An executable semantics of flexible plans in terms of timed game automata

Marta Cialdea Mayer*and Andrea Orlandini†

This is a draft version of a paper appearing in the Proceedings of the 22nd International Symposium on Temporal Representation and Reasoning (TIME 2015). It should not be cited, quoted or reproduced.

Abstract

Robust plan execution in uncertain and dynamic environments is a critical issue for plan-based autonomous systems, especially when uncertain events coexist with temporal flexibility. In this regard, many Planning and Scheduling systems model temporal uncertainty by means of flexible timelines, each of which describes the behavior of one of the system components, and consists of a sequence of events whose begin and end times range within given intervals.

This work enriches a previously proposed formal characterization of flexible timelines and plans, considering the difference between controllable and uncontrollable activities. Two main sources of uncertainty are considered: i) some components of the system may depend on an external environment and cannot be planned by the executive; ii) there may be tasks whose duration cannot be exactly foreseen in advance. Such notions are formally defined and the consequent controllability issues are addressed, focusing, in particular, on dynamic controllability.

Partially controllable flexible plans are given a semantics in terms of networks of timed game automata (TGA), showing how they can be encoded into such networks. The translation allows for exploiting existing verification tools for TGA, such as UPPAAL-TIGA, in order to check the dynamic controllability property for flexible plans and, possibly, generate a dynamic execution strategy that can be used for robust plan execution. Some preliminary experiments aimed at evaluating the feasibility of the approach are also presented.

1 Introduction

Robust plan execution in uncertain and dynamic environments is a critical issue for plan-based autonomous systems (see [21] and many others). Once a planner

*Dipart. Informatica e Automazione, Università di Roma Tre, Via della Vasca Navale 79, 00144, Roma (Italia). Email: cialdea@ing.uniroma3.it

†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

has generated a temporal plan, it is up to the executive system to decide, at run-time, how and when to execute each planned activity preserving both plan consistency and controllability. Such a capability is even more crucial when the generated plan is temporally flexible, since a flexible temporal plan is only partially specified. In fact, a flexible plan captures an envelope of potential behaviors to be instantiated during the execution taking into account temporal/causal constraints and controllable/uncontrollable activities and events[1]. Among different 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 [27, 20, 8]. Timeline-based planning has been introduced in [27], under a modeling assumption inspired by classical control theory. It pursues the general idea that P&S for controlling complex physical systems consists of the synthesis of desired temporal behaviors (or *timelines*), each of which corresponds to one of the system components (*state variables*). The evolution of the features is described by some causal laws and limited by domain constraints. In general, plans synthesized by 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, and aimed at facing uncertainty during actual execution. In this regard, they can be exploited by an executive system for robust on-line execution.

This work carries on the work started in [12], where a formal account of flexible timelines and plans is given, without however addressing controllability issues. This paper refines and extends the definition of flexible plans, introducing quantitative temporal relations as well as taking into account the difference between controllable and uncontrollable activities. In this respect, it addresses, in particular, the *dynamic controllability* issue (see for instance [25]). Beyond formally defining the related main notions for flexible plans, a formal semantics of flexible plans is given in terms of *Timed Game Automata* (TGA) [22]. TGA allow one to model real-time systems and controllability problems representing uncontrollable activities as *adversary moves* within a game between the controller and the environment. Following the same approach presented in [9, 29], a verification tool, UPPAAL-TIGA [5], is exploited to verify whether a flexible plan is dynamically controllable and to generate a dynamic execution strategy by solving a *reachability game*. It is worth pointing out that the encoding of flexible plans into networks of TGA, allowing one to synthesize execution strategies that do not require run-time reasoning, is linear in the size of the plan. In order to evaluate the feasibility of the proposed approach, some preliminary empirical results are presented, considering a benchmark domain introduced in [9].

Previous proposals have tackled the issue of robust plan execution within the Constraint-based Temporal Planning (CBTP) framework, deploying specialized techniques based on temporal-constraint networks [31, 26, 19, 24]. Controllability issues have been formalized and investigated for Simple Temporal Problems with Uncertainty (STPU) in [31], where basic formal notions are given to the aim of properly defining *dynamic* controllability. Several authors (e.g. [19, 24]) have proposed a *dispatchable execution* approach, where a flexible temporal plan is used by a controller that schedules activities on-line while guaranteeing con-

[1] *Uncontrollable* events are those that cannot be planned for as they are decided by Nature – the external environment.

straint satisfaction. Recently, the use of TGA has been considered to address dynamic controllability in temporal networks with uncertainty [14] also in the case of disjunctive problems [13]. Since CBTP systems often rely on temporal networks for their solving process, such work could be indirectly exploited to furnish a mapping from flexible plans to TGA, and consequently give a methodology for synthesizing execution strategies for flexible plans. However, representing a flexible timeline-based plan as a STNU entails a sort of simplification of the associated actual plan structure causing a lost of information on the "dependencies" among its components which can usefully be taken into account. As a consequence, we consider it important to define controllability notions directly for flexible plans, abstracting away from the concrete representation they can be internally given, and to give a general methodology which allows one to map plans into networks of TGA and exploit such encoding for control synthesis.

A comprehensive and semantically well founded planning framework is provided in [15], that includes temporal uncertainty in timeline-based approaches. However, it does not address time flexibility and focuses only on strong controllability. Finally, the above notions have also been extended to a timeline-based framework [9], following an approach similar to what presented here, i.e. based on model checking with TGA to check flexible plans against dynamic controllability and to generate a robust plan controller able to execute flexible timeline-based plans [29, 28]. The present work advances the above work on different perspectives: a more comprehensive approach is presented here, since controllability information is included in the description of the plan itself, thus avoiding the need of considering additional information derived from the specific execution contexts. Moreover, the encoding into TGA does not require to consider also the specification of the planning domain (like in the previously cited works), thus allowing for a more compact and straightforward translation of plans in terms of TGA. In summary, the whole information needed to encode and control the flexible plan is contained in its description. Finally, the present methodology allows also for the encoding of partially specified plans.

Plan of the paper: Section 2 gives a formalization of flexible plans, and the associated controllability issues are addressed in Section 3. A sketchy background on TGA is given in Section 4, and the encoding of flexible plans in terms of networks of TGA is described in Section 5. An empirical evaluation performed on a benchmark domain derived from a real word application is presented and discussed in Section 6. Finally, some remarks conclude the paper in Section 7.

