A Formal Account of Planning with Flexible Timelines

Marta Cialdea Mayer and Alessandro Umbrico
Dipartimento di Ingegneria
Università degli Studi Roma TRE
Via della Vasca Navale 79, 00144, Roma (Italia)
Email: cialdea@dia.uniroma3.it, alessandro.umbrico@uniroma3.it

Andrea Orlandini
Istituto di Scienze e Tecnologie della Cognizione
Consiglio Nazionale delle Ricerche
Via S. Martino della Battaglia 44, 00185, Roma (Italia)
Email: andrea.orlandini@istc.cnr.it

Abstract—Planning for real world problems with explicit temporal constraints is a challenging problem. Among several approaches, the use of flexible timelines in Planning and Scheduling (P&S) has demonstrated to be successful in a number of concrete applications, such as, for instance, autonomous space systems. A flexible timeline describes an envelope of possible solutions which can be exploited by an executive system for robust on-line execution. A remarkable research effort has been dedicated to design, build and deploy software environments, like EUROPA, ASPEN, and APSI-TRF, for the synthesis of timeline-based P&S applications. Several attempts have also been made to characterize the concept of timelines. Nevertheless, a formal characterization of flexible timelines and plans is still missing. This paper presents a formal account of flexible timelines aiming at providing a general semantics for related planning concepts such as domains, goals, problems, constraints and flexible plans. Some basic properties of the defined concepts are also stated and proved. A simple running example inspired by a real world planning domain is exploited to illustrate the proposed formal notions. Finally, a planning tool, called Extensible Planning and Scheduling Library (EPSL), is briefly presented, which is able to generate flexible plans that are compliant with the given semantics.

I. INTRODUCTION

Planning for real world problems with explicit temporal constraints is a challenging problem [1]. Among several approaches, the use of flexible timelines in Planning and Scheduling (P&S) has demonstrated to be successful in a number of concrete applications, such as, for instance, autonomous space systems [2], [3], [4]. Timeline-based planning has been introduced in [2] under a modeling assumption inspired by classical control theory. A planning problem is modeled by identifying a set of relevant components whose temporal evolution has to be controlled to obtain a desired behavior. Components represent logical or physical subsystems whose properties may vary over time. In this respect, the set of domain features under control are modeled as a set of temporal functions whose values have to be decided over a temporal horizon. Such functions are synthesized during problem solving by posting planning decisions. The evolution of a single temporal feature over a temporal horizon is called the timeline of that feature. In general, plans synthesized by temporal P&S systems may be temporally flexible. They are made up of flexible timelines, describing transition events that are associated with temporal intervals (with given lower and upper bounds), instead of exact temporal occurrences. In other words, a flexible plan describes an envelope of possible solutions aimed at facing uncertainty during actual execution. In this regard, they can be exploited by an executive system for robust on-line execution.

As a matter of fact, many P&S architectures return an envelope of potential solutions in form of a flexible plan which is commonly accepted to be less brittle of a single plan when coping with execution. A remarkable research effort has been dedicated to design, build and deploy software environments, like EUROPA [5], ASPEN [6], and APSI-TRF [7] for the synthesis of timeline-based P&S applications. Several attempts have also been made to characterize the concept of timelines. In [8], a domain description language (i.e., DDL) able to represent physical domains to solve P&S problems is described. For the DDL language a syntactic specification and a model theoretic characterization are given. In [9], a constraint-based attribute and interval planning paradigm for representing and reasoning about temporal plans is presented. A characterization of the paradigm, based on temporal intervals and attributes is given in order to enable the description of planning domains with time, resources, concurrent activities, etc. In [10], a basic timeline representation is described to represent a set of states, resources, timing, and transition constraints aiming at generalizing previous efforts provided in a number of P&S systems designed for space applications. The work in [11] aims at defining what is a timeline under a knowledge engineering point of view, i.e., discussing it as a data structure with associated services that a planning system may utilize in order to solve problems. Finally, a comprehensive and semantically well founded framework is provided in [12], that includes also temporal uncertainty in timeline-based approaches. Additionally, some aspects of flexible plan execution have been studied by working on the temporal network which underlies the constraint based plan representation often used by such systems – see for example [13], [14]. Nevertheless, to the best of our knowledge, a formal semantics of flexible timelines and plans is still missing.

We are currently addressing the general problem of formalizing flexible temporal plans and putting related execution properties on a firmer ground. In this regard, this paper presents the initial result of such a research effort, i.e., a formal account of flexible timelines aiming at providing a general semantics for related planning concepts such as domains, goals, problems, constraints and flexible plans. The proposed approach shares some similarities with [15], where however temporal networks are exploited in order to represent constraints.

The paper is organized as follows. A simple running
example inspired by a real world planning domain, described in Section II, is exploited to illustrate the proposed formal notions. Section III provides general concepts and definitions for timeline-based planning, while timelines and flexible timelines are treated, respectively, in Section IV and Section V. Some basic properties of the defined concepts are also stated and proved. As already remarked, this work represents only the first step in the direction of investigating the formal connection of flexible timeline-based plans with robust plan execution. In fact, the significance of flexible timelines is due to the fact that sometimes the duration of activities cannot be foreseen exactly. Therefore, they are naturally related to controllability issues, that are however not addressed here. In this respect, the present work is orthogonal to [12], where uncertainty is taken into account without explicitly formalizing the notions of flexible timelines and plans. Finally, a planning tool, called Extensible Planning and Scheduling Library (EPSL), is briefly presented in Section VI. EPSL is able to generate flexible plans that are compliant with the given semantics. A more detailed description of the system can be found in [16].

II. CASE STUDY: A SATELLITE PLANNING DOMAIN

In this section, a case study is presented as a running example throughout the whole paper to support the formal definitions given below. The domain is inspired by a space mission long term planning problem as described in [17]. The mission consists of a remote satellite operating around a target planet. The satellite can either point to the planet and use its instruments to produce scientific data or point towards a communication station (e.g., an Earth ground station) and communicate previously produced data. The satellite is controlled by a planner and an executive system to accomplish required tasks, i.e., scientific observations, communication, and maintenance activities. A set of operative constraints are to be satisfied: the satellite has to point towards the planet, thus allowing observations of the planet surface (Science operations); the satellite has to point to Earth for transmitting data and, also, communication with Earth must occur within a ground-station availability window; finally, some maintenance operations are also to be planned.

An example of mission for such a domain may be constituted by an action sequence in which the satellite is pointing to Earth and starts slewing towards the target planet. When a scientific target is locked, the satellite starts making some scientific operations (e.g., taking pictures with an infrared camera). Once such operations are completed, the satellite slews back to Earth in order to communicate the collected data. When the Earth is locked and the ground station is available, the satellite is finally able to transfer scientific results. Moreover, the satellite may have to perform some maintenance operations.

III. PLANNING DOMAINS AND GOALS

The timeline-based approach to planning pursues a general idea that planning and scheduling (P&S) for controlling complex physical systems consist in the synthesis of desired temporal behaviors (or timelines). According to this paradigm a domain is modeled as a set of features with an associated set of temporal functions on a finite set of values. The time varying features are called multi-valued state variables [2]. Like in classical control theory, the evolution of the features is described by some causal laws and limited by domain constraints. These are specified in a domain specification. The task of a planner is to find a sequence of decisions that brings the timelines into a final desired set always satisfying the domain specification and special conditions called goals. Causal and temporal constraints specify which value transitions are allowed, the duration of each valued interval and (so-called) synchronization constraints between different state variables. This section is devoted to properly define all these notions.

