Synthesis of Timeline-Based Planning Strategies Avoiding Determinization*

Renato Acampora
University of Udine, Italy
renato.acampora@uniud.it

Dario Della Monica
University of Udine, Italy
dario.dellamonica@uniud.it

Luca Geatti
University of Udine, Italy
luca.geatti@uniud.it

Nicola Gigante
Free University of Bozen-Bolzano, Italy
nicola.gigante@unibz.it

Angelo Montanari
University of Udine, Italy
angelo.montanari@uniud.it

Pietro Sala
University of Verona, Italy
pietro.sala@univr.it

Qualitative timeline-based planning models domains as sets of independent, but interacting, components whose behaviors over time, the timelines, are governed by sets of qualitative temporal constraints (ordering relations), called synchronization rules. Its plan-existence problem has been shown to be PSPACE-complete; in particular, PSPACE-membership has been proved via reduction to the nonemptiness problem for nondeterministic finite automata. However, nondeterministic automata cannot be directly used to synthesize planning strategies as a costly determinization step is needed. In this paper, we identify a large fragment of qualitative timeline-based planning whose plan-existence problem can be directly mapped into the nonemptiness problem of deterministic finite automata, which can then be exploited to synthesize strategies. In addition, we identify a maximal subset of Allen's relations that fits into such a deterministic fragment.

1 Introduction

Timeline-based planning is an approach that originally emerged and developed in the context of planning and scheduling of *space* operations [16]. In contrast to common action-based formalisms, such as PDDL [9], timeline-based languages do not make a distinction between actions, states, and goals. Rather, the domain is modeled as a set of independent, but interacting, components whose behavior over time, the timelines, is governed by a set of temporal constraints. It is worth pointing out that timeline-based planning was born with an application-oriented flavor, with various successful stories, and only relatively recently some foundational work about its expressiveness and complexity has been produced. The present paper aims at bringing back theory to practice by identifying expressive enough and computationally well-behaved fragments.

Timeline-based planning has been successfully employed by planning systems developed at NASA [5, 6] and at ESA [10] for both short- to long-term mission planning and on-board autonomy. More recently, timeline-based planning systems such as PLATINUm [18] are being employed in collaborative robotics applications [19]. All these applications share a deep reliance on *temporal reasoning* and the need for a tight integration of planning with *execution*, both features of the timeline-based framework. The latter feature is usually achieved by the use of *flexible timelines*, which represent a set of possible executions of the system that differ in the precise timing of the events, hence handling the intrinsic *temporal uncertainty* of the environment. A formal account of timeline-based planning with uncertainty

*This work is partially supported by the INdAM-GNCS Project *Analisi simbolica e numerica di sistemi ciberfisici* (project n. CUP_E53C22001930001).

A. Achilleos and A. Francalanza (Eds.): Fifteenth International Symposium on Games, Automata, Logics, and Formal Verification (GandALF 2024).
EPTCS 409, 2024, pp. 5–18, doi:10.4204/EPTCS.409.5

has been provided by [7], and much theoretical research followed, including *complexity* [3, 4, 12] and *expressiveness* [14, 11] analyses, based on such a formalization, which is the one we use here as well.

To extend the reactivity and adaptability of timeline-based systems beyond temporal uncertainty, the framework of *timeline-based games* has been recently proposed. In timeline-based games, the system player tries to build a set of timelines satisfying the constraints independently from the choices of the environment player. This framework allows one to handle general nondeterministic environments in the timeline-based setting. However, this expressive power comes at the cost of increasing the complexity of the problem. While the plan-existence problem for timeline-based planning is EXPTIME-complete [12], deciding the existence of strategies for timeline-based games is 2EXPTIME-complete [13], and a controller synthesis algorithm exists that runs in doubly exponential time [1].

Such a high complexity motivates the search for simpler fragments that can nevertheless be useful in practical scenarios. One of these is the *qualitative* fragment, where temporal constraints only concern the relative order between events and not their distance. The qualitative fragment already proved itself to be easier for the plan-existence problem, being PSPACE-complete [8], and this makes it a natural candidate for the search of a good fragment for the strategy existence problem.

A *deterministic* arena is crucial to synthesize a non-clairvoyant strategy in *reactive synthesis* problems (see, for instance, [17]). However, determinizing the nondeterministic (exponentially sized) automaton built for the qualitative case in [8] would cause an exponential blowup, thus resulting in a procedure of doubly-exponential complexity. In this paper, we show that, by imposing some natural restrictions on the set of temporal constraints of the qualitative fragment, it is possible to lower the complexity of the strategy existence problem to EXPTIME. We show that, on the one hand, these restrictions are *sufficient* to directly synthesize a *deterministic* finite automaton (DFA) of singly-exponential size, thus usable as an arena to play the game in an asymptotically optimal way, and, on the other hand, the resulting fragment is expressive enough to capture a large subset of Allen's relations [2], defined in Section 7.

The rest of the paper is organized as follows. Section 2 recalls some background knowledge on timeline-based planning. Section 3 defines the considered fragment, that directly maps into a DFA of singly exponential size. Section 4 gives a word encoding of timelines, and vice versa. Section 5 builds an automaton to recognize plans, and Section 6 shows how to construct an automaton that accepts solution plans. Section 7 identifies the maximal subset of Allen's relations which is captured by the fragment of Section 3. Finally, Section 8 summarizes the main contributions of the work and discusses possible future developments.

2 Background

In this section, we recall the basic notions of timeline-based planning and of its qualitative variant.

2.1 Timeline-Based Planning

The key element of the framework is the notion of *state variable*. Let $\mathbb{N}^+$ be the set of positive natural numbers.

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* $x = v$.

A *timeline* is a finite sequence of *tokens*, each denoting a value v and (the duration of) a time interval d, that describes how a state variable x behaves over time.

Definition 2 (Tokens and 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, with* $D_x(v) = (d_{min}^{x=v}, d_{max}^{x=v})$ *and* $d_{min}^{x=v} \leq d \leq d_{max}^{x=v}$*. A* timeline *for a state variable* x *is a finite sequence* $\mathsf{T} = \langle \tau_1, \ldots, \tau_k \rangle$ *of tokens for* x*, for some* $k \in \mathbb{N}$*, such that, for any* $1 \leq i < k$*, if* $\tau_i = (x, v_i, d_i)$*, then* $v_{i+1} \in T_x(v_i)$.

For any timeline $\mathsf{T} = \langle \tau_1, \ldots, \tau_k \rangle$ and any token $\tau_i = (x, v_i, d_i)$ in T, we define the functions $\text{start}(\mathsf{T}, i) = \sum_{j=1}^{i-1} d_j$ and $\text{end}(\mathsf{T}, i) = \text{start}(\mathsf{T}, i) + d_i$. We call the *horizon* of T the end time of the last token in T, that is, $\text{end}(\mathsf{T}, k)$. We write $\text{start}(\tau_i)$ and $\text{end}(\tau_i)$ to indicate $\text{start}(\mathsf{T}, i)$ and $\text{end}(\mathsf{T}, i)$, respectively, when there is no ambiguity.

The overall behavior of state variables is subject to a set of temporal constraints known as *synchronization rules* (or simply *rules*). We start by defining their basic building blocks. Let $\mathscr{N}$ be a finite set of *token names*. *Atoms* are formulas of the following form:

$$\begin{aligned} \mathit{atom} &:= \mathit{term} \leq_{l,u} \mathit{term} \mid \mathit{term} <_{l,u} \mathit{term} \\ \mathit{term} &:= \text{start}(a) \mid \text{end}(a) \mid t \end{aligned}$$

where $a \in \mathscr{N}$, $l, t \in \mathbb{N}$, and $u \in \mathbb{N} \cup \{+\infty\}$. As an example, atom $\text{start}(a) \leq_{l,u} \text{end}(b)$ (resp., $\text{start}(a) <_{l,u} \text{end}(b)$) relates tokens a and b by stating that the end of b cannot precede (resp., must succeed) the beginning of a, and the distance between these two endpoints must be at least l and at most u. An atom $\mathit{term} \leq_{l,u} \mathit{term}$, with $l = 0$ and $u = +\infty$, is *qualitative* (the subscript is usually omitted in this case).