2 Flexible plans

This section briefly summarizes the definition of flexible timelines and plans given in [12] (with some minor differences), enriching it with the distinction between controllable and uncontrollable tasks. The first mandatory requirement a flexible plan must satisfy is to be valid with respect to the underlying planning domain. In particular, it must satisfy the *synchronization rules* of the domain, constraining the behavior of some components in relation to others. However, differently from [9, 29], plans are defined so that all necessary information is embedded in the plans themselves, with no need to "look outside" (i.e. at the domain specifications) in order to check whether a particular scheduling of a flexible plan satisfies the domain contraints. Therefore, this presentation focuses

on flexible plans, abstracting away from their relation to planning domains. The reader is referred to [12] for the structure of planning domains and the notion of plan validity.

This work considers two sources of uncertainty. On one hand, the evolution of some components of the system may be completely outside the control of the executive; such components are modeled by means of *external* state variables, and what the planner and the executive know about them is only what is specified in the underlying planning problem. The distinction between external and *planned* state variables is part of the description of the planning domain. On the other hand, some events may be only partially controllable: the system can decide when to start an activity, but is not allowed to fix its duration exactly. When the duration of a value cannot be controlled, it is tagged as *uncontrollable*, and what the planner and the executive may assume is only that its duration is included within given lower and upper bounds.

Apart from the distinction between external and planned variables, the essential source of uncertainty relevant to the present work is therefore due to activities whose duration cannot be controlled exactly by the executive. In fact, an external state variable is simply a variable whose values are all uncontrollable, in the above sense.

For the sake of generality, temporal instants and durations are taken from an infinite set of non negative numbers $\mathbb{T}$, including 0. Sometimes, ∞ is given as an upper bound to allowed numeric values, with the meaning that $t < \infty$ for every $t \in \mathbb{T}$. The notation $\mathbb{T}^\infty$ will be used to denote $\mathbb{T} \cup \{\infty\}$.

2.1 Timelines

When planning with timelines,[2] time flexibility is taken into account by allowing that the durations of valued intervals, called *tokens*, range within given bounds. The main component of a flexible plan is a set of timelines, and it represents a whole set of *scheduled timelines*, i.e. timelines whose tokens have a fixed duration (within the allowed bounds). In order to guarantee that every scheduled timeline represented by a given flexible plan Π is valid w.r.t. the underlying planning domain, the plan is equipped with additional information about the temporal relations that have to hold in order to satisfy the synchronization rules of the domain. Relations are presented in Section 2.1.2, while this section is devoted to properly define the notions of timeline, schedule and flexible plan.

Definition 1. *A token is a tuple of the form:* $(v, [e, e'], [d, d'], \gamma)$*, where* v *is called the value of the token,* $e, e', d, d' \in \mathbb{T}$*,* $e \leq e'$*,* $d \leq d'$*, and* $\gamma \in \{c, u\}$ *is the* controllability tag *of the token. If* $\gamma = c$*, then the token is* controllable *and if* $\gamma = u$*, then it is an* uncontrollable *token.*

A timeline FTL_x *for the state variable* x *in the temporal horizon* H *is a finite sequence of tokens:*

$$x^1, \ldots, x^k = \\ (v_1, [e_1, e'_1], [d_1, d'_1], \gamma_1), \ldots, (v_k, [e_k, e'_k], [d_k, d'_k], \gamma_k)$$

where $e_k = e'_k = H$*, and for all* $i = 1 \ldots k-1$*,* $e'_i \leq e_{i+1}$*.*

[2] In this work, "timeline" refers to what in [12] is called flexible timeline, while non-flexible timelines are called "scheduled timelines". The latter are defined as particular cases of (flexible) timelines.

If $x^i = (v, [e, e'], [d, d'], \gamma)$ *is a token in the timeline* FTL_x*, the following notations will be used:* end_time$(x^i) = [e, e']$*;* start_time$(x^1) = [0, 0]$ *and* start_time$(x^{i+1}) =$ end_time(x^i)*;* duration$(x^i) = [d, d']$.

Intuitively, a token x^i represents the set of valued intervals with start times in start_time(x^i), end times in end_time(x^i) and whose durations are in the range duration(x^i) (see Definition 2 below). It is worth pointing out that in [12] (and often in the literature), a flexible token contains, beyond its value, end and duration intervals, also a *start* interval. However, once a token x^i is embedded in a timeline, the time interval to which its start point belongs (start_time(x^i)) can easily be computed like shown in the definition above. Thus, including it as part of the token itself is redundant.

When considering sets **FTL** of timelines, it is always assumed that they have the same temporal horizon H, that is also called the horizon of **FTL**.

2.1.1 Schedules

The next definition introduces the notion of *schedule* of a timeline. Tokens, timelines and sets of timelines represent the set of their schedules. In general, TL_x and **TL** will be used as meta-variables for scheduled timelines and sets of scheduled timelines, respectively, while FTL_x and **FTL** as meta-variables for generic (flexible) timelines and sets of timelines. The schedule of a token corresponds to one of the valued intervals it represents, i.e. it is obtained by choosing an exact end point in the allowed interval. A scheduled timeline is a sequence of scheduled tokens satisfying the duration requirements. In what follows, an interval of the form $[t, t]$, consisting of a single time point, will be identified with the time point t (and, with an abuse of notation, singleton intervals are allowed as operands of additions, subtractions, comparison operators, etc.).

Definition 2. *A* scheduled *token is a token of the form* $(v, [t, t], [d, d'], \gamma)$ *– or succintly* $(v, t, [d, d'], \gamma)$*. A* schedule *of a token* $x^i = (v, [e, e'], [d, d'], \gamma)$ *is a scheduled token* $(v, t, [d, d'], \gamma)$*, where* $e \leq t \leq e'$.

A scheduled timeline TL_x *is a timeline consisting only of scheduled tokens and such that, if* k *is the length of the timeline (i.e. the number of tokens in* TL_x*):*

1. *for all* $1 \leq i \leq k-1$*, if* $[d_i, d'_i] =$ duration(x^i)*, then* $d_i \leq$ end_time$(x^i) -$ start_time$(x^i) \leq d'_i$*;*
2. *if* x *is a planned variable, then also* $d_k \leq$ end_time$(x^k) -$ start_time$(x^k) \leq d'_k$*, where* $[d_k, d'_k] =$ duration(x^k).
3. *if* x *is an external variable, then* end_time$(x^k) -$ start_time$(x^k) \leq d'_k$*, where* $[d_k, d'_k] =$ duration(x^k).