For the sake of generality, temporal instants and durations are taken from an infinite set of non negative numbers \( \mathbb{T} \), including 0. For instance, \( \mathbb{T} \) can be the set of natural numbers \( \mathbb{N} \) (in a discrete time framework), as well as the non-negative real numbers \( \mathbb{R}_{\geq 0} \). Sometimes, \( \infty \) is given as an upper bound to allowed numeric values, with the meaning that \( \forall t < \infty \) for every \( t \in \mathbb{T} \). The notation \( \mathbb{T}^\infty \) will be used to denote \( \mathbb{T} \cup \{ \infty \} \).

A state variable is characterized by four components: its name, the set \( V \) of values the variable may assume, a function \( T \) associating to each value in \( v \in V \) the set of values that are allowed to follow \( v \), and a function \( D \) which may set upper and lower bounds on the duration of each variable value.

**Definition 1.** A state variable is a tuple \((x, V, T, D)\), where \( x \) is a unique identifier, called the variable name; \( V \) is a non-empty set, called the set of the state variable values; \( T : V \rightarrow 2^V \) is a total function, called the state variable value transition function; \( D : V \rightarrow \mathbb{T} \times \mathbb{T}^\infty \) is a total function such that, if \( D(v) = (t, t') \), then \( t \leq t' \). \( D \) is called the state variable duration function.

The intuition behind the duration function is that if \( D(v) = (t_\text{min}, t_\text{max}) \), then the duration of each interval in which \( x \) has the value \( v \) is included between \( t_\text{min} \) and \( t_\text{max} \) inclusive, if \( t_\text{max} \in \mathbb{T} \), and is not shorter than \( t_\text{min} \) and has no upper bound, if \( t_\text{max} = \infty \).

If \((x, V, T, D)\) is a state variable, then its name \( x \) will be used to denote the state variable itself, sometimes writing \( x = (V, T, D) \). The component \( V \) of the state variable \( x \) will be denoted by \( \text{values}(x) \). The letters \( x, y, z \) will be used as meta-variables for state variable names, \( v \) for elements of \( V \) and \( t, b, e \) for natural numbers (time points), all such meta-variables possibly decorated by subscripts and superscripts.

**Example 1.** In our running example, the timeline-based specification identifies two state variables, that will be called pm (for “pointing mode”) and gs (for “ground station visibility”).

The state variable \( pm \) models the satellite’s pointing sub-system which can be oriented towards the earth to communicate with the ground station or towards a predefined target to gather scientific data. The state variable is the tuple \((pm, V, T, D)\), where \( V = \{\text{Earth, Comm, Maintenance, Science, Slewing}\} \); \( T \) is the value transition function such that \( T(\text{Earth}) = \{\text{Comm, Maintenance, Slewing}\} \); \( T(\text{Comm}) = \{\text{Earth}\} \); \( T(\text{Maintenance}) = \{\text{Earth}\} \); \( T(\text{Science}) = \{\text{Slewing}\} \); \( T(\text{Slewing}) = \{\text{Earth, Science}\} \), and \( D \) is the duration function such that \( D(\text{Earth}) = (1, \infty) \), \( D(\text{Comm}) = (60, \infty) \), \( D(\text{Maintenance}) = (90, 90) \), \( D(\text{Science}) = (36, 58) \), \( D(\text{Slewing}) = (30, 30) \).

The state variable \( gs \) models the ground station visibility windows during the satellite orbit and it is the tuple \((gs, V', T', D')\), where \( V' = \{\text{Visible, NotVisible}\} \); \( T' \) is such that \( T'(\text{Visible}) = \{\text{NotVisible}\} \); \( T'(\text{NotVisible}) = \{\text{Visible}\} \); and \( D' \) is the duration function mapping every value in \( V' \) to \((1, \infty)\).
In what follows, we assume that, whenever a set of state variables \( SV \) is considered, for every distinct pair \( x = (V, T, D) \) and \( y = (V', T', D') \in SV, x \neq y \) and \( V \cap V' = \emptyset \).

The behavior of state variables may be restricted by requiring that some temporal relations hold between intervals with given state variable values. The temporal relations considered in this work are Allen's qualitative temporal relations between intervals [18],\(^1\) represented by the elements of the set \( Rel = \{ \text{equals}, \text{before}, \text{meets}, \text{overlaps}, \text{finishes}, \text{contains}, \text{starts} \} \) (the corresponding inverse relations are omitted), and a set of relations between intervals and time points \( tpRel = \{ \text{starts}_{\text{at}}, \text{ends}_{\text{at}}, \text{starts}_{\text{before}}, \text{starts}_{\text{after}}, \text{ends}_{\text{before}}, \text{ends}_{\text{after}} \} \). The temporal intervals considered in this work are usually open on the right: an interval \([b, e)\) denotes the set \( \{ t | b \leq t < e \} \). When needed, the standard notation \([b, e]\) will be used to denote \( \{ t | b \leq t \leq e \} \).

**Definition 2.** A temporal relation between intervals is an expression of the form \([b, e) R [b', e')\) with \( b, b', e, e' \in \mathbb{T} \). The relation \([b, e) R [b', e')\) is said to hold if:

<table>
<thead>
<tr>
<th>Relation</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>equals</td>
<td>( b = b' \text{ and } e = e' )</td>
</tr>
<tr>
<td>before</td>
<td>( e &lt; b' )</td>
</tr>
<tr>
<td>meets</td>
<td>( e = b' )</td>
</tr>
<tr>
<td>overlaps</td>
<td>( b &lt; b' &lt; e &lt; e' )</td>
</tr>
<tr>
<td>finishes</td>
<td>( b &lt; b' \text{ and } e = e' )</td>
</tr>
<tr>
<td>contains</td>
<td>( b &lt; b' \text{ and } e &lt; e' )</td>
</tr>
<tr>
<td>starts</td>
<td>( b = b' \text{ and } e &lt; e' )</td>
</tr>
</tbody>
</table>

A temporal relation between an interval and a timepoint is an expression of the form \([b, e) R t\), where \( R \in tpRel \) and \( b, e, t \in \mathbb{T} \). The relation \([b, e) R t\) holds iff:

<table>
<thead>
<tr>
<th>Relation</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>starts at</td>
<td>( b = t )</td>
</tr>
<tr>
<td>ends at</td>
<td>( e = t )</td>
</tr>
<tr>
<td>before</td>
<td>( b &lt; t )</td>
</tr>
<tr>
<td>after</td>
<td>( b &gt; t )</td>
</tr>
<tr>
<td>before</td>
<td>( e &lt; t )</td>
</tr>
<tr>
<td>after</td>
<td>( e &gt; t )</td>
</tr>
</tbody>
</table>

\(^{1}\)In order to simplify the presentation, quantitative temporal constraints, used by systems such as EUROPA [5] and APSI-TIP [7], are not defined in this work, though they can easily fit in this framework.

In order to properly state the form of a synchronization among state variables, a (potentially infinite) set \( X \) of names will be used, whose elements are all different from variable names, values and numbers. The elements of \( X \) are called *token variables* and are intended to denote intervals with associated state variable values.

**Definition 3.** An atom is either the special constant \( \top \) or an expression of the form \( a_i R a_j \) or \( a_i R' t\), where \( a_i \) and \( a_j \) are token variables, \( R \in Rel, R' \in tpRel \) and \( t \in \mathbb{T} \). A positive boolean formula (PBF, for short) is an expression built up from atoms using the binary operators \( \land \) (conjunction) and \( \lor \) (disjunction). If \( C \) is a PBF, a choice for \( C \) is any conjunction \( A_1 \land \cdots \land A_n (n \geq 1) \) of atoms occurring in \( C \) that logically implies \( C \).