An *existential statement* $\mathscr{E}$ is a constraint of the form:

$$\exists a_1[x_1 = v_1]a_2[x_2 = v_2] \ldots a_n[x_n = v_n].\, \mathscr{C}$$

where $x_1, \ldots, x_n$ are state variables, $v_1, \ldots, v_n$ are values, with $v_i \in V_{x_i}$, $a_1, \ldots, a_n$ are symbols from the set $\mathscr{N}$ of token names, and $\mathscr{C}$ is a finite conjunction of *atoms*, involving only tokens $a_1, \ldots, a_n$, plus, possibly, the *trigger token* (usually denoted by a_0) of the *synchronization rule* in which the existential statement is embedded, as described below.[1]

Intuitively, an existential statement asks for the existence of tokens $a_1, a_2, \ldots, a_n$ whose state variables take the corresponding values $v_1, v_2, \ldots, v_n$ and are such that their start and end times satisfy the atoms in $\mathscr{C}$.

Synchronization rules are clauses of one of the following forms:

$$\begin{aligned} a_0[x_0 = v_0] &\to \mathscr{E}_1 \vee \mathscr{E}_2 \vee \ldots \vee \mathscr{E}_k \\ \top &\to \mathscr{E}_1 \vee \mathscr{E}_2 \vee \ldots \vee \mathscr{E}_k \end{aligned}$$

where $a_0 \in \mathscr{N}$, x_0 is a state variable, $v_0 \in V_{x_0}$, and $\mathscr{E}_i$ is an existential statement, for each $1 \leq i \leq k$. In the former case, $a_0[x_0 = v_0]$ is called *trigger* and a_0 is the *trigger token*, and the rule is considered *satisfied* if *for all* the tokens a_0 for which the variable x_0 takes the value v_0, at least one of the existential statements is satisfied. In the latter case, the rule is said to be *triggerless*, and it states the truth of the body without any precondition.[2] We refer the reader to [7] for a formal account of the semantics of the rules.

[1] W.l.o.g., we assume that if a token a appears in the quantification prefix $\exists a_1[x_1 = v_1]a_2[x_2 = v_2] \ldots a_n[x_n = v_n]$ of $\mathscr{E}$, then at least one among $\text{start}(a)$ and $\text{end}(a)$ occurs in one of its atoms.

[2] W.l.o.g., for non-triggerless rules, we assume that both $\text{start}(a_0)$ and $\text{end}(a_0)$ occur in all of its existential statements.

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)**.** *A* timeline-based planning problem *is defined as a pair* $P = (\mathsf{SV}, S)$*, where* SV *is a set of state variables and* S *is a set of synchronization rules involving state variables in* SV*.*

A *solution plan* for a given timeline-based planning problem is a set of timelines, one for each state variable, that satisfies all the synchronization rules.

Definition 4 (Plan and solution plan)**.** *A* plan *over a set of state variables* SV *is a finite set of timelines with the same horizon, one for each state variable* $x \in \mathsf{SV}$*. A* solution plan *for a timeline-based planning problem* $P = (\mathsf{SV}, S)$ *is a plan over* SV *such that all the rules in* S *are satisfied.*

The problem of determining whether a solution plan exists for a given timeline-based planning problem is EXPSPACE-complete [12].

Definition 5 (Qualitative timeline-based planning)**.** *A timeline-based planning problem* $P = (\mathsf{SV}, S)$ *is said to be* qualitative *if the following conditions hold:*

1. $D_x(v) = (1, +\infty)$*, for all state variables* $x \in \mathsf{SV}$ *and* $v \in V_x$*.*
2. *all synchronization rules in* S *involve only* qualitative *atoms.*

Unlike timeline-based planning, such a qualitative variant is PSPACE-complete [8]. A reduction of qualitative timeline-based planning to the nonemptiness problem for non-deterministic finite automata (NFA) has been provided in [8].

3 A Well-Behaved Fragment

In this section, we introduce a meaningful fragment of qualitative timeline-based planning for which we will show that it is possible to construct DFAs of singly exponential size.

The fragment is characterized by means of some conditions on the admissible patterns of synchronization rules (eager rules). The distinctive feature of eager rules is that they can be checked using an eager/greedy strategy, that is, when a relevant event (start/end of a token involved in some atom) occurs, we are guaranteed that the starting/ending point of such a token is useful for rule satisfaction. Instead, in case of non-eager rules, it may happen that a relevant event happens that is not useful for rule satisfaction: some analogous event in the future will be.

W.l.o.g., we assume that no constraint of the forms $\text{start}(a) \leq \text{end}(a)$ and $\text{start}(a) < \text{end}(a)$ occurs explicitly in synchronization rules, even though they hold tacitly, as they follow from the definition of token (Definition 2).

As a preliminary step, we define a sort of transitive closure of a clause. First, by slightly abusing the notation, we identify a clause $\mathscr{C}$ with the finite set of atoms occurring in it. Let t, t_1, t_2, t_3 be terms of the form $\text{start}(a)$ or $\text{end}(a)$, with $a \in \mathscr{N}$. We denote by $\hat{\mathscr{C}}$ the *transitive closure* of $\mathscr{C}$, defined as the smallest set of atoms including $\mathscr{C}$ and such that: *(i)* if term t occurs in $\mathscr{C}$, then atom $t \leq t$ belongs to $\hat{\mathscr{C}}$, *(ii)* if terms $\text{start}(a)$ and $\text{end}(a)$ both occur in $\mathscr{C}$ for some token name a, then atom $\text{start}(a) < \text{end}(a)$ belongs to $\hat{\mathscr{C}}$, *(iii)* if atom $t_1 < t_2$ belongs to $\hat{\mathscr{C}}$, then atom $t_1 \leq t_2$ belongs to $\hat{\mathscr{C}}$ as well, *(iv)* if atoms $t_1 \leq t_2$ and $t_2 \leq t_3$ belong to $\hat{\mathscr{C}}$, then atom $t_1 \leq t_3$ belongs to $\hat{\mathscr{C}}$ as well, *(v)* if atoms $t_1 < t_2$ and $t_2 \leq t_3$ belong to $\hat{\mathscr{C}}$, then atom $t_1 < t_3$ belongs to $\hat{\mathscr{C}}$ as well, *(vi)* if atoms $t_1 \leq t_2$ and $t_2 < t_3$ belong to $\hat{\mathscr{C}}$, then atom $t_1 < t_3$ belongs to $\hat{\mathscr{C}}$ as well.[3]

[3] W.l.o.g., we assume that $\hat{\mathscr{C}}$ is consistent, *i.e.*, it admits at least a solution. We point out that this check can be done in polynomial time, since it is an instance of linear programming.

Notice that, in some particular cases, condition *(ii)* may introduce in the closure of a clause atoms of the form $\text{start}(a) < \text{end}(a)$, which, according to our assumption, do not belong to any clause.

Let us now define the core notion of *eager rule*.

