy
https://users.dimi.uniud.it/~angelo.montanari
angelo.montanari@uniud.it

Abstract

The timeline-based approach to automated planning was originally developed in the context of space missions. In this approach, problem domains are expressed as systems consisting of independent but interacting components whose behaviors over time, the *timelines*, are governed by a set of temporal constraints, called *synchronization rules*. Although timeline-based system descriptions have been successfully used in practice for decades, the research on the theoretical aspects only started recently. In the last few years, some interesting results have been shown concerning both its expressive power and the computational complexity of the related planning problem. In particular, the general problem has been proved to be EXPSPACE-complete. Given the applicability of the approach in many practical scenarios, it is thus natural to ask whether computationally simpler but still expressive fragments can be identified. In this paper, we study the timeline-based planning problem with the restriction that only *qualitative* synchronization rules, i.e., rules without explicit time bounds in the constraints, are allowed. We show that the problem becomes PSPACE-complete.

2012 ACM Subject Classification Computing methodologies → Temporal reasoning

Keywords and phrases Timeline-based planning, qualitative temporal constraints, complexity

Digital Object Identifier 10.4230/LIPIcs.TIME.2020.16

Funding The authors acknowledge the partial support by the Italian INdAM-GNCS project *Ragionamento Strategico e Sintesi Automatica di Sistemi Multi-Agente*. Nicola Gigante and Angelo Montanari are partially supported by the PRID project *ENCASE - Efforts in the uNderstanding of Complex interActing SystEms*.

1 Introduction

Timeline-based planning is an approach to automated planning and scheduling that arose in connection to space operations [16]. In this setting, planning domains are modeled as systems made of independent but interacting components, whose behavior over time, the *timelines*, is governed by a set of temporal constraints. Compared to other well-established *action-based* planning formalisms such as STRIPS [7] and PDDL [15], timelines conform to the *declarative* paradigm, and are very effective in modeling the behavior of complex systems where multiple components have to interact to obtain a common goal rather than scenarios where the target of the planning process is a single agent. Moreover, being born in a context

where scheduling of operations is often as important as planning, timeline-based languages are particularly tailored to *temporal reasoning*, and to the modeling of real-time systems. These features have proven to be quite useful in practice, as timeline-based planning systems have being successfully deployed both at NASA [2, 3] and ESA [8, 9] for short- to long-term mission planning over the last three decades [4, 6].

From a theoretical perspective, timeline-based modeling languages are interesting for their rich syntax and powerful semantics. However, the study of their expressiveness and computational complexity has started only recently. The expressive power of the language has been compared with action-based counterparts [11] and the general plan-existence problem for timeline-based planning has been proved to be EXPSPACE-complete [5, 12]. Timeline-based *games* [13], where some of the components are under control of the environment and the controller has to ensure the satisfaction of the constraints independently from the environment's choices, have also been studied to formalize and extend the practice of *flexible timelines* used in current timeline-based planning systems. Deciding whether there is a winning strategy for a timeline-based game has been proved to be 2EXPTIME-complete.

Despite the high computational complexity of the related decision problems, timeline-based models have been routinely and successfully used in complex real-world scenarios for decades. This apparent paradox indeed suggests the existence of interesting fragments of the general theory with more tractable complexities that are still suitable for meaningful real-world applications. Starting from this observation, this paper identifies the *qualitative* fragment of the timeline-based planning problems, *i.e.*, problems where the system behavior is described without referring to explicit time bounds. This restriction is nevertheless suitable for many practical scenarios as also testified by the plethora of approaches based on qualitative time models in many application domains including automated planning [10].

We prove that the plan-existence problem for this fragment is PSPACE-complete as opposed to the EXPSPACE-completeness of the general problem. The proof is given by means of an automata-theoretic construction that builds on the technique developed by Della Monica *et al.* [5] to prove the complexity in the general case. We also discuss some interesting consequences that this result brings to the table.

The paper is structured as follows. Section 2 recalls the basic syntax and semantics of timeline-based planning problems, and introduces the *qualitative* fragment studied here. Then, Section 3 proves that the problem can be solved in polynomial space, while Section 4 proves that it is also PSPACE-hard. Section 5 discusses the consequences of these results together with some interesting future developments.

2 Qualitative timeline-based planning

In this section, we introduce the notion of qualitative timeline-based planning. To this end, we recall the main definitions about timeline-based planning. We start with the definition of state variable, which is the basic building block of the framework.

▶ **Definition 1** (State variable). *A* state variable *is a tuple* $x = (V_x, T_x, D_x)$*, where:*

- V_x *is the* finite domain *of the variable;*
- $T_x : V_x \to 2^{V_x}$ *is the* value transition function*, which maps each value* $v \in V_x$ *to the set of values that can (immediately) follow it;*
- $D_x : V_x \to \mathbb{N}^+ \times (\mathbb{N}^+ \cup \{+\infty\})$ *is a function that maps each* $v \in V_x$ *to the pair* $(d_{min}^{x=v}, d_{max}^{x=v})$ *of minimum and maximum durations allowed for intervals where* $x = v$*.*

The behavior of a state variable x over time is modeled by a *timeline*, *i.e.*, a finite sequence of *tokens* each one denoting a value v and a time interval d meaning that x evaluates to v within d. Formally:

▶ **Definition 2** (Timelines). *A* token *for x is a tuple $\tau = (x, v, d)$, where x is a state variable, $v \in V_x$ is the value held by the variable, and $d \in \mathbb{N}^+$ is the* duration *of the token. A* timeline *for a state variable $x = (V_x, T_x, D_x)$ is a finite sequence $\mathsf{T} = \langle \tau_1, \ldots, \tau_k \rangle$ of tokens for x.*

For any token $\tau_i = (x, v_i, d_i)$ in a timeline $\mathsf{T} = \langle \tau_1, \ldots, \tau_k \rangle$ we can define the functions start-time$(\mathsf{T}, i) = \sum_{j=1}^{i-1} d_j$ and end-time$(\mathsf{T}, i) =$ start-time$(\mathsf{T}, i) + d_i$, hence mapping each token to the corresponding time interval [start-time, end-time) (right endpoint excluded). The *horizon* of a timeline T is defined as the end time end-time(T, k) of its last token τ_k. When there is no ambiguity, we write start-time(τ_i) and end-time(τ_i) to denote, respectively, start-time(T, i) and end-time(T, i).

The problem domain and the goal are modeled by a set of temporal constraints, called *synchronization rules*. A synchronization rule is of the form:

$$\textit{rule} := a_0[x_0 = v_0] \rightarrow \mathcal{E}_1 \vee \mathcal{E}_2 \vee \ldots \vee \mathcal{E}_k, \ \textit{with}$$
$$\mathcal{E}_i := \exists a_1[x_1 = v_1] a_2[x_2 = v_2] \ldots a_n[x_n = v_n] \,.\, \mathcal{C}_i$$

where $x_0, \ldots, x_n$ are state variables, $v_0, \ldots, v_n$ are values, with $v_i \in V_{x_i}$ for all i, $a_0, \ldots, a_n \in \mathcal{N}$ are symbols from a set of token names, and $\mathcal{C}$ is a conjunction of atomic clauses as described below. The semantics of synchronization rules is only informally recalled because of space concerns (see [13, Definitions 7 and 8] for a formal definition). Each rule consists of a *trigger* ($a[x_0 = v_0]$) and a disjunction of *existential statements*. It is satisfied if for each token satisfying the trigger, at least one of the disjuncts is satisfied. The trigger can also be empty ($\top$), in which case the rule is said to be *triggerless* and asks for the satisfaction of the body without any precondition. Each existential statement requires the existence of some tokens such that the clause $\mathcal{C}$ is satisfied. The clause is in turn a conjunction of *atoms*, that is, atomic relations between endpoints of the quantified tokens, of the form:

$$\textit{term} := \text{start}(a) \mid \text{end}(a) \mid t \qquad\qquad \textit{atom} := \textit{term} \leq_{[l,u]} \textit{term}$$

where $a \in \mathcal{N}$, $l \in \mathbb{N}$, $t \in \mathbb{N}$, and $u \in \mathbb{N} \cup \{+\infty\}$. As an example, the atom start$(a) \leq_{[l,u]}$ end(b) relates the two mentioned endpoints of the tokens a and b by stating that the distance between them must be at least l and at most u. When $u = +\infty$, there is no upper bound on the distance between endpoints. In this case, the atom is said to be *unbounded*, and *bounded* otherwise. An atom $\textit{term} \leq_{[l,u]} \textit{term}$ with $l = 0$ and $u = +\infty$ is said to be *qualitative*, and the subscript is usually omitted in this case. An endpoint of a token can also be related with an absolute time point (*e.g.* start$(a) \leq_{[0,3]} 4$). Such an atom is called a *time-point atom*.