A synchronization rule is an expression of the form

\[
a_0(x_0 = v_0) \rightarrow \exists a_1[x_1 = v_1] \ldots a_n[x_n = v_n].C
\]

where (i) \( a_0, \ldots, a_n \) are distinct token variables; (ii) for all \( i = 0, \ldots, n, x_i \) is a state variable and \( v_i \in \text{values}(x_i) \); and (iii) \( C \) is a PBF where only the token variables \( a_0, \ldots, a_n \) occur. The left-hand part of the synchronization rule, \( a_0(x_0 = v_0) \), is called the trigger of the rule.

A synchronization rule with empty trigger is an expression of the form:

\[\top \rightarrow \exists a_1[x_1 = v_1] \ldots a_n[x_n = v_n].C\]

Intuitively, a synchronization rule with non-empty trigger of the above form requires that, whenever the state variable \( x_0 \) assumes the value \( v_0 \) in some interval \( a_0 \), there are tokens \( a_i \) (\( 1 \leq i \leq n \)) where the variable \( x_i \) has the value \( v_i \) and one of the possible choices for making \( C \) true holds. (if \( C = \top \), no temporal relation is required to hold). When the trigger is empty, the existence of the intervals \( a_i \) and the relations among them have to hold unconditionally. The use of token variables (that is absent in [12]) allows for mentioning different intervals having the same value. In fact, although the token variables \( a_0, \ldots, a_n \) are pairwise different, multiple occurrences of state variable names and values are allowed. Synchronization rules with empty trigger are useful to represent *domain invariants*, as well as planning goals (both called “facts” in [12]).

It is worth pointing out that if only expressions in disjunctive normal form are allowed instead of PBFs in the \( C \)-part of synchronization rules, then the choices for \( C \) can just be taken as the disjuncts making up \( C \). However, though expressiveness stays the same, the formulation of constraints can grow exponentially larger.

**Example 2.** A sample operational constraint in the satellite domain may require that the satellite communicates with the Earth only when the ground station is visible. A synchronization rule expressing this constraint is the following:

\[
a_{0[H]}[pm = \text{Comm}] \rightarrow \exists a_1[gv = \text{Visible}].a_1 \text{ contains } a_0
\]

According to this rule, whenever the state variable \( pm \) assumes the value \( \text{Comm} \) in an interval \( a_0 \), the state variable \( gv \) has the value \( \text{Visible} \) in an interval \( a_1 \) containing \( a_0 \).

Synchronization rules with empty trigger may be useful to state known facts, such as, for instance:

\[
\top \rightarrow \exists a_1[pm = \text{Earth}].a_1 \text{ starts at } 0
\]
This rule represents the fact that, at the beginning, the satellite is locked on the Earth. Synchronization rules with empty trigger are also used to represent planning goals, as will be described later on.

**Definition 4.** A planning domain is a pair \((SV, S)\), where \(SV\) is a set of state variables and \(S\) a set of synchronization rules.

The set of state variables of a planning domain could be partitioned into controllable and uncontrollable ones. Since however this work does not address controllability issues, this distinction is ignored here.

This work considers temporally extended goals: a planning goal specifies that some variables have to assume some given values in some intervals and, possibly, that such intervals must satisfy some temporal relations.

**Definition 5.** A planning goal \(G\) for a domain \(D = (SV, S)\) is a pair \((\Gamma, \Delta)\), where: (i) \(\Gamma\) is a set of expressions of the form \((g, x, v)\), where \(g\) is a token variable, called the goal name, \(x \in SV\) and \(v \in values(x)\); (ii) \(\Delta\), called relational goal, is a PBF containing only goal names occurring in \(\Gamma\).

A goal \(G = (\Gamma, \Delta)\), with \(\Gamma = \{(g_1, x_1, v_1), \ldots, (g_n, x_n, v_n)\}\), is represented by a synchronization rule \(S_G\) with empty trigger of the form: \(\rightarrow \exists g_1[x_1 = v_1] \ldots g_n[x_n = v_n] \Delta\).

If \((\Gamma, \Delta)\) is a planning goal, the elements of \(\Gamma\) will sometimes be denoted by their name, using also the notation \(g = (x, v)\).

It is worth pointing out that restrictions on the begin and end intervals of a given goal (like in [12], [7]) can be expressed by means of relational goals. In particular, if the begin point of a given goal \(g\) is required to be in the interval \([b, b')\) and its end point in \([e, e')\), then such restrictions are expressed by the conjunction of the following PBFs:

\[
(g \text{ starts after b }) \lor (g \text{ starts at b }), \text{ if } b > 0, \text{ otherwise } T; \\
g \text{ starts before } b', \text{ if } b' < \infty, \text{ otherwise } T; \\
g \text{ ends after e } \lor (g \text{ ends at e }), \text{ if } e > 0, \text{ otherwise } T; \\
g \text{ ends before } e', \text{ if } e' < \infty, \text{ otherwise } T.
\]

**Example 3.** A simple planning goal for the satellite domain may be that the satellite has to perform a scientific operation and then communicate the results to the ground station in order to accomplish the mission. It is represented by the pair \((\Gamma, \Delta)\) where \(\Gamma = \{(g_1, pm, Science), (g_2, pm, Comm)\}\) and \(\Delta = g_1 \land g_2\).

It is represented by the synchronization rule

\[
\rightarrow \exists g_1[pm = Science] g_2[pm = Comm] g_1 \land g_2
\]

Analogously, if the satellite has to perform a scientific and a maintenance operation and we want to specify an ordering constraint between them, i.e. the maintenance operation cannot start until the scientific one is completed, the goal is \(G = (\Gamma, \Delta)\) where \(\Gamma = \{(g_1, pm, Science), (g_2, pm, Maintenance)\}\) and \(\Delta = g_1 \land g_2\). The corresponding synchronization rule is:

\[
\rightarrow \exists g_1[pm = Science] g_2[pm = Maintenance] g_1 \land g_2
\]

A planning problem consists of a domain, a goal and a planning horizon, i.e. the time up to which the behavior has to be planned.

**Definition 6.** A planning problem is a triple \((D, G, H)\), where \(D\) is a planning domain, \(G\) a planning goal for \(D\) and \(H \in \mathbb{T}\) is the planning horizon.

**IV. TIMELINES AND PLANS**

The focus of this work is on flexible temporal plans, each of which represents a set of “standard” timelines, i.e., timelines where begin and end points of every interval are fixed. Therefore, “non-flexible” timelines and plans also need to be formally defined. In the following, they will be generically called “timelines” and “plans”.

**Definition 7.** Let \(x = (V, T, D)\) be a state variable. A token for the variable \(x\) is a tuple of the form \((x^0, v, b, e)\), where \(j \in \mathbb{N}, v \in V, b < e\) are numbers in \(\mathbb{T}\), and \(t_{min} \leq e - b \leq t_{max}\) where \((t_{min}, t_{max}) = D(v)\).

If \(x = (V, T, D)\) is a state variable, a timeline for \(x\) is a finite sequence of tokens for \(x\), whose identifiers are \(x^0, x^1, \ldots, x^k\): 

\[
TL_x = (x^0, v_0, b_0, e_0), (x^1, v_1, b_1, e_1), \\
\ldots, (x^k, v_k, b_k, e_k)
\]

where \(v_0 = 0\), and for all \(i = 0, \ldots, k - 1\), \(e_i = b_{i+1}\) and \(v_i + 1 \in T(v_i)\). The value \(e_k\) is called the temporal horizon of the timeline, and \(k + 1\) its length.