Definition 6 (Eager rules)**.** *Let $\mathscr{R}$ be a synchronization rule and let $\mathscr{C}_1, \ldots, \mathscr{C}_k$ be the clauses occurring in its existential statements. We say that $\mathscr{R}$ is* eager *if and only if, for all $\mathscr{C} \in \{\mathscr{C}_1, \ldots, \mathscr{C}_k\}$ and $a_1, a_2 \in \mathscr{N}$ appearing in $\mathscr{C}$, the following conditions hold:*

1. *if both a_1 and a_2 are non-trigger tokens and $\{\text{start}(a_2) \leq \text{end}(a_1), \text{end}(a_1) \leq \text{end}(a_2)\} \subseteq \hat{\mathscr{C}}$, then $\text{end}(a_1) \leq \text{start}(a_2) \in \hat{\mathscr{C}}$ (i.e., the end of a_1 and the start of a_2 coincide),*
2. *if a_1 is either a trigger token or a non-trigger one, a_2 is a non-trigger token, and $\{\text{start}(a_2) \leq \text{start}(a_1), \text{start}(a_1) \leq \text{end}(a_2)\} \subseteq \hat{\mathscr{C}}$, then $\text{start}(a_1) \leq \text{start}(a_2) \in \hat{\mathscr{C}}$ (i.e., a_1 and a_2 start together), and*
3. *if a_1 is a trigger token, a_2 is a non-trigger one, and $\{\text{start}(a_1) \leq \text{start}(a_2), \text{end}(a_1) \leq \text{end}(a_2)\} \subseteq \hat{\mathscr{C}}$, then $\text{start}(a_2) \leq \text{start}(a_1) \in \hat{\mathscr{C}}$ (i.e., a_1 and a_2 start together).*

We define the *eager fragment* of a qualitative timeline-based planning problem as the set of qualitative timeline-based planning problems $P = (\mathsf{SV}, S)$ such that S contains only eager rules.

An explanation of the restrictions in Definition 6 is due. Given a non-trigger token a_2, Condition 1 forces any other non-trigger token a_1 ending during a_2 (that is, such that $\text{start}(a_2) \leq \text{end}(a_1) \leq \text{end}(a_2)$) to end exactly when a_2 starts, while Condition 2 forces any other (trigger or non-trigger) token a_1 starting during a_2 (that is, such that $\text{start}(a_2) \leq \text{start}(a_1) \leq \text{end}(a_2)$) to start simultaneously to a_2. Finally, whenever a non-trigger token a_2 starts during a trigger token a_1 and ends not before the end of a_1, Condition 3 forces the two tokens to start at the same time.

Conditions 1, 2, and 3 suffice to obtain a singly exponential DFA, whose construction will be illustrated in the next sections. We give here a short intuitive account of the rationale of the above conditions.

Consider the following rule:

$$a_0[x_0 = v_0] \rightarrow \exists a_1[x_1 = v_1]. \\ (\text{start}(a_0) = \text{start}(a_1) \wedge \text{end}(a_0) \leq \text{end}(a_1)),$$

where $\text{start}(a_0) = \text{start}(a_1)$ is an abbreviation for $\text{start}(a_0) \leq \text{start}(a_1) \wedge \text{start}(a_1) \leq \text{start}(a_0)$. This rule is eager because Conditions 1, 2, and 3 are fulfilled; in particular, we have that $\text{start}(a_0) = \text{start}(a_1)$. This is crucial for any DFA $\mathscr{A}$ recognizing solution plans, because, when $\mathscr{A}$ reads the event $\text{start}(a_0)$, it can *eagerly* and *deterministically* go to a state representing the fact that both $\text{start}(a_0)$ and $\text{start}(a_1)$ have happened. Moreover, if later it reads the event $\text{end}(a_1)$, but it has not read $\text{end}(a_0)$ yet, then it transitions to a rejecting state, that is, a state from which it cannot accept any plan.

Let us provide now an example of a non-eager rule that cannot be checked in an eager/greedy fashion. Consider the rule obtained from the above one by replacing $=$ with $\leq$:

$$a_0[x_0 = v_0] \rightarrow \exists a_1[x_1 = v_1]. \\ (\text{start}(a_0) \leq \text{start}(a_1) \wedge \text{end}(a_0) \leq \text{end}(a_1)).$$

This rule is *not* eager, because atom $\text{start}(a_1) \leq \text{start}(a_0)$ does not belong to $\hat{\mathscr{C}}$ (Condition 3 is violated). Indeed, for this rule, a DFA $\mathscr{A}$ that first reads event $\text{start}(a_0)$, but not $\text{start}(a_1)$, and then, strictly after, reads event $\text{start}(a_1)$ has to *nondeterministically* guess the order between the end of such a token a_1 and the end of a_0, making the construction of an automaton of singly exponential size impossible in the

general case. Indeed, if token a_1 ends before token a_0, the rule is not satisfied, but we cannot exclude the existence of another token for $x_1 = v_1$ that starts after that one and ends after the end of a_0, thus satisfying the rule.

We conclude by showing that excluding constraints of the forms $\text{start}(a) \leq \text{end}(a)$ and $\text{start}(a) < \text{end}(a)$ from clauses makes it sometimes possible to turn an otherwise non-eager rule into an eager one. As an example, rule $a_0[x_0 = v_0] \rightarrow \exists a_1[x_1 = v_1].(\text{start}(a_1) < \text{end}(a_1) \wedge \text{start}(a_0) = \text{end}(a_1))$ is not eager (Condition 2 is violated); however, it can be rewritten as $a_0[x_0 = v_0] \rightarrow \exists a_1[x_1 = v_1].\, \text{start}(a_0) = \text{end}(a_1)$, which is eager.

In what follows, we give a reduction from the plan-existence problem for the eager fragment of the qualitative timeline-based planning problem to the nonemptiness problem of DFAs of *singly exponential* size with respect to the original problem. The approach is inspired by those in [8, 14] for non-eager timeline-based planning problems, where an NFA of exponential size is built for any timeline-based planning problem. However, the reductions presented there use nondeterministic automata, which cannot be used as arenas to solve timeline-based games without a previous determinization step that would cause a second exponential blowup.

First, we show how to encode timelines and plans as finite words, and vice versa (Section 4). Then, given a planning problem P, we show how to build a DFA whose language encodes the set of solution plans for P. The DFA consists of the intersection of two DFAs: one aims at verifying the constraint on the alternation of token values expressed by functions T_x, for $x \in \mathsf{SV}$, as well as that the word correctly encodes a plan over SV (Section 5); the other one verifies that the encoded plan is indeed a solution plan for P (Section 6).

From now on, we consider only qualitative timeline-based planning problems belonging to the eager fragment and, for the sake of brevity, we sometimes refer to them simply as *planning problems*.

4 From Plans to Finite Words and *Vice Versa*

In this section, as a first step in the construction of the DFA corresponding to an eager qualitative timeline-based planning problem, we show how to encode timelines and plans as *words* that can be recognized by an automaton, and *vice versa*.

Let $P = (\mathsf{SV}, S)$ be an eager qualitative timeline-based planning problem, and let $V = \cup_{x \in \mathsf{SV}} V_x$. We define the *initial alphabet* Σ^I_{SV} as $(\{-\} \times V)^{\mathsf{SV}}$, that is the set of functions from SV to $(\{-\} \times V)$.[4] Similarly, we define the *non-initial alphabet* Σ^N_{SV} as $((V \times V) \cup \{\circlearrowleft\})^{\mathsf{SV}}$, where the pairs $(v, v') \in V \times V$ are supposed to represent the value v of the token that just ended and the value v' of the token that has just started, and $\circlearrowleft$ represents the fact that the value for the state variable has not changed. The *input alphabet* (or, simply, *alphabet*) associated with SV and denoted by Σ_{SV} is the union $\Sigma^I_{\mathsf{SV}} \cup \Sigma^N_{\mathsf{SV}}$. Observe that the size of the alphabet Σ_{SV} is at most exponential in the size of SV, precisely $|\Sigma_{\mathsf{SV}}| = |\Sigma^I_{\mathsf{SV}}| + |\Sigma^N_{\mathsf{SV}}| = |V|^{|\mathsf{SV}|} + (|V|^2 + 1)^{|\mathsf{SV}|}$.

We now show how to encode the basic structure[5] underlying each plan over SV as a word in $\Sigma^I_{\mathsf{SV}} \cdot (\Sigma^N_{\mathsf{SV}})^* \cup \{\varepsilon\}$, where ε is the empty word (and corresponds to the empty plan), $(\Sigma^N_{\mathsf{SV}})^*$ is the Kleene's closure of Σ^N_{SV}, and $\cdot$ denotes the concatenation symbol. Intuitively, let ν be the symbol at position i of a word $\sigma \in \Sigma^I_{\mathsf{SV}} \cdot (\Sigma^N_{\mathsf{SV}})^* \cup \{\varepsilon\}$. Then, if $\nu(x) = (v, v')$ for some $x \in \mathsf{SV}$, then at time i a new token begins in the timeline for x with value v'; instead, if $\nu(x) = \circlearrowleft$, then no change happens at time i in the timeline

[4] The symbol $\{-\}$ is a technicality that allows us to consider pairs instead of just values in V.