A scheduled timeline TL_x *for the state variable* x *is a schedule of* FTL_x *if* TL_x *and* FTL_x *have the same length* k*, and for all* i*,* $1 \leq i \leq k$*, the token* x^i *of* TL_x *is a schedule of the token* x^i *of* FTL_x.

Let **FTL** *be a set of timelines for the state variables in the set* SV*. A schedule* **TL** *of* **FTL** *is a set of scheduled timelines for the state variables in* SV*, where each* $TL_x \in$ **TL** *is a schedule of the timeline* $FTL_x \in$ **FTL**.

The two conditions 1 and 2 in the above definition require that the actual duration of scheduled tokens enforce the corresponding duration requirements, except for the last token of the timeline for an external variable. In that case, it is only required that the duration of the scheduled token does not exceed the allowed duration (condition 3). As a matter of fact, the valued intervals represented by the token at the very end of a flexible timeline for an external variable are allowed to violate the minimal duration requirement and to have an actual end point which goes beyond the horizon. The reason is that what is only observed by the system could also continue after the temporal horizon (that places a cut on the observed evolution of the uncontrollable part of the world). On the contrary, the actual durations of the valued intervals represented by any other token x^i (including the last ones of planned variables, which may model the accomplishment of a planning goal) must be in the interval duration(x^i).

2.1.2 Relations

In a planning domain, some constraints may be set on the temporal evolution of the system components. Like in [12], a flexible plan contains all the information needed for its execution, therefore, in particular, it may contain a set of temporal constraints that the tokens in its timelines are required to enforce.

In this work, quantitative temporal constraints are considered (thus extending [12]) and, for the sake of simplicity, a small set of primitive relations is chosen, all of which are parametrized by a (single) temporal interval. In what follows, if $b, e \in \mathbb{T}$ and $b < e$, the time interval $[b, e]$ denotes the set of time points $\{t \mid b \leq t \leq e\}$.

Definition 3. *A* temporal relation between intervals *is an expression of the form* $A\ r_{[lb,ub]} B$*, where* $A = [b_A, e_A]$ *and* $B = [b_B, e_B]$ *are time intervals, with* $b_A, e_A, b_B, e_B \in \mathbb{T}$*,* $lb \in \mathbb{T}$*,* $ub \in \mathbb{T}^\infty$*, and* $r \in \mathbf{R} = \{$start_before_start, end_before_end, start_before_end, end_before_start$\}$*. The following table defines when a relation* $A\ r_{[lb,ub]} B$ *holds:*

the relation	*holds if*
A start_before_start$_{[lb,ub]}$ B	$lb \leq b_B - b_A \leq ub$
A end_before_end$_{[lb,ub]}$ B	$lb \leq e_B - e_A \leq ub$
A start_before_end$_{[lb,ub]}$ B	$lb \leq e_B - b_A \leq ub$
A end_before_start$_{[lb,ub]}$ B	$lb \leq b_B - e_A \leq ub$

A temporal relation between an interval and a timepoint *is an expression of the form* $A\ r_{[lb,ub]} t$*, where* $A = [b, e]$ *is a time interval, with* $b, e \in \mathbb{T}$*,* $r \in \mathbf{R}' = \{$starts_before, starts_after, ends_before, ends_after$\}$*,* $t, lb \in \mathbb{T}$ *and* $ub \in \mathbb{T}^\infty$*. The following table defines when a relation* $A\ r_{[lb,ub]} t$ *holds:*

the relation	*holds if*
A starts_before$_{[lb,ub]}$ t	$lb \leq t - b \leq ub$
A starts_after$_{[lb,ub]}$ t	$lb \leq b - t \leq ub$
A ends_before$_{[lb,ub]}$ t	$lb \leq t - e \leq ub$
A ends_after$_{[lb,ub]}$ t	$lb \leq e - t \leq ub$

Other constraints, such as those used by systems like EUROPA [2] and APSI-TRF [6], can be easily defined in terms of the primitive ones. For instance,

A contains $[lb_1, ub_1][lb_2, ub_2]\, B$ can be defined as the conjunction of the relations

$$A \text{ start\_before\_start } [lb_1, ub_1]\, B \text{ and}$$
$$B \text{ end\_before\_end } [lb_2, ub_2]\, A$$

Similarly, A overlaps $[lb_1, ub_1][lb_2, ub_2]\, B$ is equivalent to the conjunction of

$$A \text{ start\_before\_start } [lb_1, ub_1]\, B,$$
$$A \text{ end\_before\_end } [lb_2, ub_2]\, B \text{ and}$$
$$B \text{ start\_before\_end } [0, \infty]\, A$$

A table with the definition of the most commonly used quantitative temporal relations can be found in [11].

Definition 4. *Let $t, lb \in \mathbb{T}$, $ub \in \mathbb{T}^\infty$, and x^i and y^j be scheduled tokens, with* start_time$(x^i) = b_i$, end_time$(x^i) = e_i$, start_time$(y_j) = b_j$, end_time$(y_j) = e_j$. *Expressions of the form $x^i\ r_{[lb,ub]}\ y^j$, for $r \in \mathbf{R}$, and $x^i\ r_{[lb,ub]}\ t$, for $r \in \mathbf{R}'$, are called* relations on tokens. *The relation $x^i\ r_{[lb,ub]}\ y^j$ holds iff $[b_i, e_i]\ r_{[lb,ub]}\ [b_j, e_j]$ holds. And the relation $x^i\ r_{[lb,ub]}\ t$ holds iff $[b_i, e_i]\ r_{[lb,ub]} t$ holds. When a relation on tokens holds we also say that the tokens whose names occur in the relation satisfy it, and that any set of scheduled timelines containing such tokens satisfies the relation.*

2.1.3 Flexible Plans

A flexible plan is made up by a set of timelines and a set of relations:

Definition 5. *A* flexible plan Π *over the horizon H is a pair $(\mathbf{FTL}, \mathcal{R})$, where $\mathbf{FTL}$ is a set of timelines over the same horizon H and $\mathcal{R}$ is a set of relations on tokens, involving token identifiers in some timelines in $\mathbf{FTL}$.*