A timeline-based planning problem consists of a set of state variables and a set of rules that represent the problem domain and the goal.

▶ **Definition 3** (Timeline-based planning problem). *An instance of a timeline-based planning problem, commonly referred to as a* timeline-based planning problem*, is a pair $P = (\mathsf{SV}, S)$, where SV is a set of state variables and S is a set of synchronization rules over SV.*

A solution plan for a given timeline-based planning problem is a set of timelines, one for each state variable.

▶ **Definition 4** (Solution plan). *A* solution plan *for a problem $P = (\mathsf{SV}, S)$ is a set of timelines $\Gamma = \{\mathsf{T}_x \mid x \in \mathsf{SV}\}$, one for each $x \in \mathsf{SV}$, all with the same* horizon*, such that $v_{i+1} \in T_x(v_i)$ and $d_{min}^{x=v_i} \leq d_i \leq d_{max}^{x=v_i}$ for all tokens $\tau_i = (x, v_i, d_i) \in \mathsf{T}_x$, and all the rules in S are satisfied.*

We know from [12] that the problem of deciding whether there exists a solution plan for a given timeline-based planning problem is EXPSPACE-complete. In this paper, we focus on the *qualitative* version of timeline-based planning problems.

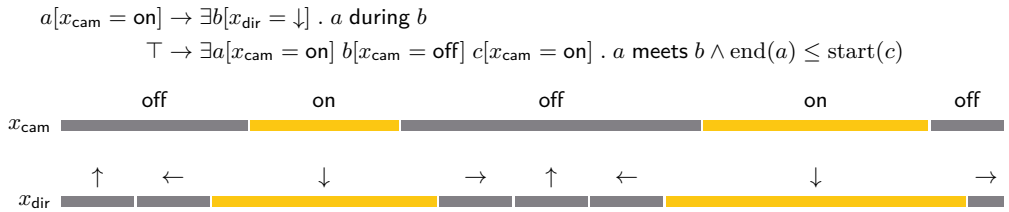

Figure 1 An example of timeline-based planning problem. Two state variables are used to represent the on/off state of a camera (x_{cam}) and its pointing direction (x_{dir}). The transition function of x_{dir} forces the camera to only move counterclockwise.

▶ **Definition 5** (Qualitative timeline-based planning problem)**.** *A timeline-based planning problem* $P = (\mathsf{SV}, S)$ *is* qualitative *if* $D_x(v) = (1, +\infty)$ *for each* $x \in \mathsf{SV}$ *and* $v \in V_x$*, and all the synchronization rules in* S *make only use of* qualitative *atoms, and no time-point atoms.*

Qualitative synchronization rules do not allow one to express *real-time* constraints, but they are still a quite expressive formalism. Equalities between endpoints and between whole tokens are definable, *i.e.*, $\mathrm{start}(a) = \mathrm{start}(b)$ can be written as $\mathrm{start}(a) \leq \mathrm{start}(b) \wedge \mathrm{start}(b) \leq \mathrm{start}(a)$ and $a = b$ as $\mathrm{start}(a) = \mathrm{start}(b) \wedge \mathrm{end}(a) = \mathrm{end}(b)$. Moreover, one can express non-strict versions of all Allen's interval relations [1]: one can define a meets b as $\mathrm{end}(a) = \mathrm{start}(b)$, a during b as $\mathrm{start}(a) \leq \mathrm{start}(b) \wedge \mathrm{end}(b) \leq \mathrm{end}(a)$, and so on.

Figure 1 shows a possible solution for a problem with two state variables, x_{cam} and x_{dir}, with $V_{x_{\mathsf{cam}}} = \{\mathsf{on}, \mathsf{off}\}$ and $V_{x_{\mathsf{dir}}} = \{\uparrow, \leftarrow, \downarrow, \rightarrow\}$, that respectively represent the on/off state of a camera and its pointing direction. The transition function $T_{x_{\mathsf{dir}}}$ of x_{dir} is such that the camera can only stay still or move counterclockwise, *e.g.* $T_{x_{\mathsf{dir}}}(\leftarrow) = \{\leftarrow, \downarrow\}$. The rules are built on the set $\mathcal{N} = \{a, b, \ldots\}$ of token names. The first rule requires the camera to point down every time it is switched on (*e.g.*, to point towards ground from an airplane). The objective of the system is to perform two shoots, provided that the camera is switched off between them in order to cool down. This goal is encoded by the second *triggerless* synchronization rule.

3 Complexity of qualitative timeline-based planning

In this section, we give a *polynomial-space* algorithm to decide whether there exists a solution plan for a given qualitative timeline-based planning problem.

Given a qualitative timeline-based planning problem P, we show how to build a *non-deterministic finite automaton* (NFA) $\mathcal{A}$ whose accepted words correspond to the solution plans of P. The approach, inspired by the one adopted by Della Monica *et al.* [5] for the general case, has been not only tailored to the qualitative setting but also refined to obtain the desired complexity result.

3.1 Plans as words

Let $P = (\mathsf{SV}, S)$ be a qualitative timeline-based planning problem and let $V = \bigcup_{x \in \mathsf{SV}} V_x$. Without loss of generality, we consider only qualitative timeline-based planning problems whose state variables have *trivial* transition functions, *i.e.*, for each $x \in \mathsf{SV}$ and $v \in V_x$, either $T_x(v) = V_x$ or $T_x(v) = \varnothing$. The first step is to encode timelines and plans as words that can be fed to an automaton. Let the alphabet associated with P be $\Sigma_P = \{\sigma : \mathsf{SV} \to V \cup \{\circlearrowleft\} \mid \sigma(x) \in V_x \cup \{\circlearrowleft\}\}$, *i.e.*, symbols map each state variable x to a value within its domain V_x

or to a special symbol $\circlearrowleft$. By fixing an ordering among the variables, we will also denote such maps as tuples in the standard way. Observe that the size of the alphabet is at most exponential in the size of P, precisely $|\Sigma_P| \leq (|V|+1)^{|\mathsf{SV}|}$.

Let the set of *initial symbols* be defined as $\Sigma_P^I = \{\sigma \in \Sigma_P \mid \forall x \in \mathsf{SV} \,.\, \sigma(x) \neq \circlearrowleft\}$. Each plan can be encoded with a word in $\Sigma_P^I \Sigma_P^*$, and *vice versa*, where each occurrence of $v \in V_x$ denotes a time point where x is updated to v and each occurrence of $\circlearrowleft$ a time point where the value of x stays unchanged. Formally, let $\overline{\sigma} = \langle \sigma_0, \ldots, \sigma_{|\overline{\sigma}|-1} \rangle$ be a word in $\Sigma_P^I \Sigma_P^*$. Given a state variable $x \in \mathsf{SV}$, let $\{i_0, i_1, \ldots, i_{k-1}\} = \{i \mid \sigma_i(x) \neq \circlearrowleft\}$, with $i_{j-1} < i_j$ for all $j \in \{1, \ldots, k-1\}$, *i.e.*, the ordered set of positions where the value of x changes. Then, the word $\overline{\sigma}$ induces a *timeline* T_x defined as $\mathsf{T}_x = \langle (x, v_{i_0}, i_1 - i_0), (x, v_{i_1}, i_2 - i_1), \ldots, (x, v_{i_{k-1}}, i_k - i_{k-1}) \rangle$, where $v_i = \sigma_i(x)$ and $i_k = |\overline{\sigma}|$, as the timeline for x induced by $\overline{\sigma}$, while the plan induced by $\overline{\sigma}$ is the set of all timelines, one for each $x \in \mathsf{SV}$, induced by $\overline{\sigma}$. A converse correspondence of plans to words can be defined accordingly.

3.2 Blueprints and viewpoints

A key concept in our construction is the *blueprint*: a description of a possible way to satisfy a synchronization rule. Here, we use a significantly revised notion of blueprint with the respect to the one introduced in [5]. This new notion allows us to gain in succinctness and plays a crucial role in achieving the wished complexity bound.