[5] With "basic structure" we refer to the fact that, in this section, we neither take into account the transition functions T_x of state variables nor their domains V_x (*cf.* Definition 1), which will be dealt with in Section 5.

for x, meaning that no token ends at that time point in the timeline for x. The value v of the token ending at time i will be used later in the construction of the automata.

We remark that not all words in $\Sigma^I_{\mathsf{SV}} \cdot (\Sigma^N_{\mathsf{SV}})^* \cup \{\varepsilon\}$ correspond to plans over SV: for a word to correctly encode a plan, the information carried by the word about the value of a starting token and the one associated to the end of the same token must coincide. Formally, given a word $\sigma = \langle \sigma_0, \ldots, \sigma_{|\sigma|-1} \rangle \in \Sigma^I_{\mathsf{SV}} \cdot (\Sigma^N_{\mathsf{SV}})^* \cup \{\varepsilon\}$ and a state variable $x \in \mathsf{SV}$, let $\mathit{changes}(x) = (i^x_0, i^x_1, \ldots, i^x_{k^x-1})$, for some $k^x \in \mathbb{N}$, be the increasing sequence of positions where x changes, *i.e.*, $\sigma_i(x) \neq \circlearrowleft$ if and only if $i \in \mathit{changes}(x)$, for all $i \in \{0, \ldots, |\sigma| - 1\}$. We denote by v^x_i and $\hat{v}^x_i$ the first and the second component of $\sigma_i(x)$, respectively, for all $x \in \mathsf{SV}$ and $i \in \mathit{changes}(x)$. We omit superscripts ^{x} when there is no risk of ambiguity.

Definition 7 (Words weakly-encoding plans)**.** *Let $\sigma \in \Sigma^I_{\mathsf{SV}} \cdot (\Sigma^N_{\mathsf{SV}})^*$ and let $\mathit{changes}(x) = (i_0, i_1, \ldots, i_{k-1})$. We say that σ* weakly-encodes a plan over SV *if $\hat{v}_{i_{h-1}} = v_{i_h}$ for all $x \in \mathsf{SV}$ and $h \in \{1, \ldots, k-1\}$. If this is the case, then the* plan induced by σ *is the set $\{\mathsf{T}_x \mid x \in \mathsf{SV}\}$, where $\mathsf{T}_x = \langle (x, \hat{v}_{i_0}, i_1 - i_0), (x, \hat{v}_{i_1}, i_2 - i_1), \ldots, (x, \hat{v}_{i_{k-1}}, i_k - i_{k-1}) \rangle$ and $i_k = |\sigma|$, for all $x \in \mathsf{SV}$.*

Intuitively, if a word weakly-encodes a plan, then it captures the dynamics of a state variable modulo its domain and its transition function, which will be taken care of in the next section. A converse correspondence from plans to words can be defined accordingly.

Before concluding the section, we introduce another notation that will come handy later. We denote by $\mathit{events}(\sigma)$ the set of events (beginning/ending of a token) occurring at a given time, encoded in the alphabet symbol σ. Formally, $\mathit{events}(\sigma)$ is the smallest set such that:

- if $\sigma(x) = (v, v')$ for some x, then $\{\mathit{end}(x, v), \mathit{start}(x, v')\} \subseteq \mathit{events}(\sigma)$, and
- if $\sigma(x) = (-, v')$ for some x, then $\mathit{start}(x, v') \in \mathit{events}(\sigma)$.

5 DFA Accepting Plans

Given an eager qualitative timeline-based planning problem $P = (\mathsf{SV}, S)$, we show how to build a DFA $\mathscr{T}_{\mathsf{SV}}$, of size at most exponential in the size of P, accepting words that correctly encode plans over SV, that is, words that weakly-encode plans (*cf.* Definition 7) *and* comply with the constraints on the alternation of token values expressed by functions T_x, for $x \in \mathsf{SV}$. In the next section, we show how to obtain a DFA, of size at most exponential in the size of P, that accepts exactly the *solution plans* for P.

For every planning problem $P = (\mathsf{SV}, S)$, the DFA $\mathscr{T}_{\mathsf{SV}}$ is the tuple $\langle Q_{\mathsf{SV}}, \Sigma_{\mathsf{SV}}, \delta_{\mathsf{SV}}, q^0_{\mathsf{SV}}, F_{\mathsf{SV}} \rangle$, whose components are defined as follows.

- Q_{SV} is the set of *states* of $\mathscr{T}_{\mathsf{SV}}$. Intuitively, a state of $\mathscr{T}_{\mathsf{SV}}$ keeps track of the token values of the timelines at the current and the previous step of the run. Therefore, a state is a function mapping each state variable x into a pair (v, v'), where v' (resp., v) denotes the token value of timeline x at the current (resp., previous) step. To formally define Q_{SV}, we exploit the definition of alphabet Σ_{SV} from Section 4. Mostly, states are alphabet symbols, except for those functions $\sigma \in \Sigma_{\mathsf{SV}}$ assigning to at least one state variable $x \in \mathsf{SV}$ value $\circlearrowleft$. For technical reasons, we also need a fresh *initial state* q^0_{SV} and a fresh *rejecting sink state* $\mathbf{s}_{\mathsf{SV}}$.

  Formally, $Q_{\mathsf{SV}} = \big(\Sigma_{\mathsf{SV}} \setminus \overline{Q}_{\mathsf{SV}}\big) \cup \{q^0_{\mathsf{SV}}, \mathbf{s}_{\mathsf{SV}}\}$, where $\overline{Q}_{\mathsf{SV}} = \{\sigma \in \Sigma_{\mathsf{SV}} \mid \sigma(x) = \circlearrowleft \text{ for some } x \in \mathsf{SV}\}$. Clearly, the size of Q_{SV} is at most as the size of Σ_{SV}, which is in turn at most exponential in the size of P.

- Σ_{SV} is the *input alphabet*, defined as in Section 4.

- $\delta_{\mathsf{SV}} : Q_{\mathsf{SV}} \times \Sigma_{\mathsf{SV}} \to Q_{\mathsf{SV}}$ is the *transition function*. Towards a definition of δ_{SV}, we say that an alphabet symbol $\sigma \in \Sigma_{\mathsf{SV}}$ is *compatible* with a state $\sigma_1 \in Q_{\mathsf{SV}}$ (we use for states the same symbols as for the alphabet, i.e., $\sigma, \sigma_1, \sigma_2, \ldots$, to stress the fact that states are closely related to alphabet symbols) if one of the following holds: *(i)* $\sigma_1 = q^0_{\mathsf{SV}}$ is the initial state and $\sigma \in \Sigma^I_{\mathsf{SV}}$ is an initial symbol such that for each $x \in \mathsf{SV}$ it holds that $\sigma(x) = (-, v)$ with $v \in V_x$; *(ii)* $\sigma_1 = (v, v') \in \Sigma_{\mathsf{SV}} \setminus \overline{Q}_{\mathsf{SV}}$ and $\sigma \in \Sigma^N_{\mathsf{SV}}$ is a non-initial symbol such that for each $x \in \mathsf{SV}$ either $\sigma(x) = \circlearrowleft$ or $\sigma(x) = (v', v'')$ with $v'' \in T_x(v') \cap V_x$.