An instance *of the flexible plan $\Pi = (\mathbf{FTL}, \mathcal{R})$ is any scheduling of $\mathbf{FTL}$ that satisfies every relation in $\mathcal{R}$.*

Since external state variables are not under the system control, a well defined planning problem must include information about their behavior. Such information is given in the form of a set $\mathbf{FTL}_E$ of flexible timelines and, when considering a solution plan $\Pi = (\mathbf{FTL}, \mathcal{R})$, we assume that $\mathbf{FTL}_E \subseteq \mathbf{FTL}$. Moreover, obviously, the strucure of the timelines in $\mathbf{FTL}$ and the relations in $\mathcal{R}$ must ensure, not only that the plan obeys the rules of the underlying planning domain, but also that the planning goals are satisfied. Goal satisfaction can however be reduced to a set of relations on the tokens of the timelines in the plan (see [12] for details). The relations $\mathcal{R}$ in a solution plan are consequently assumed to include those which represent goal satisfaction.

3 Controllability Properties of Flexible Plans

This section proposes a definition of the various notions of plan controllability, very much in the style of similar work on Simple Temporal Network with Uncertainty (STNU), such as [18, 31, 26]. There is an intuitively obvious correspondence between a flexible plan and a SNTU, and, in fact, controllability issues have been addressed for plans given in that form. In this section the

same concepts are defined directly for flexible plans, as defined in Section 2, independently from how they are represented. To the best of our knowledge, the definition of a formal equivalence between flexible temporal plans and STNUs is still an open issue (and its demonstration is out the scope of this paper). Nevertheless, as a guideline to understand the correspondence between plans and SNTU, it can be observed that token end points correspond to nodes in the network, while token durations and the temporal constraints given in the sets of relations $\mathcal{R}$ correspond to network edges. Durations of uncontrollable tokens and relations on the tokens of external state variables correspond to what are usually called *contingent links* in a STNU.

Once a flexible plan $\Pi = (\mathbf{FTL}, \mathcal{R})$ is built, controllability tags are the important features to be taken into consideration when facing the controllability problem. In what follows, if $\mathbf{FTL}$ is a set of timelines, tokens($\mathbf{FTL}$) denotes the set of all the tokens making up the timelines in $\mathbf{FTL}$, $\text{tokens}_C(\mathbf{FTL})$ is the set of controllable tokens occurring in some timeline in $\mathbf{FTL}$, and $\text{tokens}_U(\mathbf{FTL})$ contains the uncontrollable tokens of $\mathbf{FTL}$.

The notion of *situation*, introduced below, copes with the temporal uncertainty represented by the uncontrollable tokens of a set of timelines $\mathbf{FTL}$. A situation is a function assigning a (legal) value to the duration of each uncontrollable token. The set of situations defined over a set $\mathbf{FTL}$ of timelines represents all the associated uncontrollable temporal evolutions.

Definition 6. *Let* $\mathbf{FTL}$ *be a set of timelines. A* situation *for* $\mathbf{FTL}$ *is a total function*

$$\omega : \text{tokens}_U(\mathbf{FTL}) \rightarrow \mathbb{T}$$

such that if $x^i \in \text{tokens}_U(\mathbf{FTL})$ *and* $\text{duration}(x^i) = [d, d']$*, then* $d \leq \omega(x^i) \leq d'$.

The set of all the situations for $\mathbf{FTL}$ *is called the* space of situations *for* $\mathbf{FTL}$ *and is denoted by* $\Omega_{\mathbf{FTL}}$.

Every situation ω for $\mathbf{FTL}$ induces a set of timelines where the duration of every uncontrollable token x^i in $\mathbf{FTL}$ is replaced by the (singleton) value $\omega(x^i)$. The so obtained set of timelines is called a *projection* of $\mathbf{FTL}$: in a projection, the duration of each uncontrollable token is fixed. Intuitively, a projection corresponds to one of the possible combinations of uncontrollable behaviors in $\mathbf{FTL}$.

Definition 7. *Let* FTL_x *be a timeline in the set* $\mathbf{FTL}$ *and* ω *a situation for* $\mathbf{FTL}$. *The* projection $\omega(FTL_x)$ *is the timeline obtained from* FTL_x *by replacing the duration of every uncontrollable token* x^i *with* $[\omega(x^i), \omega(x^i)]$. *The projection* $\omega(\mathbf{FTL})$ *is the set* $\{\omega(FTL_x) \mid FTL_x \in \mathbf{FTL}\}$.

In other terms, if x^i is an uncontrollable token of the form $(v, [e, e'], [d, d'], u)$, then it is replaced in $\omega(\mathbf{FTL})$ by $(v, [e, e'], \omega(x^i), u)$; controllable tokens are left unchanged. It is worth observing that if $(\mathbf{FTL}, \mathcal{R})$ is a flexible plan, then $(\omega(\mathbf{FTL}), \mathcal{R})$ is a flexible plan too.

Obviously, there is a one-to-one correspondence between situations for a given set of timelines and its projections. Analogously, the set of schedules of a given set $\mathbf{FTL}$ of timelines bears a one-to-one correspondence with the set of functions assigning a fixed value to each token end time. Such functions are called *scheduling functions* and are defined below.

Definition 8. *Let* **FTL** *be a set of timelines. A* scheduling function *for* **FTL** *is a function* $\theta : \text{tokens}(\mathbf{FTL}) \to \mathbb{T}$. *The set of all the scheduling functions for* **FTL** *is denoted by* $\mathcal{T}_{\mathbf{FTL}}$.

A scheduling function θ *induces the set* $\mathbf{TL}_\theta$ *of scheduled timelines obtained from* **FTL** *by replacing the end time of each token* $x^i \in \text{tokens}(\mathbf{FTL})$ *with* $[\theta(x^i), \theta(x^i)]$.

Let $\Pi = (\mathbf{FTL}, \mathcal{R})$ *be a flexible plan. A scheduling function* θ *for* **FTL** *is* consistent *with* Π *iff the set* $\mathbf{TL}_\theta$ *of scheduled timelines induced by* θ *is an instance of* Π.

Intuitively, a scheduling function that is consistent with the plan $\Pi = (\mathbf{FTL}, \mathcal{R})$ induces a set of scheduled timelines that satisfy all the duration requirements in **FTL** and all the relations in $\mathcal{R}$.