We begin with some notation and terminology. Let $\mathcal{P}$ be a *preorder* over its domain $dom(\mathcal{P})$. For every $x \in dom(\mathcal{P})$, we define $pred(\mathcal{P}, x)$ as the set of immediate predecessors of x in $\mathcal{P}$ and $succ(\mathcal{P}, x)$ as the set of immediate successors of x in $\mathcal{P}$. We define $maximals(\mathcal{P})$ as the set of elements of $dom(\mathcal{P})$ with no successors in $\mathcal{P}$ and $minimals(\mathcal{P})$ as the set of elements of $dom(\mathcal{P})$ with no predecessors in $\mathcal{P}$. Finally, given $K \subseteq dom(\mathcal{P})$, we say that K is $\mathcal{P}$-complete if for every $x \in dom(\mathcal{P})$ there is $y \in K$ such that $x \preceq_{\mathcal{P}} y$ or $y \preceq_{\mathcal{P}} x$ and that K is minimally $\mathcal{P}$-complete if it is $\mathcal{P}$-complete and its elements are pairwise incomparable in $\mathcal{P}$.

We assume, *w.l.o.g.*, that at least one rule with $a[x = v]$ as trigger belongs to S for each $x \in \mathsf{SV}$ and $v \in V_x$. Now, fix a synchronization rule $\mathcal{R} \equiv a_0[x_0 = v_0] \rightarrow \mathcal{E}_1 \vee \ldots \vee \mathcal{E}_m$, and one of its existential statements $\mathcal{E} \equiv \exists a_1[x_1 = v_1] \cdots a_n[x_n = v_n] \,.\, \mathcal{C}$, *i.e.*, $\mathcal{E} = \mathcal{E}_j$ for some $j \in \{1, \ldots, m\}$. We assume, *w.l.o.g.*, that the atom $\mathrm{start}(a_i) \leq \mathrm{end}(a_i)$ occurs in $\mathcal{C}$ for all $i \in \{0, \ldots, n\}$; we also assume that, for every i, j with $x_i = x_j$ and $i \neq j$, if one among $\mathrm{start}(a_i) \leq \mathrm{start}(a_j)$, $\mathrm{end}(a_i) \leq \mathrm{end}(a_j)$, and $\mathrm{start}(a_i) \leq \mathrm{end}(a_j)$ occurs in $\mathcal{C}$, then $\mathrm{end}(a_i) \leq \mathrm{start}(a_j)$ occurs in $\mathcal{C}$ as well. Recall that $v_i \in V_{x_i}$ for all $i \in \{0, \ldots, n\}$. Then, $\mathcal{E}$ defines a preorder $\mathcal{P}_{\mathcal{E}}$ over the domain $dom(\mathcal{P}_{\mathcal{E}}) = \{\mathrm{start}(a_i), \mathrm{end}(a_i) \mid i \in \{0, \ldots, n\}\}$, *i.e.*, over the set of endpoints of all the tokens quantified by $\mathcal{E}$. Intuitively, a *blueprint* for $\mathcal{E}$, denoted by $B_{\mathcal{E}}$, is an extension of the preorder defined by $\mathcal{E}$ where an additional element is inserted between any pair of comparable elements, and before and after minimal and maximal elements, respectively. Such additional elements, denoted $pumps(B_{\mathcal{E}})$, are called the *pumping points* of $B_{\mathcal{E}}$. Formally, blueprints are defined as follows.

▶ **Definition 6** (Blueprint). *Let $\mathcal{E}$ be an existential statement. A* blueprint *for $\mathcal{E}$ is a* preorder *$B_{\mathcal{E}}$ defined as:*

1. *$dom(B_{\mathcal{E}}) = dom(\mathcal{P}_{\mathcal{E}}) \cup pumps(B_{\mathcal{E}})$ where*

$$pumps(B_{\mathcal{E}}) = \{\langle y_{|} x \rangle \mid x \in dom(\mathcal{P}_{\mathcal{E}}) \setminus minimals(\mathcal{P}_{\mathcal{E}}), y \in pred(\mathcal{P}_{\mathcal{E}}, x)\} \cup \{\langle -_{|} x \rangle \mid x \in minimals(\mathcal{P}_{\mathcal{E}})\} \cup \{\langle x_{|} - \rangle \mid x \in maximals(\mathcal{P}_{\mathcal{E}})\};$$

2. *for all $x, y \in dom(\mathcal{P}_{\mathcal{E}})$, $x \preceq_{B_{\mathcal{E}}} y$ if and only if $x \preceq_{\mathcal{P}_{\mathcal{E}}} y$,* i.e., *the ordering relation is unchanged over elements of $dom(\mathcal{P}_{\mathcal{E}})$;*

3. *for every* $x \in dom(\mathcal{P}_\mathcal{E}) \setminus minimals(\mathcal{P}_\mathcal{E})$ *and* $y \in pred(\mathcal{P}_\mathcal{E}, x)$*, we have* $y \preceq_{B_\mathcal{E}} \langle y|x\rangle$ *and* $\langle y|x\rangle \preceq_{B_\mathcal{E}} x$*;*
4. *for every* $x \in minimals(\mathcal{P}_\mathcal{E})$*, we have* $\langle -|x\rangle \preceq_{B_\mathcal{E}} x$*; and*
5. *for every* $x \in maximals(\mathcal{P}_\mathcal{E})$*, we have* $x \preceq_{B_\mathcal{E}} \langle x|-\rangle$*.*

Blueprints statically describe the syntactic structure of the rules. A *viewpoint* pairs a blueprint with a set of pointers to its pumping points. It is the dynamic structure that keeps track of how the rules are matching on the specific word being read.

▶ **Definition 7** (Viewpoint)**.** *A* viewpoint *is a pair* $\mathbb{V} = \langle B_\mathcal{E}, K\rangle$*, where* $B_\mathcal{E}$ *is a blueprint for an existential statement* $\mathcal{E}$ *and* $K \subseteq pumps(B_\mathcal{E})$ *is a subset of its pumping points that is* minimally $B_\mathcal{E}$-complete*.*

Given $\mathbb{V} = \langle B_\mathcal{E}, K\rangle$, K is the *frontier* of $\mathbb{V}$, and its elements are its *frontier points*. Moreover, $\mathbb{V}$ is said to be *minimal* or *maximal* if $K = minimals(B_\mathcal{E})$ or $K = maximals(B_\mathcal{E})$, respectively.

A viewpoint keeps track of how a blueprint is being matched over a plan encoding; in particular, its frontier separates the already matched part from the rest of the blueprint that still needs to be matched. When a symbol is read, each frontier point can either *pump* (*i.e.*, stay unchanged) or *step* (*i.e.*, advance to point to other pumping points). Formally, a viewpoint $\mathbb{V} = \langle B_\mathcal{E}, K\rangle$ can *evolve* into another viewpoint $\mathbb{V}' = \langle B_\mathcal{E}, K'\rangle$, written $\mathbb{V} \to \mathbb{V}'$, if and only if for all $k \in K$ either $k \in K'$ (*i.e.*, k pumps) or $k' \notin K'$ for all k' such that $k' \preceq_{B_\mathcal{E}} k$ (*i.e.*, k steps). If $\mathbb{V} = \langle B_\mathcal{E}, K\rangle$ evolves into $\mathbb{V}' = \langle B_\mathcal{E}, K'\rangle$, the points of $\mathcal{P}_\mathcal{E}$ that move into the matched part define the set of points that are *consumed* over $\mathbb{V} \to \mathbb{V}'$, namely:

$$consumed(\mathbb{V}, \mathbb{V}') = \{x \in dom(\mathcal{P}_\mathcal{E}) \mid y \preceq_{B_\mathcal{E}} x \preceq_{B_\mathcal{E}} y' \text{ for some } y \in K, y' \in K'\}.$$

We say that a state variable $x \in \mathsf{SV}$ is *open* in $\mathbb{V}$ if there is some $i \in \{0, \dots, n\}$ and some $k \in K$ such that $x_i = x$ and $\mathrm{start}(a_i) \preceq_{B_\mathcal{E}} k \preceq_{B_\mathcal{E}} \mathrm{end}(a_i)$, *i.e.*, the frontier says that the start of a token for x has matched but its end has not.

Depending on which symbol is currently being read, only some of the possible evolutions of a viewpoint are admissible.