  Now, $\delta_{\mathsf{SV}} : Q_{\mathsf{SV}} \times \Sigma_{\mathsf{SV}} \to Q_{\mathsf{SV}}$ is defined as follows. For all $\sigma_1 \in Q_{\mathsf{SV}}$ and $\sigma \in \Sigma_{\mathsf{SV}}$, if σ is not compatible with σ_1 or σ_1 is the sink state (i.e., $\sigma_1 = \mathbf{s}_{\mathsf{SV}}$), then $\delta(\sigma_1, \sigma) = \mathbf{s}_{\mathsf{SV}}$; otherwise

  - if σ_1 is the initial state (i.e., $\sigma_1 = q^0_{\mathsf{SV}}$), then $\delta(\sigma_1, \sigma) = \sigma$; in other words, in this case the automaton transitions to the state represented by the input letter;
  - if $\sigma_1 \in \Sigma_{\mathsf{SV}} \setminus \overline{Q}_{\mathsf{SV}}$, then $\delta(\sigma_1, \sigma) = \sigma_2$, where $\sigma_2(x) = \sigma_1(x)$ if $\sigma(x) = \circlearrowleft$, and $\sigma_2(x) = \sigma(x)$ otherwise, for all $x \in \mathsf{SV}$; intuitively, the automaton transitions into a state keeping track of the updated information about which tokens have changed value and which ones have not.

  We point out that, in both cases, the automaton transitions to the next state in a deterministic fashion.

- $F_{\mathsf{SV}} = Q_{\mathsf{SV}} \setminus \{\mathbf{s}_{\mathsf{SV}}\}$ is the set of *final states*.

Correctness of the DFA $\mathscr{T}_{\mathsf{SV}}$ is proved by the next lemma.

Lemma 1. *Let $P = (\mathsf{SV}, S)$ be an eager qualitative timeline-based planning problem. Then, words accepted by $\mathscr{T}_{\mathsf{SV}}$ are exactly those encoding plans over SV. Moreover the size of $\mathscr{T}_{\mathsf{SV}}$ is at most exponential in the size of P.*

6 DFA Accepting Solution Plans

In this section, we go through the construction of an automaton recognizing solution plans for a planning problem. Towards that, it will come in handy to define some auxiliary structures, namely *blueprints*, *snapshots* and *viewpoints*; moreover, we will define how these structures evolve and give a high-level intuition for each of them.

Let $P = (\mathsf{SV}, S)$ be an eager qualitative timeline-based planning problem, and let $V = \cup_{x \in \mathsf{SV}} V_x$. We first show how to build a DFA $\mathscr{A}_P$, whose size is at most exponential in the size of P, that accepts exactly those words encoding solutions plans for P when restricted to words encoding plans over SV. In different terms, if a word encodes a plan over SV, then it is accepted by $\mathscr{A}_P$ if and only if it encodes a solution plan for P. However, $\mathscr{A}_P$ may also accept words that do not encode a plan over SV. Therefore, we need the intersection of such a DFA $\mathscr{A}_P$ with DFA $\mathscr{T}_{\mathsf{SV}}$ from the previous section.

In the following, we use *preorders* to represent the ordering relation imposed by synchronization rules. Each existential statement of the form $\exists a_1[x_1 = v_1]a_2[x_2 = v_2] \ldots a_n[x_n = v_n].\mathscr{C}$, with $\mathscr{C}$ conjunction of atoms, identifies a *preorder* whose domain is the set of terms $\mathrm{start}(a)/\mathrm{end}(a)$ occurring in $\mathscr{C}$, and where term t_1 precedes term t_2 in the preorder whenever $t_1 \leq t_2$ belongs to $\hat{\mathscr{C}}$.

For a preorder $\mathscr{P}$, we denote by $dom(\mathscr{P})$ its domain and by $\preceq_{\mathscr{P}}$ the ordering relation. Moreover, we use $x \equiv_{\mathscr{P}} y$ to denote the fact that both $x \preceq_{\mathscr{P}} y$ and $y \preceq_{\mathscr{P}} x$ hold, and $x \prec_{\mathscr{P}} y$ to denote the fact that $x \preceq_{\mathscr{P}} y$ holds but $y \preceq_{\mathscr{P}} x$ does not. Finally, we denote by $[x]_{\equiv_{\mathscr{P}}}$ the equivalence class of x with respect to $\equiv_{\mathscr{P}}$ for every $x \in dom(\mathscr{P})$, that is, $[x]_{\equiv_{\mathscr{P}}} = \{y \in dom(\mathscr{P}) \mid y \equiv_{\mathscr{P}} x\}$. We omit the subscript $_{\mathscr{P}}$ when it is clear from the context. A preorder $\mathscr{P}$ induces a directed acyclic graph (DAG) $G = (V, A)$, where V is the set of equivalence classes, that is, $V = \{[x]_{\equiv} \mid x \in dom(\mathscr{P})\}$, and, for every $x, y \in dom(\mathscr{P})$ there is an arc

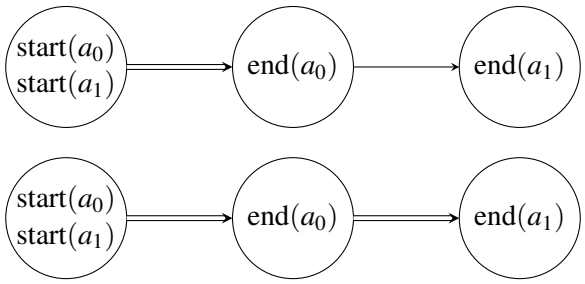

Figure 1: Above, we show the blueprint for the unique existential statement in the rule $a_0[x_0 = v_0] \rightarrow \exists a_1[x_1 = v_1].(\text{start}(a_0) = \text{start}(a_1) \wedge \text{end}(a_0) \leq \text{end}(a_1))$, from Section 3. It forces token a_0 to either be a prefix of or coincide with token a_1. Below, the blueprint obtained replacing $\text{end}(a_0) \leq \text{end}(a_1)$ with $\text{end}(a_0) < \text{end}(a_1)$, that forces a_1 to be a strict prefix of a_1.

from $[x]_{\equiv}$ to $[y]_{\equiv}$ in A (denoted by $([x]_{\equiv}, [y]_{\equiv}) \in A$ or $[x]_{\equiv} \rightarrow [y]_{\equiv}$ when set A is clear from the context) if and only if $x \prec y$ and there is no $w \in dom(\mathscr{P})$ such that $x \prec w$ and $w \prec y$. Clearly, there is a path from $[x]_{\equiv}$ to $[y]_{\equiv}$ (denoted by $[x]_{\equiv} \rightarrow^* [y]_{\equiv}$) if and only if $x \preceq y$. Therefore, given an existential statement $\mathscr{E}$ occurring in a synchronization rule $\mathscr{R}$, we refer to the associated preorder and DAG as, respectively, $\mathscr{P}_{\mathscr{E}}$ and $G_{\mathscr{E}}$.

It is important to observe that a conjunction of atoms $\mathscr{C}$ within an existential statement $\mathscr{E}$ contains atoms of both forms $t_1 \leq t_2$ and $t_1 < t_2$. To keep track of these different constraints in DAG $G_{\mathscr{E}} = (V,A)$ associated with $\mathscr{E}$, we identify the subset $A_< \subseteq A$ of arcs of $G_{\mathscr{E}}$ as the set $A_< = \{([x]_{\equiv}, [y]_{\equiv}) \in A \mid x < y \in \hat{\mathscr{C}}\}$. We sometimes write $[x]_{\equiv} \Rightarrow [y]_{\equiv}$ for $([x]_{\equiv}, [y]_{\equiv}) \in A_<$, when A is clear from the context. Figure 1 shows such a difference.

Let $\mathscr{E}$ be an existential statement occurring in a rule $\mathscr{R}$ and $G_{\mathscr{E}}$ the DAG associated with $\mathscr{E}$. The set of *events associated with a vertex* $[x]_{\equiv}$ of $G_{\mathscr{E}}$, denoted by $events_{G_{\mathscr{E}}}([x]_{\equiv})$, is the smallest set such that if $\text{start}(a) \in [x]_{\equiv}$ (resp., $\text{end}(a) \in [x]_{\equiv}$) and $a[y = v]$ either occurs in $\mathscr{E}$ or is the trigger of $\mathscr{R}$, then $start(y,v) \in events_{G_{\mathscr{E}}}([x]_{\equiv})$ (resp., $end(y,v) \in events_{G_{\mathscr{E}}}([x]_{\equiv})$). The set of *events associated with a subset* V' *of vertices* of $G_{\mathscr{E}}$, denoted by $events_{G_{\mathscr{E}}}(V')$, is the set $\bigcup_{v \in V'} events_{G_{\mathscr{E}}}(v)$.