It is worth noticing that, while a situation fixes token durations (and maybe only indirectly their end times), a scheduling function assigns values to token end times. Moreover, situations are defined only on uncontrollable tokens, while scheduling functions are defined for all the tokens in the set of timelines.

Execution strategies are defined next. An execution strategy for a given plan Π maps every situation to a scheduling function: once the duration of the uncontrollable tokens is known, the strategy decides how to schedule all the token end points.

Definition 9. *If* $\Pi = (\mathbf{FTL}, \mathcal{R})$ *is a flexible plan, an* execution strategy *for* Π *is a mapping* $\sigma : \Omega_{\mathbf{FTL}} \to \mathcal{T}_{\mathbf{FTL}}$.

The execution strategy σ *is* viable *if for each situation* $\omega \in \Omega_{\mathbf{FTL}}$, *the scheduling function* $\sigma(\omega)$ *is consistent with the plan* $(\omega(\mathbf{FTL}), \mathcal{R})$.

A viable strategy for the plan $(\mathbf{FTL}, \mathcal{R})$ maps each situation ω to a scheduling function inducing a set of scheduled timelines which respects the duration constraints in $\omega(\mathbf{FTL})$ – i.e. the bounds on token durations established by **FTL** and the exact durations of uncontrollable tokens given by ω – and satisfies the relations in $\mathcal{R}$.

In order to define dynamic execution strategies, i.e. strategies which are able to schedule a given event only on the base of what happened before, partial situations must be considered.

Definition 10. *Let* **FTL** *be a set of timelines,* x^i *a token in* $\text{tokens}(\mathbf{FTL})$ *and* θ *a scheduling function for* **FTL**. *The* prehistory of x^i w.r.t. θ *is the partial function* $\theta_{\prec x^i} : \text{tokens}_U(\mathbf{FTL}) \to \mathbb{T}$ *such that:*

$$\theta_{\prec x^i}(y^j) = \begin{cases} \theta(y^j) & \text{if } \theta(y^j) < \theta(x^i) \text{ and } j = 1 \\ \theta(y^j) - \theta(y^{j-1}) & \text{if } \theta(y^j) < \theta(x^i) \text{ and } j > 1 \\ \text{undefined} & \text{if } \theta(y^j) \geq \theta(x^i) \end{cases}$$

The prehistory $\theta_{\prec x^i}$ is a partial situation that is defined only for the uncontrollable tokens y^j such that $\theta(y^j) < \theta(x^i)$. When $\theta_{\prec x^i}(y^j)$ is defined, its value is the (exact) duration of the token y^i in the timelines induced by θ: $\theta_{\prec x^i}(y^j) = \theta(y^j) - \theta(y^{j-1})$, except when $j = 1$ (y^j is the first token of a timeline), where $\theta_{\prec x^i}(y^1) = \theta(y^1)$. Basically, a prehistory defines a partial projection of **FTL** fixing the duration of uncontrollable tokens occurring before x^i according to θ.

Note that, despite the notation used for prehistories, $\theta_{\prec x^i}$ is not a (partial) scheduling function, but a partial situation: it assigns values to the durations of uncontrollable tokens.

Definition 11. *If* $\Pi = (\mathbf{FTL}, \mathcal{R})$ *is a flexible plan, a* dynamic execution strategy *(DES) for* Π *is an execution strategy* σ *for* Π *such that, for all situations* $\omega, \omega' \in \Omega_{\mathbf{FTL}}$ *and every controllable token* $x^i \in \text{tokens}_C(\mathbf{FTL})$*, if* $\sigma(\omega) = \theta$ *and* $\sigma(\omega') = \theta'$*, then*

$$\theta_{\prec x^i} = \theta'_{\prec x^i} \text{ implies } \theta(x^i) = \theta'(x^i)$$

Finally, the controllability properties considered for STNUs can be defined on flexible plans.

Definition 12. *Let* $\Pi = (\mathbf{FTL}, \mathcal{R})$ *be a flexible plan for the planning problem* $\mathcal{P}$*. The plan* Π *is* weakly controllable *if there is a viable execution strategy for* Π*.*

The plan Π *is* strongly controllable *if there is a viable execution strategy* σ *such that, for all situations* $\omega, \omega' \in \Omega_{\mathbf{FTL}}$*, if* $\sigma(\omega) = \theta$ *and* $\sigma(\omega') = \theta'$*, then for every controllable token* $x^i \in \text{tokens}_C(\mathbf{FTL})$*:* $\theta(x^i) = \theta'(x^i)$

The flexible plan Π *is* dynamically controllable *if there exists a viable DES for* Π*.*

The definition above characterize a flexible plan with respect to its executability. In simple terms, if the executor of a weakly controllable plan can know in advance how the uncontrollable events will evolve (i.e. the complete situation ω), it can safely adopt the decisions induced by the scheduling function associated to ω. If the events turn out to be like modeled by ω, such decisions lead to a successful execution of the plan. But in case actual events evolve differently from ω, the executor might be unable do adapt its strategy to the new situation. Therefore, the higher the level of uncertainty in the plan, the higher is the probability to fail while executing it. When, on the contrary, a plan is strongly controllable, its executor is on a safe side: whichever the uncontrollable events turn out to be, it can take the same decisions (posted by a fixed scheduling function) to face the situation and successfully complete the execution of the plan. Unfortunately, few plans are strongly controllable, especially in highly dynamic domains. Finally, when a plan is dynamically controllable, its executor has to monitor what is happening in the world step by step, and decide what to do accordingly. However, it is always sure to be able to adapt its schedules: at each step, whatever happened in the past, there is a decision that can be taken for the next controllable event, in such a way that the plan will at the end be executed successfully. As a consequence, the desired goals can be achieved for any possible turnout of uncontrollable events. Dynamic controllability constitutes a highly desirable property for a flexible plan Π. As a matter of fact, the associated viable DES can be exploited to endow a timeline-based control architecture ensuring robust plan execution (see for instance [28]). Although weak and strong controllability have been defined above for the sake of completeness, this paper focuses only on dynamic controllability.

4 Timed Game Automata

This section is devoted to an informal presentation of Timed Game Automata (TGA), as they are implemented in the UPPAAL-TIGA system, a well known model checking tool which is able to solve games based on TGA with respect to reachability and safety properties [4, 16]. The formalism of UPPAAL-TIGA is based on [22], with some useful extensions. Here, attention is restricted to those features that are used to model flexible plans.