▶ **Definition 8** (Evolution of a viewpoint)**.** *Let* $\mathbb{V} = \langle B_\mathcal{E}, K\rangle$ *and* $\mathbb{V}' = \langle B_\mathcal{E}, K'\rangle$ *be two viewpoints over the blueprint* $B_\mathcal{E}$ *of some existential statement* $\mathcal{E}$ *and let* $\sigma \in \Sigma_P$*. We say that* $\mathbb{V}$ can evolve into $\mathbb{V}'$ when reading σ*, written* $\mathbb{V} \xrightarrow{\sigma} \mathbb{V}'$*, if the following conditions hold:*

1. *if* $\sigma(x) \neq \circlearrowleft$ *and* x *is open in* $\mathbb{V}$*, then* $\mathrm{end}(a_i) \in consumed(\mathbb{V}, \mathbb{V}')$ *for some* i *with* $x_i = x$*,*
2. *if* $T_{x_i}(v_i) = \varnothing$*, then* $\mathrm{end}(a_i) \notin consumed(\mathbb{V}, \mathbb{V}')$*,*
3. $consumed(\mathbb{V}, \mathbb{V}')$ *is compatible with* σ*, that is:*
   a. $\sigma(x_i) = v_i$ *for every* $\mathrm{start}(a_i) \in consumed(\mathbb{V}, \mathbb{V}')$*,*
   b. $\sigma(x_i) \neq \circlearrowleft$ *for every* $\mathrm{end}(a_i) \in consumed(\mathbb{V}, \mathbb{V}')$*, and*
   c. *if* $\mathrm{start}(a_i) \in consumed(\mathbb{V}, \mathbb{V}')$*, then* $\mathrm{end}(a_i) \notin consumed(\mathbb{V}, \mathbb{V}')$*.*

3.3 Automaton construction

The above notions allow us to build the automaton $\mathcal{A}_P$ for a given qualitative timeline-based planning problem $P = (\mathsf{SV}, S)$. The states of the automaton consist of sets of viewpoints over the rules of P. However, in order to keep the size of the states small, some combinations of viewpoints are excluded *a priori* by forcing a total order on the viewpoints of a state that are built on the same blueprint.

Formally, let $\mathbb{V} = \langle B_\mathcal{E}, K\rangle$ and $\mathbb{V}' = \langle B_\mathcal{E}, K'\rangle$ be two viewpoints over the same blueprint. We define $\mathbb{V} \preceq \mathbb{V}'$ if and only if for every $k \in K$ there is a $k' \in K'$ such that $k \preceq_{B_\mathcal{E}} k'$. Let $\mathbb{V}^{max}_\mathcal{E} = \langle B_\mathcal{E}, maximals(B_\mathcal{E})\rangle$. Then, $\mathbb{V}$ is said to be *final* for $B_{\mathcal{E}_i}$ if $consumed(\mathbb{V}, \mathbb{V}^{max}_\mathcal{E})$ contains only elements of the form $\mathrm{end}(a)$ for some $a \in \mathcal{N}$.

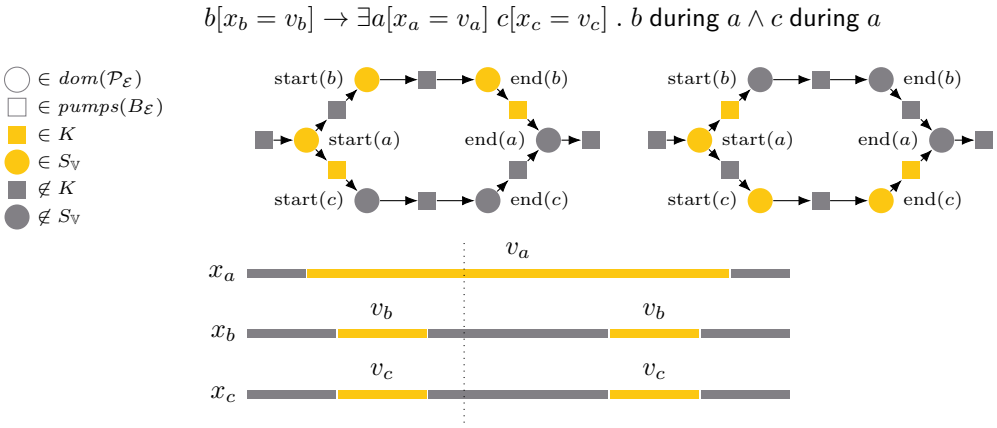

Figure 2 Two incomparable viewpoints for the same rule (on the top) and a plan where the rule is triggered twice. The current time-point in the plan is represented by the dotted line. The left (resp., right) viewpoint takes care of the satisfaction of the rule when triggered by the first (resp., second) occurrence of v_b. The incomparability is due to the left viewpoint trying to satisfy a rule triggered by an earlier occurrence of v_b by using a latter occurrence of v_c and, vice versa, the right viewpoint trying to satisfy a rule triggered by a latter occurrence of v_b by using an earlier occurrence of v_c. We will show that these situations can be avoided.

Now, let Υ_P be the set of all the viewpoints of the existential statements of the rules of P. The automaton $\mathcal{A}_P$ is defined as follows.

▶ **Definition 9** (Automaton construction). *Let $P = (\mathsf{SV}, S)$ be a qualitative timeline-based planning problem. The NFA $\mathcal{A}_P = (Q_P, \Sigma_P, q_P^I, Q_P^F, \Delta_P)$ associated with P is such that:*

1. *the set of states consists of the initial state $q_P^I \not\subseteq \Upsilon_P$ and a selection of the subsets of Υ_P:*

$$Q_P = \{q_P^I\} \cup \{\Upsilon \subseteq \Upsilon_P \mid\ \mathbb{V} \preceq \mathbb{V}' \text{ or } \mathbb{V}' \preceq \mathbb{V} \text{ for all } \mathbb{V} = \langle B_\mathcal{E}, K\rangle, \mathbb{V}' = \langle B_\mathcal{E}, K'\rangle \in \Upsilon\};$$

2. *the* final *states Q_P^F are defined as follows:*
   a. *$\Upsilon \in Q_P^F$ if and only if Υ is made of* final *viewpoints and for every* triggerless *rule $\top \rightarrow \mathcal{E}_1 \vee \ldots \vee \mathcal{E}_k$ in S, there is $i \in \{1, \ldots, k\}$ such that Υ contains a* final *viewpoint for $B_{\mathcal{E}_i}$;*
   b. *if there are no triggerless rules, then $q_P^I \in Q_P^F$;*
3. *for all $\Upsilon, \Upsilon' \subseteq Q_P \setminus \{q_P^I\}$ and $\sigma \in \Sigma_P$, $(\Upsilon, \sigma, \Upsilon') \in \Delta_P$ iff:*
   a. *for every $\mathbb{V} \in \Upsilon$, there is a $\mathbb{V}' \in \Upsilon'$ such that $\mathbb{V} \xrightarrow{\sigma} \mathbb{V}'$;*
   b. *for every $\mathbb{V}' \in \Upsilon'$, there is a $\mathbb{V} \in \Upsilon$ such that $\mathbb{V} \xrightarrow{\sigma} \mathbb{V}'$; and*
   c. *if there is a synchronization rule $a_0[x_0 = v_0] \rightarrow \mathcal{E}_1 \vee \mathcal{E}_2 \vee \ldots \vee \mathcal{E}_k$ in S and $\sigma(x_0) = v_0$, then there are $\mathbb{V} \in \Upsilon$ and $\mathbb{V}' \in \Upsilon'$ such that:*
      - *$\mathbb{V} = \langle B_{\mathcal{E}_i}, K\rangle$ for some $i \in \{1, \ldots, k\}$;*
      - *$\mathbb{V} \xrightarrow{\sigma} \mathbb{V}'$;*
      - *$\mathrm{start}(a_0) \in consumed(\mathbb{V}, \mathbb{V}')$;*
4. *for all $\Upsilon' \subseteq Q_P \setminus \{q_P^I\}$ and $\sigma \in \Sigma_P$, $(q_P^I, \sigma, \Upsilon') \in \Delta_P$ iff $\sigma \in \Sigma_P^I$ and $(\Upsilon^I, \sigma, \Upsilon') \in \Delta_P$ for some set Υ^I of minimal viewpoints.*

Note that if any possible set of viewpoints were a valid state, the size of the automaton would be doubly exponential. Instead, the *symmetry-breaking condition* imposed by Item 1 of Definition 9, *i.e.*, for every $\Upsilon \in Q_P$ and every $\mathbb{V} = \langle B_\mathcal{E}, K\rangle, \mathbb{V} = \langle B_\mathcal{E}, K'\rangle \in \Upsilon$, we have $\mathbb{V} \preceq \mathbb{V}'$ or $\mathbb{V}' \preceq \mathbb{V}$, allows us to shrink the size of $\mathcal{A}_P$ to be only exponential in the size of P. Figure 2 shows an example of incomparable viewpoints. We will show that these situations can be avoided, thus obtaining an automaton of at most *exponential* size.