6.1 Blueprints, Snapshots, and Viewpoints

A DAG associated with an existential statement $\mathscr{E}$ is also called a *blueprint* for $\mathscr{E}$. A *snapshot* for an existential statement $\mathscr{E}$ is a pair (G,K), where $G = (V,A)$ is a blueprint for $\mathscr{E}$ and $K \subseteq V$ is a *downward closed* subset of vertices of G, that is, $v \in K$ implies $v' \in K$ for all $v' \in V$ with $v' \rightarrow^* v$. The number of different snapshots for $\mathscr{E}$ is at most $2^{|V|}$, hence at most exponential in the size of P, denoted by $|P|$. A *viewpoint* $\mathbb{V}$ for a rule $\mathscr{R}$ is a set of snapshots for existential statements in $\mathscr{R}$, at most one for each statement. Let $n_{\mathscr{R}}$ be the number of existential statements in $\mathscr{R}$; then, it is easy to see that the number of different viewpoints for $\mathscr{R}$ is at most $(2^{|P|})^{n_{\mathscr{R}}}$, hence exponential in the size of P. If $K = \varnothing$ for all $(G,K) \in \mathbb{V}$, then $\mathbb{V}$ is the *initial* viewpoint of $\mathscr{R}$; analogously, if K is the entire set of vertices of G, for some $(G,K) \in \mathbb{V}$, then $\mathbb{V}$ is a *final* viewpoint of $\mathscr{R}$.

Intuitively, a viewpoint checks the satisfaction of a rule $\mathscr{R}$ by recognizing when at least one existential statement has been fulfilled. This check works by collecting, for each existential statement, information about the tokens seen so far along the plan into snapshots, which are downward closed and accurately represent all relevant symbols read. How information is collected, thus how viewpoints and snapshots evolve, is explained in the following.

States of automata $\mathscr{A}_P$ are sets of viewpoints containing at least one viewpoint for each rule of P (besides a fresh rejecting sink state $\mathbf{s}_P$); recall that viewpoints are in turn sets of snapshots. Therefore, to define automata runs, we first show how snapshots and viewpoints evolve upon reading an alphabet symbol. To this end, we need the following notions.

For a snapshot (G,K), we set $next(G,K) = K'$, where K' is the largest downward closed subset of vertices of G for which there is no pair of vertices $v, v' \in K' \setminus K$ with $v \Rightarrow v'$. Moreover, given an alphabet symbol $\sigma \in \Sigma_{\mathsf{SV}}$, we define $next((G,K),\sigma) = K'$, where K' is the largest downward closed subset of vertices of $next(G,K)$ such that $events_G(K' \setminus K) \subseteq events(\sigma)$. We say that snapshot (G,K) is *compatible* with symbol σ if for all $start(x,v) \in events_G(K)$ and $end(x,v) \in events(\sigma) \cap events_G(V \setminus K)$, it holds that $end(x,v) \in events_G(next((G,K),\sigma))$.

Intuitively, during a run of the automaton, a snapshot (G,K) evolves by suitably extending K. $next(G,K)$ identifies the only vertices that can appear in such an extension independently from the alphabet symbol read, that is, vertices in $V \setminus K$ reachable (from K) without crossing arcs in $A_<$. The exact extension, however, depends on the actual symbol σ read by the automaton: K cannot be extended with events that are not included in σ. Therefore, $next((G,K),\sigma)$ identifies precisely how a snapshot evolves. At last, observe that for a snapshot to be allowed to evolve upon reading a symbol, it must be guaranteed that no token ending is overlooked, which is formalized by the notion of compatibility of a snapshot with a symbol.

We can now characterize the evolution of snapshots and viewpoints when reading an alphabet symbol $\sigma \in \Sigma_{\mathsf{SV}}$. The *evolution of a snapshot* (G,K) when reading σ, denoted $evol((G,K),\sigma)$, is snapshot $(G, next((G,K),\sigma))$, if (G,K) is compatible with σ; it is undefined otherwise. The *evolution of a viewpoint* $\mathbb{V}$ when reading σ, denoted $evol(\mathbb{V},\sigma)$, is viewpoint $\mathbb{V}'$, defined as the smallest set such that for all $(G,K) \in \mathbb{V}$, if $evol((G,K),\sigma)$ is defined, then $evol((G,K),\sigma) \in \mathbb{V}'$.

6.2 States, Initial State, and Final States of $\mathscr{A}_P$

We have already mentioned that *states* of $\mathscr{A}_P$ are sets of viewpoints containing at least one viewpoint for each rule $\mathscr{R} \in S$ (recall that S is the set of rules in planning problem P), besides a fresh rejecting sink state $\mathbf{s}_P$. However, since it is crucial for us to bound the size of $\mathscr{A}_P$ to be at most exponential in the one of P, we impose the *linearity condition*, formalized in what follows.

First, recall that, given a rule $\mathscr{R}$, featuring existential statements $\mathscr{E}_1, \ldots, \mathscr{E}_{n_\mathscr{R}}$, a viewpoint $\mathbb{V}$ for $\mathscr{R}$ only contains at most one snapshot for each existential statement in $\mathscr{R}$; therefore, it holds that $|\mathbb{V}| \leq n_\mathscr{R}$ and there is a partial surjective function $f_\mathbb{V} : \{\mathscr{E}_1, \ldots, \mathscr{E}_{n_\mathscr{R}}\} \to \mathbb{V}$, where $f_\mathbb{V}(\mathscr{E})$ is the only snapshot for $\mathscr{E}$ in $\mathbb{V}$, if any, for all $\mathscr{E} \in \{\mathscr{E}_1, \ldots, \mathscr{E}_{n_\mathscr{R}}\}$.

Now, for all rules $\mathscr{R} \in S$, let $\Upsilon_\mathscr{R}$ be the set of viewpoints for $\mathscr{R}$, and let $\Upsilon_P = \bigcup_{\mathscr{R} \in S} \Upsilon_\mathscr{R}$. We define an ordering relation $\preceq$ between viewpoints: for all $\mathbb{V}, \mathbb{V}' \in \Upsilon_P$, it holds that $\mathbb{V} \preceq \mathbb{V}'$ if and only if *(i)* $\mathbb{V}, \mathbb{V}' \in \Upsilon_\mathscr{R}$ for some $\mathscr{R} \in S$, *(ii)* $dom(f_{\mathbb{V}'}) \subseteq dom(f_\mathbb{V})$,[6] and *(iii)* for all $\mathscr{E} \in dom(f_{\mathbb{V}'})$, we have that $f_\mathbb{V}(\mathscr{E}) = (G,K)$, $f_{\mathbb{V}'}(\mathscr{E}) = (G,K')$, and $K \subseteq K'$. Intuitively, $\mathbb{V} \preceq \mathbb{V}'$ captures the fact that $\mathbb{V}'$ has gone further than $\mathbb{V}$ in matching input symbols to satisfy a rule. Therefore, a snapshot in $\mathbb{V}$ either evolved into one in $\mathbb{V}'$, according to the symbols read, or has disappeared because it is not compatible with some of the symbols read, and thus it cannot be used anymore to satisfy the rule.

At this point, we can formalize the linearity condition, crucial to constrain the size of $\mathscr{A}_P$ (Lemma 2).

Definition 8 (Linearity condition)**.** *A set of viewpoints Υ satisfies the* linearity condition *if for all viewpoints $\mathbb{V}, \mathbb{V}' \in \Upsilon$ and rules $\mathscr{R} \in S$, if $\mathbb{V}, \mathbb{V}' \in \Upsilon_\mathscr{R}$, then $\mathbb{V} \preceq \mathbb{V}'$ or $\mathbb{V}' \preceq \mathbb{V}$ holds.*

[6] For a partial function f, we denote by $dom(f)$ the set of elements where f is defined.