A timed automaton (TA) [1, 3] is an automaton with a set of real-valued variables called *clocks*. Clocks are initialized with zero when the system is started, and then increased synchronously with the same rate. The transitions of the automaton can reset some clocks and may be constrained by clock values: a transition may be taken only if the current values of the clocks satisfy the associated constraints. TGA extend timed automata by partitioning transitions into *controllable* and *uncontrollable* ones.

If C is a finite set of clocks, the set $\mathcal{B}(C)$ denotes the set of constraints φ generated by the grammar $\varphi ::= x \sim c \mid x - y \sim c \mid \varphi \wedge \varphi$, where $c \in \mathbb{Z}$, $x, y \in C$, and $\sim \in \{<, \leq, =, \geq, >\}$.

A TGA $\mathcal{A}$ is defined by a finite set L of *locations*, the initial location $l_0 \in L$, a finite set Σ of *actions* (split into two disjoint sets, Σ_c the set of *controllable* actions and Σ_u the set of *uncontrollable* ones), a finite set C of clocks and a finite set of transitions. Locations can be labeled by *invariants*: if $l \in L$, $\mathrm{Inv}(l) \in \mathcal{B}(C)$ is the invariant of l, that, intuitively, represents a constraint that must be satisfied by the runs of the automaton while staying in location l. A *transition* connects two locations. A transition is labelled by an action and can be then either *controllable* or *uncontrollable* according to the action type (i.e. either in Σ_c or Σ_u). A transition is possibly labelled also by a *guard* and clock *updates*. The guard of a transition is an element of $\mathcal{B}(C)$ and represents a constraint that must hold when a run takes the transition. An update affecting the clock x has a side-effect: when a run takes the transition, the clock x is reset to 0.[3]

A *run* of a timed automaton $\mathcal{A}$ is a sequence of transitions between *states*, where a state consists of the current location and the current clock values, represented by a *clock assigment*, i.e. a function u mapping clocks to non-negative reals. From a state $\langle l, u \rangle$, a run ρ of $\mathcal{A}$ can take either a *time transition* or a *discrete transition*. When ρ takes a time transition, it just let time progress, i.e. the values of all the clocks increase of the same amount δ. However, when a run stays in the location l the values of the clocks (both the old and the increased ones) must satisfy the invariants of l. If there is a transition in $\mathcal{A}$ from the location l to l', the run can take a discrete transition from the state $\langle l, u \rangle$ to $\langle l', u' \rangle$, thus reaching the new location l'. The clock assignment u' assigns every clock x the value $u(x)$, except for those that are reset by the transition. The discrete transition from $\langle l, u \rangle$ to $\langle l', u' \rangle$ can be taken only if u satisfies the guard of the transition from l to l' in $\mathcal{A}$ and the new clock assignment u' satisfy the invariant of l'. A run of the automaton is a finite or infinite sequence of alternating time and discrete transitions.

In what follows, attention will be restricted to runs starting from the *initial state* $\langle l_0, \vec{0} \rangle$, where l_0 is the initial location of $\mathcal{A}$ and $\vec{0}$ assigns 0 to every clock.

[3] UPPAAL-TIGA allows clocks to be set also to non-zero values.

If ρ is one of such runs, and t_i is the time elapsed since the run has been started when ρ takes a discrete transition, thus passing from a state $\langle l, u\rangle$ to $\langle l', u'\rangle$ (i.e. t_i is the sum of the increases of the clocks in the time transitions taken before), then we say that ρ takes the discrete transition at time t_i, exiting the location l and entering l'.

A *network of TGA* (nTGA) is a finite set $\mathcal{N}$ of TGA (that may share some *global* clocks), evolving in parallel with a CCS style semantics for parallelism [23]. Essentially, the parallel composition of a set of automata is the product of the automata and a run of a nTGA consists of the parallel runs of the automata in the network.

TGA can be used to model two-player games between an agent (the controller), controlling the controllable transitions, and the environment, that controls the uncontrollable ones. In a *pure reachability game* the agent's goal is to move the nTGA from the initial state *Init* to a state satisfying a given winning condition *Goal*. The game consists in finding a *strategy* f such that the nTGA starting from *Init* and supervised by f generates winning runs, i.e. runs that finally reach a state satisfying *Goal*. A strategy tells the controller when to take the controllable transitions that will guarantee that the system, regardless of when and if the opponent chooses to take uncontrollable transitions, will eventually end up in a state satisfying *Goal*. In a given state, the strategy can suggest the controller to either do a particular controllable action or do nothing at this point in time, just wait (only so-called *memoryless* strategies need be considered here).

A pure reachability game is therefore given by defining, beyond the nTGA, the winning condition *Goal*. If the game can be won, UPPAAL-TIGA generates a winning strategy for the game. Otherwise, it can be asked to generate a *counter-strategy*, i.e. a strategy for the opponent that would lead the controller to loose the game.

5 Encoding of flexible plans as networks of timed game automata

This section is devoted to give a short description of the encoding of plans as networks of TGA. Let $\Pi = \langle \mathbf{FTL}, \mathcal{R}\rangle$ be a flexible plan and $\mathcal{N}$ the network of automata modeling Π. The encoding establishes an injective mapping μ from tokens of **FTL** to locations of automata in $\mathcal{N}$, such that for every token x^i, the location $\mu(x^i)$ has exactly one incoming *edge* (transition) and one outgoing edge. A correspondence between runs of the automata and scheduled timelines can be established via the mapping μ, according to the following definition.

Definition 13. *If ρ is a run of the nTGA $\mathcal{N}$ modeling $\Pi = \langle \mathbf{FTL}, \mathcal{R}\rangle$ and* **TL** *is an instance of* **FTL**, *then ρ corresponds to* **TL**, *and vice-versa, if, for every token x^i in* **TL**, *ρ enters $\mu(x^i)$ at time $t = \text{start\_time}(x^i)$ and exits $\mu(x^i)$ at time $t = \text{end\_time}(x^i)$.*

The encoding can be shown to be correct and complete:

Theorem 1. *Let $\Pi = \langle \mathbf{FTL}, \mathcal{R}\rangle$ be a flexible plan, and let $\mathcal{N}$ be the nTGA modeling Π. Then every instance of Π corresponds to a run of $\mathcal{N}$, and every run of $\mathcal{N}$ corresponds to an instance of Π.*

Moreover, for every token x^i of **FTL**, *x^i is uncontrollable if and only if the transition exiting from $\mu(x^i)$ is uncontrollable.*

The UPPAAL-TIGA winning conditions specified in the sequel define the reachability game where every run of the nTGA encoding the plan Π reaches the state corresponding to end of the final token of every timeline of Π. Therefore, an UPPAAL-TIGA winning strategy is a viable DES for the plan encoded by the nTGA.