The temporal horizon \(H\) of a set of timelines is the minimum among the temporal horizons of its elements.

Intuitively, a token \((x^i, v, b, e)\) in \(TL_x\) means that the state variable \(x\) has the value \(v\) in the interval \([b, e)\). The element \(x^i\) of the token is called the token identifier; it will often be used to denote the whole token, sometimes writing \(x^i = (v, b, e)\); \(\text{val}(x^i)\) will denote the value \(v\) of the token, and \(b\) and \(e\) are called, respectively, the begin and end points of the token.

The notation \(TL(SV, H)\) will be used to denote a set of timelines with horizon \(H\), for the state variables in the set \(SV\). A set of timelines with horizon \(H\) describes the behavior of each state variable in \(SV\) at least within the time point \(H\).

**Example 4.** In the satellite domain, a possible timeline for the state variable \(pm\) (the pointing sub-system) within the temporal horizon \(H = 250\) is the following

\[
TL_{pm} = (p0^0, Earth, 0, 10), \\
p1^3, Slowing, 10, 40) \\
p2^2, Science, 40, 80) \\
p3^2, Slowing, 80, 110) \\
p4^2, Earth, 110, 120) \\
p5^2, Comm, 120, 200) \\
p6^2, Earth, 200, 250)
\]

Each token \(pm^i\) contains the value the state variable assumes in the associated temporal interval \([b_i, e_i]\). For instance, in the interval \([110, 120]\) the satellite is locked on the Earth. It is important to point out that the sequence of tokens in the timeline complies with the state variable transition function \(T\) and the state variable value duration function \(D\).

The following definition introduces the semantics of synchronizations on (non-flexible) timelines. Since the statement of a synchronization rule makes use of token variables, each of them must be “interpreted”, i.e. mapped to a token of the considered timelines.

**Definition 8.** If \(x^i = (b, e)\) and \(y^j = (b', e')\) are tokens and \(t \in \mathbb{T}\), then expressions of the form \(x^iRy^j\) and \(x^iRt\) are called relations on tokens. The relation \(x^iRy^j\) holds iff \([b, e) R [b', e')\) holds. And the relation \(x^iRt\) holds iff \([b, e) R [t, t]\)
be equal to horizon of solution plans is not smaller than be determined. This is taken into account by requiring that the timeline $TL_x \in TL(SV, H)$, and such that $val(\varphi(a)) = v_i$ for all $i = 1, \ldots, n$.

Let $C$ be a PBF and $TL(SV, H)$ a set of timelines, where $SV$ includes all the state variables occurring in $C$. A token assignment $\varphi$ on $TL(SV, H)$ satisfies $C$ if there exists a choice $A_1 \wedge \cdots \wedge A_m$ for such that for every atom $A \in \{A_1, \ldots, A_m\}$, (i) if $A = a_iR_{ij}$ then the relation $\varphi(a_i)R\varphi(a_j)$ holds; and (ii) if $A = a_iR_t$ then the relation $\varphi(a_i)R_t$ holds.

Let $S = a_0[x_0 = v_0] \rightarrow \exists a_1[x_1 = v_1] \ldots a_n[x_n = v_n].C$ be a synchronization rule. A set of timelines $TL(SV, H)$ satisfies the synchronization rule $S$ if $\{x_0, x_1, \ldots, x_n\} \subseteq SV$ and for every token $x_0^k$ in $TL_x$, such that $val(x_0^k) = v_0$, there exists a token assignment $\varphi$ for the set $\{a_0[x_0 = v_0], \ldots, a_n[x_n = v_n]\}$ on $TL(SV, H)$ such that $\varphi(a_0) = x_0^k$ and $\varphi$ satisfies $C$.

A set of timelines $TL(SV, H)$ satisfies a synchronization rule with empty trigger $\top \rightarrow \exists a_1[x_1 = v_1] \ldots a_n[x_n = v_n].C$ if $\{x_1, \ldots, x_n\} \subseteq SV$ and there exists a token assignment $\varphi$ for the set $\{a_1[x_1 = v_1], \ldots, a_n[x_n = v_n]\}$ on $TL(SV, H)$ that satisfies $C$.

Let $SV$ be a set of state variables, $S$ a set of synchronization rules concerning variables in $SV$ and $TL(SV, H)$ a set of timelines for the state variables in $SV$. $TL(SV, H)$ satisfies the set of synchronizations $S$ iff $TL(SV, H)$ satisfies all the elements of $S$.

Example 5. Let us consider, for instance, the synchronization rule given in Example 2 that constrains the satellite to communicate only when the ground station is visible. Let $TL_{pm}$ and $TL_{gv}$ be two timelines. and let us assume that the timeline for the pointing system contains a single token whose value is $Comm$:

$$TL_{pm} = \ldots, (pm^t, Comm, 90, 170), \ldots$$

and that the timeline $TL_{gv}$ for the ground station visibility contains a token $(gv^v, V_{visible}, 70, 185)$.

The synchronization rule is satisfied by the set of timelines $\{TL_{pm}, TL_{gv}\}$. In fact, $pm^t$ is the only token in $TL_{pm}$ whose value is $Comm$, and, for this token, the token assignment $\varphi$ such that $\varphi(a_0) = pm^t$ and $\varphi(a_1) = gv^v$ satisfies the PBF $a_1$ contains $a_0$. This holds because $\varphi(a_1)$ contains $\varphi(a_0)$ (i.e. $gv^v$ contains $pm^t$) holds, since $[70, 185]$ contains $[90, 170]$ holds.

Before defining timeline-based plans, it must be observed that the meaning of the planning horizon $H$ in a planning problem is twofold. First of all, it represents the time up to which the evolution of every component of the system is to be determined. This is taken into account by requiring that the horizon of solution plans is not smaller than $H$ (it cannot just be equal to $H$ since the timelines in a given set may have different horizons); consequently, a solution plan is allowed to cross the temporal horizon given by the planning problem.

On the other side, every planning goal has to be accomplished within the horizon. In order to deal with this second restriction, the original planning goal $G$ has to be extended to a new one, $G(H)$, setting $H$ as the time limit to perform every required activity.

Definition 9. A plan is a set $TL(SV, H)$ of timelines for the variables in $SV$ over a given horizon $H$. A plan $TL(SV, H)$ is valid w.r.t. the planning domain $D = (SV', S)$ if $SV = SV'$ and $TL(SV, H)$ satisfies the set of synchronizations $S$.

A plan $TL(SV, H)$ fulfills the planning goal $G$ if it satisfies the synchronization rule $S_t$ representing $G$.

Let $G = (\Gamma, \Delta)$ be a planning goal, $H \in \mathbb{T}$ and $g_1, \ldots, g_n$ all the goal names occurring in $G$. Then:

$$G(H) = (\Gamma, \Delta \wedge \bigwedge_{1 \leq i \leq n} (g_i \text{ ends before } H))$$

Given a planning problem $P = (D, G, H)$, a plan $TL(SV, H')$ is a solution plan for $P$ if $TL(SV', H')$ is valid w.r.t. $D$, $H' \geq H$ and $TL(SV, H)$ fulfills the planning goal $G(H)$.