Intuitively, we impose all viewpoints for the same rule in a state of $\mathscr{A}_P$ to be linearly ordered.

We are now ready to formally characterize the set of *states* of $\mathscr{A}_P$, consisting of the sets $\Upsilon \subseteq \Upsilon_P$ of viewpoints that contain at least one viewpoint for each rule $\mathscr{R} \in S$ and that satisfy the linearity condition, and including, in addition, a fresh *rejecting sink state* $\mathbf{s}_P$. We denote it by Q_P.

The *initial state* q_P^0 of $\mathscr{A}_P$ is the set $\{\mathbb{V}^0_{\mathscr{R}} \mid \mathscr{R} \in S\}$, where $\mathbb{V}^0_{\mathscr{R}}$ is the initial viewpoint of rule $\mathscr{R}$.

Towards a definition of the set F_P of *final states* of $\mathscr{A}_P$, we introduce the notion of *enabled viewpoints*. A viewpoint $\mathbb{V}$ for rule $\mathscr{R} \in S$ is *enabled* if either $\mathscr{R}$ is triggerless or $\mathscr{R}$ has trigger token a_0 and $\text{start}(a_0) \in K$ for some $(G,K) \in \mathbb{V}$. A state q of $\mathscr{A}_P$ is *final* if every enabled viewpoint therein is final.

6.3 Transition Function of $\mathscr{A}_P$

The last step of our construction is the definition of the *transition function* δ_P for automaton $\mathscr{A}_P$.

To this end, we introduce the notion of alphabet symbol *enabling* a viewpoint $\mathbb{V}$ along with the one of states of $\mathscr{A}_P$ *compatible* with an alphabet symbol. Let $\mathbb{V}$ be a viewpoint for a non-triggerless rule $\mathscr{R}$ with trigger token a_0 and $\sigma \in \Sigma_{\mathsf{SV}}$ an alphabet symbol. We say that σ *enables* $\mathbb{V}$ if there is $(G,K) \in \mathbb{V}$ with $\text{start}(a_0) \in \mathit{next}((G,K),\sigma)$. Moreover, we say that a state $q \in Q_P \setminus \{\mathbf{s}_P\}$ is *compatible* with σ if for all non-triggerless rules $\mathscr{R} \in S$, with trigger token $a_0[x_0 = v_0]$, if $\mathit{start}(x_0, v_0) \in \mathit{events}(\sigma)$, then there is a viewpoint $\mathbb{V} \in q$ such that σ enables $\mathbb{V}$.

We are now ready to define the *transition function* δ_P of $\mathscr{A}_P$. For all $q \in Q_P$ and alphabet symbol $\sigma \in \Sigma_{\mathsf{SV}}$:

- if $q = \mathbf{s}_P$ or q is not compatible with σ, then $\delta(q,\sigma) = \mathbf{s}_P$;
- otherwise, $\delta(q,\sigma) = q'$, where q' is the smallest set such that for all $\mathbb{V} \in q$
  - $\mathit{evol}(\mathbb{V},\sigma) \in q'$ and
  - if σ enables $\mathbb{V}$, then $\mathbb{V} \in q'$.

Lemma 2. *Let $P = (\mathsf{SV}, S)$ be an eager qualitative timeline-based planning problem. Then, each finite word over Σ_{SV} that encodes a plan over SV is accepted by $\mathscr{A}_P$ if and only if it encodes a solution plan for P. Moreover, the size of $\mathscr{A}_P$ is at most exponential in the size of P.*

Proof. For lack of space, we omit the proof of soundness showing that the automaton accepts the correct language as claimed. Instead, we show that the size of $\mathscr{A}_P$ is indeed at most exponential in the size of P.

Let k be the largest number of existential statements in a rule of P and k' the largest number of atoms in an existential statement of P. Thanks to the linearity rule enjoyed by states of P, it is not difficult to convince oneself that the number of different viewpoints for the same rule in a state $q \in Q_P$ to be at most $k \times k'$. Thus, each state in Q_P contains at most $|S| \times k \times k'$ different viewpoints (the product of the number of rules in P by the number of different viewpoints for the same rule).

Therefore, the size of Q_P is at most $|\Upsilon_P|^{(|S| \times k \times k')}$. Clearly, $(|S| \times k \times k')$ is at most polynomial in the size of P. Since $|\Upsilon_P| \leq \sum_{\mathscr{R} \in S} |\Upsilon_{\mathscr{R}}|$ and, as already pointed out, $|\Upsilon_{\mathscr{R}}|$ is at most exponential in the size of P, we can conclude that the size of Q_P is at most exponential in the size of P. □

Theorem 1. *Let $P = (\mathsf{SV}, S)$ be an eager qualitative timeline-based planning problem. Then, the words accepted by the intersection automaton of $\mathscr{A}_P$ and $\mathscr{T}_{\mathsf{SV}}$ are exactly those encoding solution plans for P. Moreover, the size of the intersection automaton of $\mathscr{A}_P$ and $\mathscr{T}_{\mathsf{SV}}$ is at most* exponential *in the size of P.*

7 A Maximal Subset of Allen's Relations

Allen's interval algebra is a formalism for temporal reasoning introduced in [2]. It identifies all possible relations between pairs of time intervals over a linear order and specifies a machinery to reason about them. In this section, we isolate the maximal subset of Allen's relations captured by the eager fragment of qualitative timeline-based planning. To this end, we show how to map Allen's relations over tokens in terms of their endpoints, that is, as conjunctions of atoms over terms $\text{start}(a), \text{start}(b), \text{end}(a), \text{end}(b)$, for token names a and b. Then, we check which relation encoding satisfies the conditions of Definition 6. Let $a, b \in \mathcal{N}$.

- a before b (b after a) can be defined as $\text{end}(a) < \text{start}(b)$.
- a meets b (b is-met-by a) can be defined as $\text{end}(a) = \text{start}(b)$.
- a ends b (b is-ended-by a) can be defined as $\text{start}(b) < \text{start}(a) \wedge \text{end}(a) = \text{end}(b)$.
- a starts b (b is-started-by a) can be defined as $\text{start}(a) = \text{start}(b) \wedge \text{end}(a) < \text{end}(b)$.
- a overlaps b (b is-overlapped-by a) can be defined as $\text{start}(a) < \text{start}(b) \wedge \text{start}(b) < \text{end}(a) \wedge \text{end}(a) < \text{end}(b)$.
- a during b (b contains a) can be defined as $\text{start}(b) < \text{start}(a) \wedge \text{end}(a) < \text{end}(b)$.
- $a = b$ can be defined as $\text{start}(a) = \text{start}(b) \wedge \text{end}(a) = \text{end}(b)$.

It is not difficult to see that, if one of the tokens, let's say a, is the trigger token, then the encodings not complying with Definition 6 are the ones for Allen's relations ends, is-ended-by, overlaps, is-overlapped-by, and during. Thus, the maximal subset of Allen's relations that can be captured by an instance of the eager fragment of the timeline-based planning problem consists of relations before, after, meets, is-met-by, starts, is-started-by, contains, and $=$.

As an example, consider relation overlaps and let $\mathscr{C} = \{\text{start}(a) < \text{start}(b), \text{start}(b) < \text{end}(a), \text{end}(a) < \text{end}(b)\}$ be its encoding. Clearly, the transitive closure $\hat{\mathscr{C}}$ of $\mathscr{C}$ (cf. Section 3) includes also $\text{start}(a) \leq \text{start}(b)$ and $\text{end}(a) \leq \text{end}(b)$ but it does not include $\text{start}(b) \leq \text{start}(a)$, thus violating Condition 2 of Definition 6. A similar argument can be used for relations ends, is-ended-by, is-overlapped-by, and during.

If, instead, none of the token is a trigger token, then the only Allen's relations not violating any of the conditions of Definition 6 are before, after, meets, and is-met-by. We omit the details.