▶ **Lemma 10** (Automaton size). *Let P be a qualitative timeline-based planning problem. The size of its associated automaton $\mathcal{A}_P$ is at most* exponential *in the size of P.*

Proof. Let $P = (\mathsf{SV}, S)$ be a qualitative timeline-based planning problem and consider the set Q_P of states of its associated automaton $\mathcal{A}_P$ as defined in Definition 9. For each viewpoint $\mathbb{V} = \langle B_\mathcal{E}, K\rangle$, let $S_\mathbb{V} = \{x \in \mathit{dom}(\mathcal{P}_\mathcal{E}) \mid \exists k \in K \,.\, x \preceq_{B_\mathcal{E}} k\}$ be the set of elements *covered* by $\mathbb{V}$, *i.e.*, those elements of the blueprint $B_\mathcal{E}$ that have already been matched over the word, as witnessed by $\mathbb{V}$. Notice that $\mathbb{V} \preceq \mathbb{V}'$ implies $S_\mathbb{V} \subseteq S_{\mathbb{V}'}$. Moreover, the sets of covered elements for all the viewpoints $\mathbb{V}$ for a blueprint $B_\mathcal{E}$ form a lattice with regard to set inclusion, where $\bot = \varnothing$ (which is the set of elements covered by the minimal viewpoint $\mathbb{V}_\bot = \langle B_\mathcal{E}, \mathit{minimals}(B_\mathcal{E})\rangle$), and $\top = \mathit{dom}(\mathcal{P}_\mathcal{E})$ (which is the set of elements covered by the maximal viewpoint $\mathbb{V}_\top = \langle B_\mathcal{E}, \mathit{maximals}(B_\mathcal{E})\rangle$). According to the definition of the automaton states (Item 1 of Definition 9), the viewpoints for a blueprint $B_\mathcal{E}$ included in any state $\Upsilon \in Q_P$ form a total order. Since the number of disjuncts occurring in P as well as the distance from $\bot$ to $\top$ in the aforementioned lattice is polynomial in the size of P (in fact, it is linear), the size of Υ is polynomial, and the size of Q_P is thus at most *exponential* in the size of P. ◀

3.4 Soundness and completeness

Here we prove that the automaton construction of Definition 9 correctly captures qualitative timeline-based planning problems.

Let $P = (\mathsf{SV}, S)$ be a qualitative timeline-based planning problem that admits a solution plan $\Gamma = \{\mathsf{T}_x \mid x \in \mathsf{SV}\}$. For each $\mathcal{R} \in S$, if $\mathcal{R}$ is not triggerless, then we denote by $\mathcal{T}_\Gamma^\mathcal{R}$ the set of tokens in Γ triggering $\mathcal{R}$; if $\mathcal{R}$ is triggerless, we let $\mathcal{T}_\Gamma^\mathcal{R} = \{\top_\mathcal{R}\}$ (this notation is useful to handle triggerless rules uniformly). Moreover, we denote $\mathcal{T}_\Gamma = \bigcup_{\mathcal{R}\in S} \mathcal{T}_\Gamma^\mathcal{R}$ the set of all the triggers occurring in Γ, plus one fictitious token $\top_\mathcal{R}$ for each triggerless rule $\mathcal{R} \in S$, which is said to be triggered by $\top_\mathcal{R}$. Now, let $\mathcal{R} \equiv a_0[x_0 = v_0] \to \mathcal{E}_1 \vee \mathcal{E}_2 \vee \ldots \vee \mathcal{E}_k$ be a rule in S. Since Γ is a solution plan, for each token $\tau \in \mathcal{T}_\Gamma^\mathcal{R}$, we can identify a disjunct $\mathcal{R}(\tau) \in \{\mathcal{E}_1, \ldots, \mathcal{E}_k\}$ such that $\mathcal{R}(\tau)$ is satisfied for τ in Γ.

In order to link the words accepted by an automaton to the solution plans for P, we need to specify how *blueprints* are connected to timelines and plans.

▶ **Definition 11** (Blueprint instantiation). *Let $P = (\mathsf{SV}, S)$ be a qualitative timeline-based planning problem and let $\Gamma = \{\mathsf{T}_x \mid x \in \mathsf{SV}\}$ be a solution plan for P.*

For every $\mathcal{R} \in S$ and $\tau \in \mathcal{T}_\Gamma^\mathcal{R}$, a blueprint instantiation *for τ in $\mathcal{R}$ is a labeling function $\mathcal{L}_\tau^{\mathcal{R}(\tau)} : \mathit{dom}(\mathcal{P}_{\mathcal{R}(\tau)}) \to \mathbb{N}$ that maps every element x in the domain of the preorder $\mathcal{P}_{\mathcal{R}(\tau)}$ to a time point $\mathcal{L}_\tau^{\mathcal{R}(\tau)}(x)$ such that:*

1. *$x \preceq_{\mathcal{P}_{\mathcal{R}(\tau)}} y$ implies $\mathcal{L}_\tau^{\mathcal{R}(\tau)}(x) \leq \mathcal{L}_\tau^{\mathcal{R}(\tau)}(y)$ for every $x, y \in \mathit{dom}(\mathcal{P}_{\mathcal{R}(\tau)})$;*
2. *for every $\mathrm{start}(a_i), \mathrm{end}(a_i) \in \mathit{dom}(\mathcal{P}_{\mathcal{R}(\tau)})$, there is a (unique) token $\tau' = (x_i, v_i, d) \in \mathsf{T}_{x_i}$ such that $\text{start-time}(\tau') = \mathcal{L}_\tau^{\mathcal{R}(\tau)}(\mathrm{start}(a_i))$ and $\text{end-time}(\tau') = \mathcal{L}_\tau^{\mathcal{R}(\tau)}(\mathrm{end}(a_i))$.*

When clear from the context, we omit the superscript when referring to blueprint instantiations. Intuitively, $\mathcal{L}_\tau$ is a witness of the satisfaction of $\mathcal{R}$ when triggered by some $\tau \in \mathcal{T}_\Gamma^\mathcal{R}$. We define a partial order over blueprint instantiations such that $\mathcal{L}_1 \leq \mathcal{L}_2$ if and only if $\mathit{dom}(\mathcal{L}_1) = \mathit{dom}(\mathcal{L}_2)$ and $\mathcal{L}_1(x) \leq \mathcal{L}_2(x)$ for each $x \in \mathit{dom}(\mathcal{L}_1)$. The following statement is fundamental in proving that the symmetry-breaking condition given for the automaton states in Definition 9 does not affect the completeness of the construction.

▶ **Lemma 12.** *Let $P = (\mathsf{SV}, S)$ be a timeline-based planning problem that admits a solution plan Γ. Moreover, given $\mathcal{R} \in S$, let $\tau_1, \tau_2 \in \mathcal{T}_\Gamma^\mathcal{R}$ be two tokens that trigger $\mathcal{R}$, such that $\text{end-time}(\tau_1) \leq \text{start-time}(\tau_2)$ and $\mathcal{R}(\tau_1) = \mathcal{R}(\tau_2)$.*

There exist two blueprint instantiations, $\mathcal{L}_{\tau_1}$ for τ_1 and $\mathcal{L}_{\tau_2}$ for τ_2, such that $\mathcal{L}_{\tau_1} \leq \mathcal{L}_{\tau_2}$.

Proof. Since Γ is a solution plan for P, there are two blueprint instantiations, $\mathcal{L}_{\tau_1}$ for τ_1 and $\mathcal{L}_{\tau_2}$ for τ_2. If $\mathcal{L}_{\tau_1} \not\leq \mathcal{L}_{\tau_2}$, then we define blueprint instantiation $\mathcal{L}'_{\tau_1}$ for τ_1 such that $\mathcal{L}'_{\tau_1} \leq \mathcal{L}_{\tau_2}$. Note that, since $\mathcal{R}(\tau_1) = \mathcal{R}(\tau_2)$, $dom(\mathcal{L}_{\tau_1}) = dom(\mathcal{L}_{\tau_2})$. We define $\mathcal{L}'_{\tau_1} : dom(\mathcal{L}_{\tau_1}) \to \mathbb{N}$ as follows. For every $x \in dom(\mathcal{L}_{\tau_1})$:

$$\mathcal{L}'_{\tau_1}(x) = \begin{cases} \mathcal{L}_{\tau_1}(x) & \text{if } \mathcal{L}_{\tau_1}(x) \leq \mathcal{L}_{\tau_2}(x) \\ \mathcal{L}_{\tau_2}(x) & \text{if } \mathcal{L}_{\tau_2}(x) < \mathcal{L}_{\tau_1}(x) \end{cases}$$

It is clear that $\mathcal{L}'_{\tau_1} \leq \mathcal{L}_{\tau_2}$. We have to show that $\mathcal{L}'_{\tau_1}$ is a blueprint instantiation for τ_1. To see that Item 1 of Definition 11 holds, let $x \preceq_{\mathcal{P}_{\mathcal{R}(\tau_1)}} y$. We distinguish three cases:

1. if both $\mathcal{L}'_{\tau_1}(x) = \mathcal{L}_{\tau_1}(x)$ and $\mathcal{L}'_{\tau_1}(y) = \mathcal{L}_{\tau_1}(y)$ or both $\mathcal{L}'_{\tau_1}(x) = \mathcal{L}_{\tau_2}(x)$ and $\mathcal{L}'_{\tau_1}(y) = \mathcal{L}_{\tau_2}(y)$, then $\mathcal{L}'_{\tau_1}(x) \leq \mathcal{L}'_{\tau_1}(y)$ holds due to $\mathcal{L}_{\tau_1}$ and $\mathcal{L}_{\tau_2}$ being blueprint instantiations;
2. if $\mathcal{L}'_{\tau_1}(x) = \mathcal{L}_{\tau_1}(x)$ and $\mathcal{L}'_{\tau_1}(y) = \mathcal{L}_{\tau_2}(y)$, then $\mathcal{L}_{\tau_1}(x) \leq \mathcal{L}_{\tau_2}(x)$ holds by definition. By $x \preceq_{\mathcal{P}_{\mathcal{R}(\tau)}} y$, we know that $\mathcal{L}_{\tau_2}(x) \leq \mathcal{L}_{\tau_2}(y)$, hence $\mathcal{L}_{\tau_1}(x) \leq \mathcal{L}_{\tau_2}(y)$, thus $\mathcal{L}'_{\tau_1}(x) \leq \mathcal{L}'_{\tau_1}(y)$;
3. if $\mathcal{L}'_{\tau_1}(x) = \mathcal{L}_{\tau_2}(x)$ and $\mathcal{L}'_{\tau_1}(y) = \mathcal{L}_{\tau_1}(y)$, then $\mathcal{L}_{\tau_2}(x) < \mathcal{L}_{\tau_1}(x)$ holds by definition. By $x \preceq_{\mathcal{P}_{\mathcal{R}(\tau)}} y$, we know that $\mathcal{L}_{\tau_1}(x) \leq \mathcal{L}_{\tau_1}(y)$, hence $\mathcal{L}_{\tau_2}(x) < \mathcal{L}_{\tau_1}(y)$, thus $\mathcal{L}'_{\tau_1}(x) \leq \mathcal{L}'_{\tau_1}(y)$.

To see that Item 2 holds, consider $\text{start}(a_i), \text{end}(a_i) \in dom(\mathcal{P}_{\mathcal{R}(\tau_1)})$. Note that it cannot be the case that $\mathcal{L}'_{\tau_1}(\text{start}(a_i)) = \mathcal{L}_{\tau_1}(\text{start}(a_i))$ but $\mathcal{L}'_{\tau_1}(\text{end}(a_i)) = \mathcal{L}_{\tau_2}(\text{end}(a_i))$ (or *vice versa*), because that would imply that $\mathcal{L}_{\tau_1}(\text{start}(a_i)) \leq \mathcal{L}_{\tau_2}(\text{start}(a_i)) \leq \mathcal{L}_{\tau_2}(\text{end}(a_i)) < \mathcal{L}_{\tau_1}(\text{end}(a_i))$, which is impossible because, by hypothesis, $\mathcal{L}_{\tau_1}$ maps $\text{start}(a_i)$ and $\text{end}(a_i)$ into the endpoints of a single token τ, that cannot contain another token for the same variable. Thus, we have either $\mathcal{L}'_{\tau_1}(\text{start}(a_i)) = \mathcal{L}_{\tau_1}(\text{start}(a_i))$ and $\mathcal{L}'_{\tau_1}(\text{end}(a_i)) = \mathcal{L}_{\tau_1}(\text{end}(a_i))$ or $\mathcal{L}'_{\tau_1}(\text{start}(a_i)) = \mathcal{L}_{\tau_2}(\text{start}(a_i))$ and $\mathcal{L}'_{\tau_1}(\text{end}(a_i)) = \mathcal{L}_{\tau_2}(\text{end}(a_i))$, and the thesis follows since $\mathcal{L}_{\tau_1}$ and $\mathcal{L}_{\tau_2}$ are blueprint instantiations themselves. ◀

We can now prove the direct correspondence between solution plans and accepted words.

▶ **Lemma 13.** *Given a qualitative timeline-based planning problem P and its associated automaton $\mathcal{A}_P$, there is a solution plan for P if and only if $\mathcal{L}(\mathcal{A}_P) \neq \varnothing$.*

Proof. Let $P = (\mathsf{SV}, S)$ be a qualitative timeline-based planning problem, and let $\mathcal{A}_P = (Q_P, \Sigma_P, q_P^I, q_P^F, \Delta_P)$ be the automaton associated with P by Definition 9. It is easy to see that, given a word $\overline{\sigma}$ accepted by $\mathcal{A}_P$, the plan encoded by $\overline{\sigma}$ is a solution plan for P. We thus focus only on the proof of the other direction.

Fix a solution plan $\Gamma = \{\mathsf{T}_x \mid x \in \mathsf{SV}\}$ for P, and denote $\overline{\sigma} = \langle \sigma_0, \ldots, \sigma_m \rangle$ the corresponding word. We wish to prove that $\overline{\sigma}$ is accepted by $\mathcal{A}_P$. This direction of the proof needs some care because of the symmetry-breaking condition of Item 1 of Definition 9: we have to prove that it does not make $\mathcal{A}_P$ lose some essential solutions.

We proceed by inductively defining a particular sequence $\overline{\Upsilon} = \langle \Upsilon_0, \ldots, \Upsilon_{m+1} \rangle$ of sets of viewpoints. Then, we prove that each Υ_i is a state of $\mathcal{A}_P$, and that $\overline{\Upsilon}$ is an accepting run of $\mathcal{A}_P$. We also define (again, inductively) a sequence of covers of $\mathcal{T}_\Gamma$, which will be used for the inductive construction of $\overline{\Upsilon}$: for $i \in \{0, \ldots, m\}$, we define the cover $\{\mathcal{T}_\mathbb{V}^i\}_{\mathbb{V} \in \Upsilon_i}$ of $\mathcal{T}_\Gamma$, where, intuitively, $\mathcal{T}_\mathbb{V}^i$ is the set of tokens in $\mathcal{T}_\Gamma$ whose satisfaction is being taken care of by $\mathbb{V}$ in Υ_i. At first, we define $\Upsilon_0 = \{(B_{\mathcal{R}(\tau)}, \mathit{minimals}(B_{\mathcal{R}(\tau)})) \mid \mathcal{R} \in S, \tau \in \mathcal{T}_\Gamma^\mathcal{R}\}$ *i.e.*, the set of *minimal* viewpoints over the blueprints of the existential statements involved in the satisfaction of P by Γ, and we define the cover $\{\mathcal{T}_\mathbb{V}^0\}_{\mathbb{V} \in \Upsilon_0}$ of $\mathcal{T}_\Gamma$ as follows: for every $\mathcal{R} \in S$ and $\tau \in \mathcal{T}_\Gamma^\mathcal{R}$, $\tau \in \mathcal{T}_{(B_{\mathcal{R}(\tau)}, \mathit{minimals}(B_{\mathcal{R}(\tau)}))}$. Then, for all $i \in \{0, \ldots, m\}$, we choose the following elements of the sequence as follows. For each $\mathbb{V} = \langle B_\mathcal{E}, K \rangle \in \Upsilon_i$ and $\tau \in \mathcal{T}_\mathbb{V}^i$ let us define the set $F_i^{\tau,\mathbb{V}} = \{x \in dom(B_\mathcal{E}) \mid \mathcal{L}_\tau^\mathcal{E}(x) = i\}$. It is possible to show that for each $\mathbb{V} \in \Upsilon_i$ and $\tau \in \mathcal{T}_\mathbb{V}^i$ there is a unique viewpoint, denoted by $\mathit{next}(\mathbb{V}, \tau)$, such that $\mathbb{V} \xrightarrow{\sigma_i} \mathbb{V}'$ and $\mathit{consumed}(\mathbb{V}, \mathbb{V}') = F_i^{\tau,\mathbb{V}}$. Then, we take $\Upsilon_{i+1} = \{\mathit{next}(\mathbb{V}, \tau) \mid \mathbb{V} \in \Upsilon^i, \tau \in \mathcal{T}_\mathbb{V}^i\}$, and we define the cover $\{\mathcal{T}_\mathbb{V}^{i+1}\}_{\mathbb{V} \in \Upsilon_{i+1}}$ of $\mathcal{T}_\Gamma$ as follows: for each $\mathbb{V} \in \Upsilon_i$ and $\tau \in \mathcal{T}_\mathbb{V}^i$, $\tau \in \mathit{next}(\mathbb{V}, \tau)$.