V. FLEXIBLE TIMELINES AND PLANS

Time flexibility is essentially taken into account by considering temporal intervals (rather than time points) in the begin and end elements of tokens. This relaxation may however lead to violate some constraints of the planning domain. As a consequence, flexible timelines and plans have to be equipped with additional pieces of information, so that they represent only non-flexible timelines and plans, respectively, that are valid w.r.t. the underlying planning domain. So, although flexible timelines and plans can be seen as envelopes of non-flexible ones, some care must be taken in order to include all the necessary pieces of information in their representation.

To begin with, let us consider, for instance, a flexible token $x^t$ for a state variable $x = (V, T, D)$ representing the fact that the value of $x$ is in an interval whose begin and end points belong, respectively, to $[10, 20]$ and $[30, 50]$. Among the non-flexible tokens represented by $x^t$ there might be one beginning at 10 and ending at 50. But this should not be the case if, for instance, $D(v) = (20, 30)$. For this reason, a flexible token must also include duration constraints, that may also be stricter than what established by the state variable duration function.

Definition 10. If $x = (V, T, D)$ is a state variable, a flexible token for the variable $x$ is a tuple of the form:

$$(x^d, v, (b, b'), (e, e'), (d, d'))$$

where $j \in \mathbb{N}, b, b', e, e', d, d' \in \mathbb{T}$, $d' \in T^\infty, v \in V, b < e'$ and $d_{min} \leq d \leq d' \leq d_{max}$, where $(d_{min}, d_{max}) = D(v)$. The element $x^d$ is the token identifier and $val(x^d)$ denotes the value $v$ of the token.

A flexible timeline for $x$ is a finite sequence of flexible tokens for $x$, whose identifiers are $x_0^d, x_1^d, \ldots$:

$$FTL_x = (x_0^d, v_0, (b_0, b'_0), (e_0, e'_0), (d_0, d'_0)), (x_1^d, v_1, (b_1, b'_1), (e_1, e'_1), (d_1, d'_1)), \ldots, (x_k^d, v_k, (b_k, b'_k), (e_k, e'_k), (d_k, d'_k))$$

where $b_0 = b'_0 = 0$ and for all $i = 0, \ldots, k - 1, e_i = b_{i+1}, e'_i = b'_{i+1}$ and $v_{i+1} \in T(v_i)$. The temporal horizon of the timeline is the value $e_k$ and its length is $k + 1$. 
The temporal horizon \( H \) of a set \( FTL(SV,H) \) of flexible timelines for the state variables in \( SV \) is the minimum among the temporal horizons of its elements.

Intuitively, a token \( x^i \) of the above form represents tokens whose begin points are in the (closed) interval \([b,b']\), whose end points are in \([e,e']\) and whose duration is in \([d,d']\). The next definition introduces the notions of projection and instance of flexible timelines. Projections may not satisfy the duration requirements, while every token of an instance satisfies them. Among the projections of a flexible timeline, the particular one where the begin and end points of every token are the lower bounds of the corresponding intervals is considered.

**Definition 11.** A projection of a flexible token \( x^i = (v, (b,b'), (e,e'), (d,d')) \) is a (non-flexible) token \( (x^i, v, b', e') \), where: \( b \leq b' \leq b \) and \( e \leq e' \leq e \). The projection \( x^i, v, b, e \) is called the earliest start time projection (EST projection, for short) of \( x^i \). If a projection \( (x^i, v, b', e') \) of \( x^i \) satisfies also the condition \( d \leq e' - b' \leq d' \), then it is called an instance of the flexible token \( x^i \).

A (non-flexible) timeline \( TL_x \) for the state variable \( x \) is a projection – or, respectively, the EST projection, or an instance – of \( FTL_x \) if \( TL_x \) and \( FTL_x \) have the same length \( k \) and for all \( i \), \( 0 \leq i \leq k - 1 \), the token \( x^i \) of \( TL_x \) is a projection – or respectively, the EST projection, or an instance – of the flexible token \( x^i \) of \( FTL_x \).

Let \( FTL(SV,H) \) be a set of flexible timelines. A projection – or, respectively, the EST projection, or an instance – \( TL(SV,H') \) of \( FTL(SV,H) \) is a set of (non-flexible) timelines for the state variables in \( SV \), where each \( TL_x \in TL(SV,H') \) is a projection – or, respectively, the EST projection, or an instance – of the flexible timeline \( FTL_x \in FTL(SV,H) \).

It is worth pointing out that the temporal horizon of a projection of a flexible timeline \( FTL_x \) may be different, but not smaller, than the horizon of \( FTL_x \). Consequently, the temporal horizon of a projection of a set \( FTL(SV,H) \) is either equal to or greater than \( H \).

**Example 6.** Let us consider the following flexible timeline for the state variable \( pm \) in the satellite case study (see again Figure 2).

\[
FTL_{pm} = (pm^0, Earth, (0,0), (110,120), (110,120), (pm^1, Slewing, (110,120), (140,150), (30,30)), \]
\[
(pm^2, Science, (140,150), (181,203), (41,58)), \]
\[
(pm^3, Slewing, (181,203), (211,233), (30,30)), \]
\[
(pm^4, Earth, (211,233), (250,250), (17,39))
\]

Each projection of this timeline represents a series of choices for the tokens begin and end points, within the allowed intervals. Two possible projections are the following timelines:

\[
TL_{pm}^1 = \]
\[
(pm^0, Earth, 0, 110), \]
\[
(pm^1, Slewing, 110, 140), \]
\[
(pm^2, Science, 140, 181), \]
\[
(pm^3, Slewing, 181, 211), \]
\[
(pm^4, Earth, 211, 250)
\]

\[
TL_{pm}^2 = \]
\[
(pm^0, Earth, 0, 115), \]
\[
(pm^1, Slewing, 115, 148), \]
\[
(pm^2, Science, 148, 185), \]
\[
(pm^3, Slewing, 185, 215), \]
\[
(pm^4, Earth, 215, 250)
\]

![Fig. 2. A timeline FTL_{pm} for the pointing mode state variable and two of its projections TL_{pm}^1 and TL_{pm}^2.](image)

The timeline \( TL_{pm}^1 \) is the EST projection of the flexible timeline \( FTL_{pm} \), since each non-flexible token starts at the lower bound of the begin point interval of the corresponding flexible token.

It is worth pointing out that the token \( pm^1 \) of \( TL_{pm}^1 \) is a projection of the corresponding token in \( FTL_{pm} \), but it is not an instance thereof, since the duration of the interval \((148 – 115)\) is not in the allowed interval \([30,30]\).

A flexible token represents the set of its instances, and the same holds for flexible timelines and sets of flexible timelines. Nevertheless, a set of flexible timelines does not convey enough information to represent a flexible plan. Let us consider, for instance, a domain with a synchronization rule \( S \) of the form \( a_0[x = v] \rightarrow \exists a_1[y = v'] \). \( a_0 \) meets \( a_1 \) and flexible timelines for the state variables \( x \) and \( y \) containing, respectively, a token \( x^i = (v, (10,20), (30,50), (20,30)) \) and \( y^j = (y', (30,50), (45,70), (15,20)) \). Obviously, not every pair of instances of \( x^i \) and \( y^j \) satisfies \( S \) (see also Example 7 below). Thus, the representation of a flexible plan must include also information about the relations that have to hold between tokens in order to satisfy the synchronization rules of the planning domain. In the example above, it would include the relation \( x^i \) meets \( y^j \). In general, a flexible plan includes a set of relations on tokens. When there are different ways how a synchronization rule can be satisfied by the same set \( FTL(SV,H) \) of flexible timelines, each flexible plan represents a choice among them, and different plans with the same set \( FTL(SV,H) \) represent different ways to satisfy the synchronizations.