A detailed description of the encoding can be found in [11], where arguments are also given, step by step, showing that Theorem 1 holds. The encoding allows also for modeling "partial plans", where timelines may have undefined temporal slots, i.e. tokens without value and possibly with no associated duration. This allows for the UPPAAL-TIGA tool being involved also during the plan construction phase, thus interacting with the planner and possibly suggesting to abandon on search routes that would lead to plans that cannot be executed (like in, e.g., CIRCA [17]). This feature is however not described here.

If $\Pi = \langle \mathbf{FTL}, \mathcal{R} \rangle$ is the plan to be encoded and $\mathcal{N}$ is the nTGA modeling Π, each automaton in $\mathcal{N}$ models a timeline in **FTL**. The main properties of the TGA $\mathcal{A}_x$ modeling the timeline FTL_x are:

1. the states of $\mathcal{A}_x$ are $\{start, finish\} \cup \{\mu(x^i) \mid x^i \text{ is a token of } FTL_x\}$;
2. the initial state of $\mathcal{A}_x$ is *start*;
3. every state of $\mathcal{A}_x$ has exactly one incoming edge, except for *start*, that has none;
4. every state of $\mathcal{A}_x$ has exactly one outgoing edge, except for $finish$, that has none;
5. there is an edge in $\mathcal{A}_x$ from *start* to $\mu(x^1)$, where x^1 is the first token of FTL and there is an edge in $\mathcal{A}_x$ from $\mu(x^k)$ to $finish$, where x^k is the last token of FTL;
6. for every pair of consecutive tokens x^i and x^{i+1} in FTL, there is an edge from $\mu(x^i)$ to $\mu(x^{i+1})$ in $\mathcal{A}_x$. Moreover, if x^i is uncontrollable (which holds, in particular, if x is an external variable), then the transition from $\mu(x^i)$ to $\mu(x^{i+1})$ is uncontrollable, too.

Therefore, every automaton has a "final state", $finish$, and the pure reachability game is specified by a goal of the form $\mathcal{A}_{x_1}.finish \wedge \cdots \wedge \mathcal{A}_{x_k}.finish$, where $\mathcal{N} = \{\mathcal{A}_{x_1}, \ldots, \mathcal{A}_{x_k}\}$ whichever the behaviour of uncontrollable components, the automata in $\mathcal{N}$ can reach their final states (at the same time).

Clocks are used to enforce token durations and start/end times: the set of global clocks of the nTGA includes a clock *plan_clock*, whose value corresponds to the time elapsed since the beginning, and is used to constrain transitions so that they respect the start and end times of tokens. Moreover, each automaton $\mathcal{A}_x$ has a local clock $clock_x$, that is reset to zero on the incoming edge of every location and checked on the outgoing ones, in order to enforce the corresponding token duration. Its value when a run exits a location $\mu(x^i)$ represents in fact the time elapsed since it entered $\mu(x^i)$ (i.e. the duration of the corresponding scheduled token).

For instance, if $x^i = (v, [100, 120], [30, 40], c)$, then the invariant of the location $\mu(x^i)$ is $plan\_clock \leq 120 \wedge clock_x \leq 40$, and the guard of the transition from $\mu(x^i)$ to $\mu(x^{i+1})$ includes $plan\_clock \geq 100 \wedge clock_x \geq 30$.

The transition from *start* to $\mu(x^1)$ is constrained by a guard ($plan\_clock = 0$) that forces runs to enter $\mu(x^1)$ at time $0 = \text{start\_time}(x^1)$. If x is an external variable, the guard of the last transition of the automaton A_x, from $\mu(x^k)$ to *finish* (where x^k is the last token of FTL_x), does not require that $clock_x$ has reached the least duration value allowed for x^k, thus mirroring the fact that the last token of an external timeline is allowed not to respect the least duration requirement.

We now turn to give a sketchy description of how relations are modeled. Relations between a token and a time point are quite simple to be encoded by use of the plan clock. For instance, the relation x^i starts_before$_{[lb,ub]}$ t is encoded by adding the guard $plan\_clock \leq t - lb \wedge plan\_clock \geq t - ub$ to the transition entering $\mu(x^i)$.

In order to model relations between two tokens, a global clock is defined for each of them. A relation R of the form x^n $\mathsf{r}_{[lb,ub]}$ y^k is enforced by resetting the clock c_R associated to R on the edge entering/exiting $\mu(x^n)$ and checked by the guard on the edge entering/exiting $\mu(y^k)$. Whether the incoming or outgoing edges are concerned depends on the particular relation r. For instance, if $R = x^n$ start_before_end$_{[lb,ub]}$ y^k, then the clock c_R is reset on the edge entering $\mu(x^n)$ and checked on the edge exiting $\mu(y^k)$.

The encoding of relations exploits the possibility offered by UPPAAL-TIGA of setting clocks to non-zero values. It must in fact be guaranteed that guards concerning a relation clock c_R are not satisfied when the relation is not. So, a value H greater than the plan horizon is used to reset relation clocks (without explicit assignment no clock could reach H during any run). In the example above ($R = x^n$ start_before_end$_{[lb,ub]}$ y^k), the clock c_R is assigned the value H on the edge entering $\mu(x^n)$ and the guard on the edge exiting $\mu(y^k)$ is conjoined with $H + lb \leq c_R$ and (if $ub \neq \infty$) $c_R \leq H + ub$.

The encoding of a flexible plan $\Pi = \langle \mathbf{FTL}, \mathcal{R} \rangle$ in terms of a network of TGA is clearly linear in the size of the plan, measured in terms of the number of tokens making up the timelines of **FTL** and the number of relations in $\mathcal{R}$.