Now, we argue that each Υ_i satisfies the symmetry-breaking condition (Item 1 of Definition 9). Let $\mathbb{V} = \langle B_{\mathcal{E}}, K\rangle, \mathbb{V}' = \langle B_{\mathcal{E}}, K'\rangle \in \Upsilon_i$, with $K \neq K'$, and let $\tau \in \mathcal{T}^i_{\mathbb{V}}$ and $\tau' \in \mathcal{T}^i_{\mathbb{V}'}$. We can suppose *w.l.o.g.* that end-time$(\tau) \leq$ start-time(τ'). By Lemma 12, we can suppose as well that $\mathcal{L}_\tau \leq \mathcal{L}_{\tau'}$. Now, let $S_{\mathbb{V}}$ and $S_{\mathbb{V}'}$ be the set of elements *covered* by K and K', respectively, *i.e.*, $S_{\mathbb{V}} = \{x \in dom(\mathcal{P}_{\mathcal{E}}) \mid \exists k \in K \,.\, x \preceq_{B_{\mathcal{E}}} k\}$. For any $x \in dom(\mathcal{P}_{\mathcal{E}})$, $\mathcal{L}_\tau(x) \leq i$ iff $x \in S_{\mathbb{V}}$ and $\mathcal{L}_{\tau'}(x) \leq i$ iff $x \in S'_{\mathbb{V}}$, thanks to the way in which we defined Υ_i. Then, it follows that $S_{\mathbb{V}'} \subseteq S_{\mathbb{V}}$, which implies that K dominates K', hence $\mathbb{V}' \preceq \mathbb{V}$.

As a consequence, $\overline{\Upsilon}$ is a sequence of $\mathcal{A}_P$ states and we can check that $(\Upsilon_i, \sigma_i, \Upsilon_{i+1}) \in \Delta_P$ for all $i \in \{0, \ldots, m\}$. Thus, $\overline{\Upsilon}$ identifies a run of $\mathcal{A}_P$ if we replace Υ_0 with q^I_P.

To conclude the proof, we only need to show that the above run is accepting, *i.e.*, that Υ_{m+1} contains only *final* viewpoints. This is indeed ensured by construction, since $\mathcal{L}_\tau(x) \leq m$ for every token $\tau \in \mathcal{T}^{\mathcal{R}}_\Gamma$, every $\mathcal{R} \in S$, and every $x \in dom(\mathcal{P}_{\mathcal{R}(\tau)})$. Therefore, the word $\overline{\sigma}$ is accepted by $\mathcal{A}_P$. ◀

We can now finally state our main result.

▶ **Theorem 14** (Complexity of qualitative timeline-based planning)**.** *Whether a qualitative timeline-based planning problem P admits a solution plan can be decided in* polynomial space*.*

Proof. Given $P = (\mathsf{SV}, S)$, let $\mathcal{A}_P$ be its associated automaton as specified by Definition 9. By Lemma 13, we know that $\mathcal{L}(\mathcal{A}_P) \neq \varnothing$ if and only if P admits a solution plan. Then, we can build $\mathcal{A}_P$ and check for the emptiness of its language, which in turn consists of checking for the reachability of the final states. By Lemma 10, the size of $\mathcal{A}_P$ is at most *exponential* in the size of P. Since this automaton can be constructed *on-the-fly* and solving reachability requires logarithmic space in the size of the automaton, we get that the qualitative timeline-based planning can be decided in polynomial space. ◀

4 Hardness

In this section, we show that qualitative timeline-based planning is PSPACE-hard. The proof is by a reduction from the emptiness problem for the intersection of n finite automata that is known to be PSPACE-complete (see [14]).

▶ **Theorem 15** (Qualitative timeline-based planning is PSPACE-hard)**.** *Let $P = (\mathsf{SV}, S)$ be a* qualitative *timeline-based planning problem. Deciding whether P admits any solution plan is* PSPACE*-hard.*

Proof. We provide a reduction from the emptiness problem for the intersection of n finite automata. For the main definitions on finite automata, we refer the reader to [14].

For $i \in \{1, \ldots, n\}$, let $A_i = (\Sigma, Q_i, q^0_i, \delta_i, q^*_i)$ be a deterministic finite automaton, where Σ is a finite alphabet, Q_i is a finite set of states, q^0_i is the initial state, δ_i: $Q_i \times \Sigma \to Q_i$ is the transition function, and q^*_i is the final state.

Denote by $A = A_1 \times \ldots \times A_n$ the finite automaton obtained by the standard construction to capture the intersection of the languages $\mathcal{L}(A_1), \ldots, \mathcal{L}(A_n)$, where, for $i \in \{1, \ldots, n\}$, $\mathcal{L}(A_i)$ denotes the language accepted by A_i. Thus, $\mathcal{L}(A) = \mathcal{L}(A_1) \cap \ldots \cap \mathcal{L}(A_n)$.

We build a *qualitative* timeline-based planning problem P such that P admits a solution plan if and only if $\mathcal{L}(A) \neq \varnothing$. The overall idea is to model each finite automaton as a different state variable and then express intersection and acceptance by synchronization rules. More specifically, $P = (\mathsf{SV}, S)$ is defined as follows.

The set of state variables is $\mathsf{SV} = \{x_i \mid i \in \{1, \ldots, n\}\}$, *i.e.*, we take a variable x_i for each finite automaton A_i, with $i \in \{1, \ldots, n\}$. Each variable x_i is equal to (V_i, T_i, D_i), where:

1. $V_i = Q_i \times \Sigma$;
2. $D_i(v) = (1, +\infty)$, for all $v \in V_i$;
3. $T_i((q, \sigma)) = \{\delta_i(q, \sigma)\} \times \Sigma$, for all $(q, \sigma) \in V_i$.

The transition function of each state variable mirrors the transition function of the corresponding automaton, while handling the fact that automata are meant to read words over Σ while state variables only represent *state machines*, with no language recognition semantics.

The set S contains the following synchronization rules, which are designed in such a way that state variables change their values synchronously.

The first rule requires the existence of two sets of tokens, each containing exactly a token from each state variable and such that (i) the first set maps an initial state for each automaton, (ii) the second set maps a final state for each automaton, (iii) the tokens from the first set precede those from the second set, and (iv) the tokens in each set start and end at the same time. Formally:

$$\top \rightarrow \bigvee_{\sigma,\sigma' \in \Sigma} \exists a_1^0[x_1 = (q_1^0, \sigma)] \cdots a_n^0[x_n = (q_n^0, \sigma)] a_1^*[x_1 = (q_1^*, \sigma')] \cdots a_n^*[x_n = (q_n^*, \sigma')] \; .$$
$$\mathrm{end}(a_1^0) \leq \mathrm{start}(a_1^*) \wedge \bigwedge_{i=1}^{n-1} a_i^0 = a_{i+1}^0 \wedge \bigwedge_{i=1}^{n-1} a_i^* = a_{i+1}^* . \tag{1}$$

The remaining rules just synchronize the different state variables so that they are aligned over tokens that refer to the same input symbol for the corresponding automaton:

$$a_i[x_i = (q_i, \sigma)] \rightarrow \bigvee_{q_{i+1} \in Q_{i+1}} \exists a_{i+1}[x_{i+1} = (q_{i+1}, \sigma)] \; . \; a_i = a_{i+1} \tag{2}$$

for each $q_i \in Q_i$, $\sigma \in \Sigma$ and $i \in \{1, \ldots, n-1\}$.

To complete the proof, it suffices to show that the above construction is correct, that is, P admits a solution plan if and only if $\mathcal{L}(A) \neq \varnothing$

We first show that if $\mathcal{L}(A) \neq \varnothing$, then P admits a solution plan. To this end, consider a word $\overline{\sigma} = \sigma^1 \ldots \sigma^m$ accepted by A. By definition, for $i \in \{1, \ldots, n\}$, we know that $\overline{\sigma}$ is accepted by A_i. Let $\overline{q_i} = \langle q_i^0, \ldots, q_i^m \rangle$ be the sequence of states visited along the run of A_i over $\overline{\sigma}$. Since $\overline{\sigma}$ is accepted by A_i, q_i^m must be the final state q_i^*.