**Definition 12.** A flexible plan \( \Pi \) over the horizon \( H \) is a pair \( (FTL(SV,H), R) \), where \( FTL(SV,H) \) is a set of flexible timelines and \( R \) is a set of relations on tokens, involving token identifiers in some timeline in \( FTL(SV,H) \).

A (non-flexible) plan \( TL(SV,H') \) is an instance of the flexible plan \( \Pi = (FTL(SV,H), R) \) iff \( TL(SV,H') \) is an instance of \( FTL(SV,H) \) and it satisfies every relation in \( R \).

A flexible plan represents the set of its instances. The notation \( \Pi(H) \), when used, will denote a flexible plan whose horizon is \( H \). Note, here too, that the temporal horizon of instances may be greater than the flexible plan horizon.

We now turn to define the semantics of synchronizations on flexible timelines. Essentially, a plan \( \Pi = (FTL(SV,H), R) \) satisfies a synchronization rule \( S \) if the constraints represented by \( S \) are guaranteed to hold whenever the relations in \( R \) are satisfied by \( FTL(SV,H) \). In other terms, \( R \) represents a possible choice to satisfy \( S \).

The notion of token assignment is generalized to flexible timelines in the obvious way: a token assignment for the set \( \{a_1[x_1 = v_1], \ldots, a_n[x_n = v_n]\} \) on \( FTL(SV,H) \) is a function \( \varphi \) mapping every \( a_i \) to a token of the timeline
Definition 13. Let $C$ be a PBF, $\Pi = (\text{FTL}(SV, H), R)$ a flexible plan, where $SV$ contains all the state variables occurring in $C$, and $\varphi$ a token assignment on $\text{FTL}(SV, H)$. The plan $\Pi$ satisfies $C$ with $\varphi$ if there exists a choice $A_1 \land \cdots \land A_m$ for $C$ such that for every atom $A \in \{A_1, \ldots, A_m\}$, (i) if $A = a_i R a_j$, then $\varphi(a_i) R \varphi(a_j) \in R$; and (ii) if $A = a_i R t$, then $\varphi(a_i) t R \in R$.

A flexible plan $\Pi = (\text{FTL}(SV, H), R)$ satisfies a synchronization rule with non-empty trigger $a_0[x_0 = v_0] \rightarrow \exists a_1[x_1 = v_1] \ldots a_n[x_n = v_n].C$ if for every flexible token $x_0^m$ of the timeline $\text{FTL}_x = \text{FTL}(SV, H)$ such that $\text{val}(x_0^m) = v_0$, there exists a token assignment $\varphi$ for the set $\{a_0[x_0 = v_0], \ldots, a_n[x_n = v_n]\}$ on $\text{FTL}(SV, H)$ such that $\varphi(a_0) = x_0^m$ and $\Pi$ satisfies $C$ with $\varphi$.

$\Pi$ satisfies a synchronization rule with empty trigger $T \rightarrow \exists a_1[x_1 = v_1] \ldots a_n[x_n = v_n].C$ if there exists a token assignment $\varphi$ for the set $\{a_1[x_1 = v_1], \ldots, a_n[x_n = v_n]\}$ on $\text{FTL}(SV, H)$, such that $\Pi$ satisfies $C$ with $\varphi$.

Example 7. Let $\text{FTL}_{pm}$ be a flexible timeline for the state variable $pm$ representing the evolution of the pointing sub-system through the states $Earth \rightarrow Slewing \rightarrow Science \rightarrow Slewing \rightarrow Earth \rightarrow Comm \rightarrow Earth$, whose sixth token (the only one where the satellite is in communication mode) is:

$$pm^5 = (Comm, (180, 190), (250, 300), (70, 100))$$

Let moreover $\text{FTL}_{gv}$ be a timeline containing the token

$$gv^4 = (Visible, (170, 200), (280, 320), (110, 150))$$

The flexible plan $\Pi = (\text{FTL}(SV, H), R)$, where $SV = \{pm, gv\}$, $\text{FTL}(SV, H) = \{\text{FTL}_{pm}, \text{FTL}_{gv}\}$ and $R = \{gv^4 \text{ contains } pm^5\}$, satisfies the synchronization rule of Example 2: taking $\varphi$ such that $\varphi(a) = pm^2$ and $\varphi(a_1) = gv^4$, we have $\varphi(a_1)$ contains $\varphi(a_0) ∈ R$.

The case of synchronization rules with empty trigger is treated similarly.

Let $\mathcal{P} = (D, G, H)$ be a planning problem and $\Pi(H')$ a flexible plan. $\Pi(H')$ is a flexible solution problem for $\mathcal{P}$ if $\Pi(H')$ is valid w.r.t. $D$, $H' \geq H$, and it satisfies the synchronization rule $S_{G,H}$ representing $G(H)$.

The next result establishes that the information encoded by a complete flexible plan $\Pi$ is sufficient to ensure that every instance of $\Pi$ is valid w.r.t. the planning domain.

Theorem 1. If the flexible plan $\Pi$ is complete w.r.t. the planning domain $D$, then every instance of $\Pi$ is valid w.r.t. $D$.

Proof: Since the state variables of a flexible plan $\Pi$ and any of its instances are the same, it is sufficient to show that if $\Pi$ satisfies a synchronization rule $S$, then every instance of $\Pi$ satisfies $S$.

Let us assume that $\Pi = (\text{FTL}(SV, H), R)$ satisfies

$$S = a_0[x_0 = v_0] \rightarrow \exists a_1[x_1 = v_1] \ldots a_n[x_n = v_n].C$$

and $\text{FTL}(SV, H')$ is an instance of $\Pi$.

Let $x_0^m = (v_0, b, e)$ be a token in $TL_{x_0} \subseteq \text{FTL}(SV, H')$. Since $\Pi$ satisfies $S$, there exists a token assignment $\varphi$ for the annotated token variables $a_0[x_0 = v_0], \ldots, a_n[x_n = v_n]$ with $\varphi(a_0) = x_0^m$, such that $\Pi$ satisfies $C$ with $\varphi$. Therefore, in particular, there exists tokens $\varphi(a_i)$ such that $\text{val}(\varphi(a_i)) = v_i$, for all $i = 0, \ldots, n$. Let then $\{a_1, \ldots, a_m\}$ be the choice for $C$ such that for every atom $A \in \{A_1, \ldots, A_m\}$: (i) if $A = a_i R a_j$, then $\varphi(a_i) R \varphi(a_j) \in R$; since $\text{FTL}(SV, H')$ is an instance of $\Pi$, it satisfies the relation $\varphi(a_i) R \varphi(a_j)$; (ii) if $A = a_i R t$, then $\varphi(a_i) t R \in R$, and, like above, $\text{FTL}(SV, H')$ satisfies $\varphi(a_i) R t$. Therefore, $\text{FTL}(SV, H')$ satisfies $S$.

As a direct consequence of Definition 14 and the above theorem, if the flexible plan $\Pi$ is valid w.r.t. a planning domain $D$ then there exists an instance of $\Pi$ that is valid w.r.t. $D$.