8 Conclusions

In this paper, we identified a meaningful fragment of timeline-based planning whose solutions can be recognized by DFAs of singly exponential size. Specifically, we identified restrictions on the allowed synchronization rules, which we called *eager rules*, for which we showed how to build the corresponding deterministic automaton of exponential size, that can then be directly exploited to synthesize strategies. Moreover, we isolated a maximal subset of Allen's relations captured by such a fragment.

Whether the fragment of timeline-based planning identified by the eager rules is maximal or not is an open question currently under study. Further research directions include a parametrized complexity analysis over the number of synchronization rules and a characterization in terms of temporal logics, like the one in [15].

References

[1] Renato Acampora, Luca Geatti, Nicola Gigante, Angelo Montanari & Valentino Picotti (2022): *Controller Synthesis for Timeline-based Games*. In Pierre Ganty & Dario Della Monica, editors: *Proceedings of the 13th International Symposium on Games, Automata, Logics and Formal Verification, GandALF 2022, Madrid, Spain, September 21-23, 2022*, *EPTCS* 370, pp. 131–146, doi:10.4204/EPTCS.370.9.

[2] James F. Allen (1983): *Maintaining Knowledge about Temporal Intervals*. *Commun. ACM* 26(11), pp. 832–843, doi:10.1145/182.358434.

[3] Laura Bozzelli, Alberto Molinari, Angelo Montanari & Adriano Peron (2018): *Complexity of Timeline-Based Planning over Dense Temporal Domains: Exploring the Middle Ground*. In Andrea Orlandini & Martin Zimmermann, editors: *Proceedings of the 9th International Symposium on Games, Automata, Logics, and Formal Verification*, *EPTCS* 277, pp. 191–205, doi:10.4204/EPTCS.277.14.

[4] Laura Bozzelli, Alberto Molinari, Angelo Montanari & Adriano Peron (2018): *Decidability and Complexity of Timeline-Based Planning over Dense Temporal Domains*. In Michael Thielscher, Francesca Toni & Frank Wolter, editors: *Principles of Knowledge Representation and Reasoning: Proceedings of the Sixteenth International Conference, KR 2018, Tempe, Arizona, 30 October - 2 November 2018*, AAAI Press, pp. 627–628. Available at https://aaai.org/ocs/index.php/KR/KR18/paper/view/17995.

[5] Steve Chien, Gregg Rabideau, Russell Knight, Robert Sherwood, Barbara Engelhardt, Darren Mutz, Tara Estlin, Benjamin Smith, Forest Fisher, T Barrett et al. (2000): *ASPEN-Automating space mission operations using automated planning and scheduling*. In: *SpaceOps 2000*, AIAA Press.

[6] Steve A. Chien, Rob Sherwood, Daniel Tran, Benjamin Cichy, Gregg Rabideau, Rebecca Castaño, Ashley Davies, Rachel Lee, Dan Mandl, Stuart Frye, Bruce Trout, Jerry Hengemihle, Jeff D'Agostino, Seth Shulman, Stephen G. Ungar, Thomas Brakke, Darrell Boyer, Jim Van Gaasbeck, Ronald Greeley, Thomas Doggett, Victor R. Baker, James M. Dohm & Felipe Ip (2004): *The EO-1 Autonomous Science Agent*. In: *3rd International Joint Conference on Autonomous Agents and Multiagent Systems*, IEEE Computer Society, pp. 420–427, doi:10.1109/AAMAS.2004.10022.

[7] Marta Cialdea Mayer, Andrea Orlandini & Alessandro Umbrico (2016): *Planning and execution with flexible timelines: a formal account*. *Acta Informatica* 53(6-8), pp. 649–680, doi:10.1007/s00236-015-0252-z.

[8] Dario Della Monica, Nicola Gigante, Salvatore La Torre & Angelo Montanari (2020): *Complexity of Qualitative Timeline-Based Planning*. In Emilio Muñoz-Velasco, Ana Ozaki & Martin Theobald, editors: *27th International Symposium on Temporal Representation and Reasoning, TIME 2020, September 23-25, 2020, Bozen-Bolzano, Italy*, *LIPIcs* 178, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, pp. 16:1–16:13, doi:10.4230/LIPICS.TIME.2020.16.

[9] Maria Fox & Derek Long (2003): *PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains*. *J. Artif. Intell. Res.* 20, pp. 61–124, doi:10.1613/jair.1129.

[10] Simone Fratini, Amedeo Cesta, Andrea Orlandini, Riccardo Rasconi & Riccardo De Benedictis (2011): *APSI-based Deliberation in Goal Oriented Autonomous Controllers*. In: *ASTRA 2011*, 11, ESA.

[11] Nicola Gigante, Angelo Montanari, Marta Cialdea Mayer & Andrea Orlandini (2016): *Timelines Are Expressive Enough to Capture Action-Based Temporal Planning*. In Curtis E. Dyreson, Michael R. Hansen & Luke Hunsberger, editors: *23rd International Symposium on Temporal Representation and Reasoning,*, IEEE Computer Society, pp. 100–109, doi:10.1109/TIME.2016.18.

[12] Nicola Gigante, Angelo Montanari, Marta Cialdea Mayer & Andrea Orlandini (2017): *Complexity of Timeline-Based Planning*. In Laura Barbulescu, Jeremy Frank, Mausam & Stephen F. Smith, editors: *Proceedings of the Twenty-Seventh International Conference on Automated Planning and Scheduling, ICAPS 2017, Pittsburgh, Pennsylvania, USA, June 18-23, 2017*, AAAI Press, pp. 116–124. Available at https://aaai.org/ocs/index.php/ICAPS/ICAPS17/paper/view/15758.

[13] Nicola Gigante, Angelo Montanari, Andrea Orlandini, Marta Cialdea Mayer & Mark Reynolds (2020): *On timeline-based games and their complexity*. *Theoretical Computer Science* 815, pp. 247–269, doi:10.1016/j.tcs.2020.02.011.

[14] Dario Della Monica, Nicola Gigante, Angelo Montanari & Pietro Sala (2018): *A Novel Automata-Theoretic Approach to Timeline-Based Planning*. In Michael Thielscher, Francesca Toni & Frank Wolter, editors: *Principles of Knowledge Representation and Reasoning: Proceedings of the Sixteenth International Conference, KR 2018, Tempe, Arizona, 30 October - 2 November 2018*, AAAI Press, pp. 541–550. Available at https://aaai.org/ocs/index.php/KR/KR18/paper/view/18024.

[15] Dario Della Monica, Nicola Gigante, Angelo Montanari, Pietro Sala & Guido Sciavicco (2017): *Bounded Timed Propositional Temporal Logic with Past Captures Timeline-based Planning with Bounded Constraints*. In Carles Sierra, editor: *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, ijcai.org, pp. 1008–1014, doi:10.24963/IJCAI.2017/140.

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

[17] Amir Pnueli & Roni Rosner (1989): *On the Synthesis of an Asynchronous Reactive Module*. In Giorgio Ausiello, Mariangiola Dezani-Ciancaglini & Simona Ronchi Della Rocca, editors: *16th International Colloquium on Automata, Languages and Programming*, *Lecture Notes in Computer Science* 372, Springer, pp. 652–671, doi:10.1007/BFB0035790.

[18] Alessandro Umbrico, Amedeo Cesta, Marta Cialdea Mayer & Andrea Orlandini (2017): *PLATINUm: A New Framework for Planning and Acting*. In Floriana Esposito, Roberto Basili, Stefano Ferilli & Francesca A. Lisi, editors: *Proceedings of the 16th International Conference of the Italian Association for Artificial Intelligence*, *LNCS* 10640, Springer, pp. 498–512, doi:10.1007/978-3-319-70169-1_37.

[19] Alessandro Umbrico, Amedeo Cesta & Andrea Orlandini (2023): *Human-Aware Goal-Oriented Autonomy through ROS-Integrated Timeline-based Planning and Execution*. In: *32nd IEEE International Conference on Robot and Human Interactive Communication*, IEEE, pp. 1164–1169, doi:10.1109/RO-MAN57019.2023.10309516.