Now, for all $i \in \{1, \ldots, n\}$, let:

$$\mathsf{T}_i = \langle (x_i, (q_i^0, \sigma^1), 1), (x_i, (q_i^1, \sigma^2), 1), \ldots, (x_i, (q_i^{m-1}, \sigma^m), 1), (x_i, (q_i^m, \sigma^*), 1) \rangle$$

be the timeline corresponding to $\overline{\sigma}$. It can be observed that, by construction, T_i satisfies the transition function of x_i. Moreover, the synchronization rule (1) defined above is satisfied, since each timeline T_i starts with a token where $x_i = (q_i^0, \sigma^1)$ and ends with a token where $x_i = (q_i^*, \sigma^*)$. Moreover, by construction, the tokens are all of the same duration, and overlapping tokens have the same σ component; thus, the second set of synchronization rules (2) is satisfied as well. Hence, the plan consisting of the T_i timelines is a solution plan for P.

In order to show that if P admits a solution plan, then $\mathcal{L}(A) \neq \varnothing$, consider a solution plan for P consisting of a set of timelines $\mathsf{T}_i = \langle \tau_i^1, \ldots, \tau_i^{m_i} \rangle$, one for each $x_i \in \mathsf{SV}$. By the rules (2), all the tokens of these timelines can be seen as being aligned one over the other forming a grid, where each *column* shares a common symbol σ. Moreover, by the triggerless rule (1), there are two columns of such a grid, say them h and k, with $h < k$, containing, respectively, the initial states and the final states for the respective automata A_i. Let $\overline{\sigma} = \sigma^h \ldots \sigma^{k-1}$ be the sequence of symbols occurring in columns h to $k-1$ and let q_i^j be the state of A_i

associated with token τ_i^j, for $i \in \{1, \ldots, n\}$ and $j \in \{h, \ldots, k\}$. Finally, by construction, at each step the transition function of each state variable enforces the token following any token of the form (q, σ) to be of the form (q', σ'), where $q' = \delta(q, \sigma)$, *i.e.*, the evolution of the timeline mirrors the transition function of the automaton. Thus, for all $i \in \{1, \ldots, n\}$, $\overline{q}_i = \langle q_i^h, \ldots, q_i^k \rangle$ is an accepting run of A_i over $\overline{\sigma}$. Thus, we conclude that $\overline{\sigma} \in \mathcal{L}(A)$. ◀

5 Concluding remarks

In this paper, we show that the problem of checking the existence of a plan for the *qualitative* fragment of timeline-based planning is PSPACE-complete.

The key step in the decision procedure builds a finite automaton that accepts a word encoding a plan if and only if the plan is a solution of the given instance of the planning problem. The construction is inspired by the one used by Della Monica *et al.* [5] to prove the EXPSPACE-completeness of the general quantitative problem, but adapted to exploit the distinctive features of the qualitative setting. In particular, blueprints were linear orders in the general case, accounting for all possible combinations of distances between pairs of endpoints satisfying the quantitative constraints of the problem. Here, blueprints are preorders, which compactly represent all the possible ways of matching a particular disjunct of a rule, leading to a smaller automaton.

The automata-theoretic construction of our solution has some interesting consequences. It provides a direct algorithm to generate a solution plan by exploiting the standard machinery to decide the emptiness of finite automata and, moreover, it can be used as a basis for interesting future research directions. For example, one may show how to perform model checking of timeline-based systems against Linear Temporal Logic (LTL), still in polynomial space.

The achieved result sheds some light on how a problem of such a high complexity could form the basis of planning systems that have been deployed in real-world scenarios for the last three-decades. The *quantitative* aspect of the problem accounts for a great part of the complexity, and while temporal reasoning is predominant in these applications, the *magnitude* of the involved timestamps does not need to be significantly high. A proper parameterized complexity analysis of the problem would complete the picture in this regard.

Last but not least, the qualitative fragment appears to be a quite expressive language on its own, powerful enough to express an interesting class of linear-time temporal properties including those captured by LTL. It would be interesting to establish the exact relationship with LTL and possibly characterize this logic in terms of a proper fragment of timeline-based planning. From a logical standpoint, the synchronization rules can be seen as a fragment of first-order logic with one successor relation and one quantifier alternation (specifically, with $\forall\exists$ alternation). A key aspect that can be observed is that disjunctions are only allowed between existentially quantified formulas and, by relaxing this limitation, the complexity lower bound seems to rise to EXPSPACE again even in the qualitative setting.

References

1 James F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, 1983. doi:10.1145/182.358434.

2 Javier Barreiro, Matthew Boyce, Minh Do, Jeremy Frank, Michael Iatauro, Tatiana Kichkaylo, Paul Morris, James Ong, Emilio Remolina, Tristan Smith, and David Smith. EUROPA: A Platform for AI Planning, Scheduling, Constraint Programming, and Optimization. In *Proceedings of the 4th International Competition on Knowledge Engineering for Planning and Scheduling*, 2012.

3 S. Chien, G. Rabideau, R. Knight, R. Sherwood, B. Engelhardt, D. Mutz, T. Estlin, B. Smith, F. Fisher, T. Barrett, G. Stebbins, and D. Tran. Aspen - automating space mission operations using automated planning and scheduling. In *Proceedings of the International Conference on Space Operations*, 2000.

4 Steve A. Chien, Gregg Rabideau, Daniel Tran, Martina Troesch, Joshua Doubleday, Federico Nespoli, Miguel Perez Ayucar, Marc Costa Sitja, Claire Vallat, Bernhard Geiger, Nico Altobelli, Manuel Fernandez, Fran Vallejo, Rafael Andres, and Michael Kueppers. Activity-based scheduling of science campaigns for the rosetta orbiter. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence*, pages 4416–4422. AAAI Press, 2015.

5 Dario Della Monica, Nicola Gigante, Angelo Montanari, and Pietro Sala. A novel automata-theoretic approach to timeline-based planning. In Michael Thielscher, Francesca Toni, and Frank Wolter, editors, *Proceedings of the 16th International Conference on Principles of Knowledge Representation and Reasoning*, pages 541–550. AAAI Press, 2018.

6 Alessandro Donati, Nicola Policella, Erhard Rabenau, Giovanni Righini, and Emanuele Tresoldi. An automatic planning and scheduling system for the mars express uplink scheduling problem. *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, 41(6):942–954, 2011. doi:10.1109/TSMCC.2011.2114880.

7 Richard Fikes and Nils J. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2(3/4):189–208, 1971. doi:10.1016/0004-3702(71)90010-5.

8 Simone Fratini, Amedeo Cesta, Andrea Orlandini, Riccardo Rasconi, and Riccardo De Benedictis. Apsi-based deliberation in goal oriented autonomous controllers. In *ASTRA 2011*, volume 11. ESA, 2011.

9 Simone Fratini and L. Donati. Apsi timeline representation framework v. 3.0. Technical report, European Space Agency - ESOC, 2011.

10 Malik Ghallab, Dana S. Nau, and Paolo Traverso. *Automated Planning and Acting*. Cambridge University Press, 2016. URL: http://www.cambridge.org/de/academic/subjects/computer-science/artificial-intelligence-and-natural-language-processing/automated-planning-and-acting?format=HB.

11 Nicola Gigante, Angelo Montanari, Marta Cialdea Mayer, and Andrea Orlandini. Timelines are expressive enough to capture action-based temporal planning. In Curtis E. Dyreson, Michael R. Hansen, and Luke Hunsberger, editors, *Proceedings of the 23rd International Symposium on Temporal Representation and Reasoning*, pages 100–109. IEEE Computer Society, 2016. doi:10.1109/TIME.2016.18.

12 Nicola Gigante, Angelo Montanari, Marta Cialdea Mayer, and Andrea Orlandini. Complexity of timeline-based planning. In Laura Barbulescu, Jeremy Frank, Mausam, and Stephen F. Smith, editors, *Proceedings of the 27th International Conference on Automated Planning and Scheduling*, pages 116–124. AAAI Press, 2017.

13 Nicola Gigante, Angelo Montanari, Andrea Orlandini, Marta Cialdea Mayer, and Mark Reynolds. On timeline-based games and their complexity. *Theor. Comput. Sci.*, 815:247–269, 2020. doi:10.1016/j.tcs.2020.02.011.

14 John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to automata theory, languages, and computation, 3rd Edition*. Pearson international edition. Addison-Wesley, 2007.

15 Drew McDermott, Malik Ghallab, Adele Howe, Craig Knoblock, Ashwin Ram, Manuela Veloso, Daniel Weld, and David Wilkins. PDDL - The Planning Domain Definition Language. Technical Report Technical Report TR98003, Yale Center for Computational Vision and Control, 1997.

16 Nicola Muscettola. HSTS: Integrating Planning and Scheduling. In Monte Zweben and Mark S. Fox, editors, *Intelligent Scheduling*, chapter 6, pages 169–212. Morgan Kaufmann, 1994.