Checking plan consistency is an important task. Considering that a flexible plan represents the set of its instances, it can be useful to identify a set $\Theta$ of flexible plans for which an effective consistency check procedure exists, and yet every (non-flexible) valid plan is an instance of a plan in $\Theta$. The next theorem establishes that, when $T = \mathbb{N}$, the set of flexible plans can be safely restricted to the set $\Theta_{EST}$ such that $\Pi = (\text{FTL}(SV, H), R) ∈ \Theta_{EST}$ if and only if the $EST$ projection of $\text{FTL}(SV, H)$ is an instance of $\Pi$. In other words, $\Theta_{EST}$ represents the whole set of non-flexible valid plans. Since goals are represented by synchronization rules, the theorem implies that restricting the attention to flexible plans in $\Theta_{EST}$ does not lead to miss any non-flexible solution plan.

It is worth noticing that, from a practical point of view, the significance of Theorem 2 is due to the fact that, although the Simple Temporal Problem formalism assumes time to be continuous, this feature is usually not exploited by real P&S systems.

Below, the notation $\Pi ⊆ \Pi'$ will be used to mean that every instance of $\Pi$ is an instance of $\Pi'$.

Theorem 2. If $T = \mathbb{N}$, then for every consistent flexible plan $\Pi(H)$ there exists a consistent plan $\Pi'(H') ∈ \Theta_{EST}$, with $H' \geq H$, such that $\Pi(H) \subseteq \Pi'(H')$.

Proof: If $x^d = (v, (b, b'), (e, e'), (d, d'))$ is a flexible token, the values $b$ and $e$ will be denoted by low_b(x^d) and low_e(x^d), respectively.
We first show that, for any flexible timeline $\text{FTL}_a$ with a non-empty set of instances, there exists a timeline $\text{FTL}'_a$ over a temporal horizon equal to or greater than the horizon of $\text{FTL}_a$, having the same set of instances as $\text{FTL}_a$, and such that

\[(\alpha) \text{ for any token } (x^k, v, (b, b'), (e, e'), (d, d')) \text{ in } \text{FTL}_a, \quad d \leq e - b \leq d'.\]

Let $(x^k, v, (b, b'), (e, e'), (d, d'))$ be any token in $\text{FTL}_a$ such that either $e - b < d$ or $e - b > d'$. In the first case $(e < b + d)$ the corresponding token $x^k$ in $\text{FTL}'_a$ is obtained from $x^k$ by replacing $\text{low}_e(x^k)$ with $b + d$; otherwise, if $e - b > d'$ (i.e. $b < e - d$), then the corresponding token in $\text{FTL}'_a$ is obtained from $x^k$ by replacing $\text{low}_b(x^k)$ with $e - d'$. The token which follows (or precedes) $x^k$ in the timeline, is modified accordingly, so that its begin (or end) interval becomes $(b+d, e')$ (or $(e-d', b')$). Obviously, since the begin and end points of any instance $x^k$ in $\text{FTL}_a$ satisfy the duration requirements, the tokens $x^k$ in $\text{FTL}_a$ and $\text{FTL}'_a$ have the same set of instances. Moreover, if the horizon of $\text{FTL}'_a$ changes, it is because $x^k$ is the last token of $\text{FTL}_a$ and the case $e < b + d$ holds; in that case, the lower bound $e$ of the end point of the last token is replaced by $b + d < e$.

As a consequence, we may assume w.l.g. that any token in any timeline in $\text{FTL}(SV, H)$ satisfies the above “local” condition $\alpha$.

Let $\Pi = (\text{FTL}(SV, H), \mathcal{R})$. If $\Pi$ is not consistent, then there exists a token $x^a$ (or a pair of tokens $x^a$ and $y^a$) in some timelines in $\text{FTL}(SV, H)$ and a relation $x^a \geq t_R$ (or $x^a \leq t_R$) in $\mathcal{R}$ that is not satisfied by the EST projection(s) of $x^a$ (and $y^a$).

The construction of $\Pi' = (\text{FTL}'(SV, H'), \mathcal{R})$ is performed by steps, building a sequence of sets of flexible timelines $\text{FTL}_i(SV, H_i) = \text{FTL}(SV, H), \text{FTL}_1(SV, H_1), \ldots, \text{FTL}_n(SV, H_n) = \text{FTL}(SV, H),$ where $\text{FTL}_{i+1}(SV, H_{i+1})$ is obtained from $\text{FTL}_i(SV, H_i)$ by taking into consideration one of the relations in $\mathcal{R}$ that are violated by the EST projection of $\text{FTL}_i(SV, H_i)$. The construction ensures that for all $i, H_i \leq H_{i+1}$, therefore $H' \geq H$. In the following, the elements of such a sequence are denoted simply by $\text{FTL}_i,$ instead of $\text{FTL}_i(SV, H_i).$ Analogously, sets of non-flexible timelines will be simply denoted by $\text{TL}_i$.

At each step, different cases are considered according to the form of the relation on tokens violated by the EST projection of $\text{FTL}_i$. Let us first consider binary relations, of the form $x^a \geq t_R$. For the sake of conciseness, in what follows $x^a$ and $y^a$ denote the token identifiers occurring in the relation, and $x^a$ and $y^a$ the corresponding tokens in the set of timelines $\text{FTL}_i$. Each case shows how to increase the lower bounds of the begin and/or end points of $x^a$ and/or $y^a$ in order to obtain the tokens $x^a$ and $y^a$ whose EST projections satisfy the relation $x^a \geq t_R$. The lower bounds that are not mentioned are left unchanged, and upper bounds are never modified. Moreover, it has to be shown that for every instance $\text{TL}_i$ of $\text{FTL}_i$, $\text{TL}_i$ is also an instance of $\text{FTL}_{i+1}$. The generic tokens corresponding to $x^a$ and $y^a$ in $\text{TL}_i$ will be denoted by $x^a$ and $y^a$, respectively, and they will be shown to be projections of $x^a$ and $y^a$, respectively (i.e. their begin and end points belong to the new allowed ranges), knowing that, since $\text{TL}_i$ is an instance of $\text{FTL}_i$, they satisfy the relation $x^a \geq t_R$.

Let $\text{low}_b(x^a), \text{low}_b(y^a), \text{low}_e(x^a), \text{low}_e(y^a).$ Let moreover $\text{TL}_i$ be any instance of $\text{FTL}_i$, with $x^a = (v, b, b', e)$ and $y^a = (v', b', b', e')$. Since $\text{TL}_i$ is a projection of $\text{FTL}_i$, (i) $b' \leq b$, (ii) $e' \leq e$, (iii) $b' \leq b'_k$ and (iv) $e' \leq e'_k$. Then:

1) If $R = \text{equals}$, then $\text{low}_b(x^a) = \text{low}_b(y^a) = \text{low}_e(x^a) = \text{low}_e(y^a).$ Since $\text{TL}_i$ satisfies $x^a \geq t_R$, $b'_k \leq b_k$ and $e'_k \leq e_k$. Consequently, using also (i)-(iv) $- b_k' \leq b_k,$ $b'_k \leq b_k,$ $e'_k \leq e_k$ and $e'_k \leq e_k$, so that $x^a$ and $y^a$ are projections of $x^a$ and $y^a$, respectively.

2) If $R = \text{before}$, then $\text{low}_b(y^a) = e_n + 1.$ Since, by hypothesis, $x^a$ and $y^a$ do not satisfy $x^a \geq t_R$, $e_n \geq b_k$.
tokens, in such a way that finally the EST projection of the plan satisfies all the required relations on tokens. It is worth pointing out that such a construction is part of the algorithm implemented by the system EPSL (see Section VI) for building a consistent plan, starting from a planning domain, a goal and a set of incomplete timelines.

The assumption $\mathbb{T} = \mathbb{N}$, used in the above proof, could be replaced by the hypothesis that $\Pi = (\text{FTL}(SV, H), \mathcal{R})$ has a finite number of instances. If this is the case, for every token $x'$ of some timeline in $\text{FTL}(SV, H)$, there exists a minimum $b_{min}$ among the begin points of its projections belonging to instances of $\Pi$, and analogously, there exits a minimum $e_{min}$ among the end points of such projections. The lower bounds of the begin and end intervals of $x'$ can be then directly replaced by $b_{min}$ and $e_{min}$, respectively. Then, it can be shown that the EST projection of the so obtained set of timelines satisfies all the relations in $\mathcal{R}$. This non-constructive approach, however, would be of no help when $\mathbb{T} = \mathbb{R}_{\geq 0}$.

VI. EXTENSIBLE PLANNING AND SCHEDULING LIBRARY

The Extensible Planning and Scheduling Library (EPSL), is a P&S tool that has been developed to solve planning problems by exploiting the semantics of flexible timelines given above. More in general, EPSL is the result of a research effort started after the analysis of different timeline-based planning systems, such as [9], [6], [19], [20], as well as some previous experiences in deploying timeline-based P&S solvers for real world problems [21], [22].

EPSL relies on the APSI-TRF [7] modelling framework which provides a timeline-based representation framework for planning domains and problems. In order to comply with the formalization introduced in Sections III, IV and V, the accepted input language specification and the internal timelines representation have been properly adapted. In particular, EPSL currently conforms with the semantics of flexible timelines given in Definition 11 and, consequently, it is able to generate flexible plans compliant with Definition 12. The internal temporal representation, called Plan Database, is still implemented by means of the Simple Temporal Problem formalism [23]. Then, on top of such modelling functionalities, EPSL provides a software machinery which allows users to easily define timeline-based planners by specifying strategies and heuristics to be applied in specific P&S problems. A more detailed description of the framework’s architecture is provided in [16]. However, broadly speaking, the EPSL key point is the “planner-interpretation”. According to this interpretation, a timeline-based planner is defined as the composition of several independent modules that together implement the planner’s solving procedure. Therefore EPSL defines a planner as a tuple $(S, \mathcal{H}, \mathcal{E})$. $S$ is the strategy used to manage the fringe of the search space. The strategy $S$ can be choosen among several built-in options (A*, DFS, BFS, etc). $\mathcal{H}$ is the heuristic function utilized to analyze the current plan looking for flaws and selecting the most promising flaw to solve. Currently the framework is endowed with a built-in option for $\mathcal{H}$ which implements a hierarchical plan decomposition policy. A flaw represents either goals or threats that must be solved in order to find a feasible solution. A threat to timeline consistency is either a timeline gap (i.e. an empty interval on the timeline) or a set of non-scheduled decision (i.e. overlapping decisions that violate state variable capacity constraint). $\mathcal{E}$ is the resolver engine encapsulating the reasoning capabilities of the EPSL framework. $\mathcal{E}$ is composed by several built-in algorithms, called resolvers. Each resolver recognizes and encapsulates a dedicated algorithm to solve a particular type of flaw. These algorithms represent the operators used by the planner to refine the plan. An EPSL planner solves a planning problem $\mathcal{P} = (D, G, H)$ and generates a valid flexible plan $\Pi = (\text{FTL}(SV, H'), \mathcal{R})$ with $H' \succeq H$. During the solving procedure, resolvers are in charge of generating tokens for the planned variables as well as a proper set of temporal constraints to enforce that $\Pi$ represents a suitable set of instances for the flexible timelines also fulfilling the planning goals $G$. Then, the way how an EPSL planner works is a refinement procedure, that resembles the iterative stepwise procedure used to prove Theorem 2.

The “abstract” solving procedure of a generic EPSL planer is summarized below (see [16] for a detailed presentation). The planner initializes the search tree on a problem $\mathcal{P}$ and defines the initial partial plan $\Pi$. While the current plan $\Pi$ contains flaws, the planner tries to refine it until a solution is found. At each solving step, the flaws present in the current plan $\Pi$ are identified and the more relevant is selected according to the heuristic $\mathcal{H}$. Then, the plan $\Pi$ is refined by applying all the resolvers in the engine $\mathcal{E}$ that can handle the selected flaw and successors nodes are enqued into the search space fringe according to the strategy $S$. Namely, the strategy $S$ defines the node evaluation criteria. If all the open planning branches have been investigated and still unsolved flaws are present in $\Pi$, then the planning procedure generates a failure, i.e., a solution plan has not been found. Otherwise, a new current plan is extracted from the fringe and the procedure iterates again. Finally, once the current plan $\Pi$ contains no unsolved flaws, then $\Pi$ represents a solution plan and it is provided as a result. In [16], the combination of DFS search strategy $(S)$ with a hierarchical plan decomposition $(\mathcal{H})$ has been demonstrated to be the most performing EPSL planner configuration for solving two different real world problems. Namely, hierarchical plan decomposition heuristic computes a timeline hierarchy by building an oriented graph where nodes are timelines and edges represent synchronizations between timelines. Then timelines’ hierarchy degrees are computed by running a topological sort algorithm on this graph. Such a hierarchy is exploited during plan refinement to compute flaw’s priority, i.e. the higher is the timeline’s hierarchy degree the higher will be the priority of the related flaws. Therefore when flaws are detected from the current plan $\Pi$ the planner will solve the flaw with the highest priority. Moreover when there are two or more flaws with the same priority (i.e. flaws related to the same timeline) the fail-first principle is applied. Namely, the heuristic selects the hardest flaw to solve, i.e. the flaw with less possible solutions.

EPSL provides an enhanced framework for developing applications in which designers may focus on single aspects of the solving process (i.e., a particular heuristic function or a new resolver in order to add reasoning capabilities to the library) without taking care of all details related to timeline-based planner implementation. In this regard, [16] presents promising experimental results demonstrating the capabilities of EPSL.
VII. CONCLUDING REMARKS

In this paper, a formal account of planning with flexible timelines has been presented aiming at providing a general semantics for planning concepts such as domains, goals, problems, constraints and flexible plans. Some basic properties of the defined concepts have been also stated and proved. Finally, a planning tool, called EPSL, has been briefly presented discussing how it generates flexible plans compliant with the given semantics, i.e., operationalizing the formal semantics introduced in the paper.

The paper presents an initial result with respect to a general research objective, i.e., to define a comprehensive formalization of timeline-based planning considering also execution aspects. In fact, controllability issues have been formalized and investigated for the Simple Temporal Problems with Uncertainty (STPU) in [13], where basic formal notions are given for the dynamic controllability (DC) property (e.g., see a more recent work in [14]). Analogously, the same problem has been addressed following a different perspective, i.e., exploiting a method based on Timed Game Automata to generate DC controllers suitable for the execution of flexible timeline-based plans [24], [25]. Thus, the formal account given here is to be extended representing uncontrollable events and durations as well as properly defining different controllability properties. In this regard, the definition of dynamic controllability and the characterization of flexible plan controllers constitute a necessary further work in order to bridge the gap between planning and execution.

ACKNOWLEDGMENTS

The authors wish to thank the anonymous referees of this work for useful comments and suggestions. Orlandini is partially supported by the Italian Ministry of Education, University and Research (MIUR) and the National Research Council of Italy (CNR): GECKO Project - Progetto Bandiera "La Fabbrica del Futuro".

REFERENCES