6 Experimental Results

This section investigates the practical feasibility of the approach and presents some preliminary experiments carried out by using a case study as a benchmark. Our aim is to test the performances of the verification process and the generation of a control strategy in a real world scenario to check whether on-line control synthesis is viable and compatible with the latencies of a planning and execution cycle. In particular, the experimental analysis considers a benchmark domain presented in [9] and inspired by a Space Long Term Mission Planning problem [7]. The mission consists of a remote satellite operating around a target planet. The satellite can either point to the planet and use its devices to produce scientific data or point towards a communication station (e.g. an Earth ground station) and communicate previously produced data. The satellite is endowed with a set of scientific devices or payloads (e.g. stereo cameras, altimeters, spectrometers, etc.) whose activities are to be planned during planet-pointing

phases taking into account some physical constraints. The satellite is controlled by a planner and an executive system to accomplish required tasks, i.e., scientific observations and communication. A set of operative constraints are to be satisfied: the satellite has to point towards the planet, thus allowing observations of the planet surface and Science operations by means of devices; the satellite has to point to Earth for transmitting data and communication with Earth must occur within a ground-station availability window. Communication opportunities are not under the system control, and consequently the availability window is represented by an *external* state variable. Moreover, the exact duration of the communication activity is uncertain, and can range in some given temporal interval. The same happens for the processing time of the satellite devices.

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* using the available scientific devices such as infrared camera to take pictures. 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 the scientific results to the Earth ground-station.

It is worth pointing out that the introduction of multiple scientific payloads considered here (and introduced in [9]) entails an increasing complexity of the real world planning problem described in [7] as it introduces a significant combinatorial effect on the alternation of scientific operations to be planned.

The flexible plans used in the experiments were generated by the EPSL (Extensible Planning and Scheduling Library) tool [10, 30], used as a CBTP Domain Independent Planner, run on the specifications of the planning domains and goals. The planner (that has suitably been modified in order to fit the needs of the present work) generates its results in the form of text files describing flexible plans, i.e. a set of flexible timelines and a set of relations on their tokens as defined in Section 2. The generated plans include information on uncontrollable tokens and the distinction between planned and external timelines. The translation of plans into UPPAAL-TIGA systems and queries, according to the encoding described in Section 5, was accomplished by PLAN2TIGA, that can be found at http://cialdea.dia.uniroma3.it/plan2tiga. Finally, the files generated by the encoding were processed by the UPPAAL-TIGA verifier for the synthesis of control strategies.

In general, this tool and methodology can be used by any planner, provided that it outputs the generated plans in a text file obeying the general syntax accepted by the program.

In order to empirically check the feasibility of our verification method, we deployed it in different configurations of the benchmark domain. In this way we analyzed various executive contexts both for checking *dynamic controllability* and synthesizing a *winning strategy*. The performances of the tool have been analyzed in different planning/execution scenarios obtained by varying the problem complexity along the following dimensions: i) the number of available *scientific devices*; ii) the number of *goal requests*; iii) the degree of *temporal uncertainty*. The different classes of problems were designed so that the impact of the following three dimensions could be evaluated: the number of timelines in the plan, their length (number of tokens) and the width of the duration intervals of uncontrollable tokens.

More specifically:

- *Number of scientific devices, determining the number of timelines in the plan.* The satellite can be endowed with different sets of scientific devices. We considered configurations from 1 to 4 devices. Each device is considered as a component of the system, thus, varying the number of devices affects the number of state variables (each for each device plus two others, for the overall system and the Earth visibility window) and synchronization constraints in the planning model and, as a consequence, the size of the generated plans to be checked, in terms of number of timelines and relations.

- *Goal requests, determining the complexity (number of tokens) in the timelines representing devices.* Each goal consists of using some device to perform its scientific activity, consisting of the following sequence of tasks: warm-up, process, turn-off. The available devices are used in a cycling sequence, so that, when the number of goals is greater than the number of available devices, some of them perform their sequence of tasks more than once. Conversely, if there are more devices than goals to be accomplished, some devices are unused and rest in their idle status all the time.

- *Temporal uncertainty, determining the width of the time intervals constraining the duration of uncontrollable activities.* For each uncontrollable activity (i.e., the devices process phase, communication and, obviously, the activities of the ground-station visibility window) we set a minimal a-priori known duration, but allow temporal flexibility on its termination, namely, we considered a tolerance of either 10 or 30 time units. This temporal interval represents the degree of temporal uncertainty in the system.

In principle, among all the generated problem instances, the ones with higher number of devices and goals as well as larger temporal uncertainty are the hardest ones to be both planned and controlled. In these scenarios, we collected and compared the running times for planning and verification/strategy synthesis, the overhead of the encoding process being neglectable. In these scenarios, we collected and compared the running times for planning (EPSL), on one hand, and verification/strategy synthesis (UPPAAL-TIGA), on the other. The overhead of the encoding process is neglectable.

The experiments were run on a PC endowed with an Intel Core i7 (2.80GHz) processor and 6GB RAM, and a timeout of 10 minutes was given to both systems. In what follows the reported timings are in seconds.

The diagram in Figure 1 plots the EPSL planning times for solving problems with 1 to 4 devices against the number of goals, considering the case of maximum temporal flexibility, i.e., 30. The problems with either 3 or 4 devices and more than 6 goals could not be solved by the planner, and the corresponding points are not plotted in the chart.

Figure 2 shows the corresponding diagram plotting the running times of UPPAAL-TIGA verifying the plans generated by EPSL. The plans with 4 devices could not be verified by UPPAAL-TIGA in the allowed 10 minutes time, but for the simpler ones (one and two goals, respectively in 93 and 269 seconds). The line corresponding to the plans with 4 devices is not plotted at all in the diagram. The points corresponding to plans with 3 devices and more than 6 goals are missing because EPSL did not generate them.

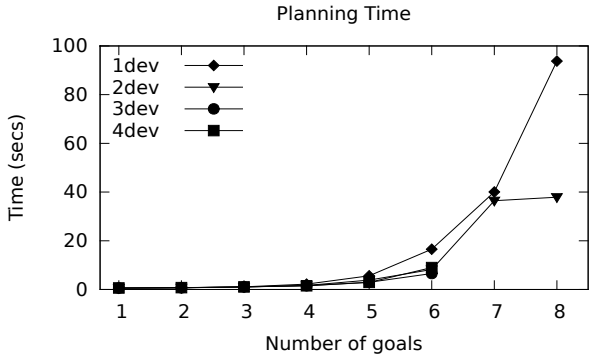

Figure 1: Planning time for problems with temporal flexibility 30.

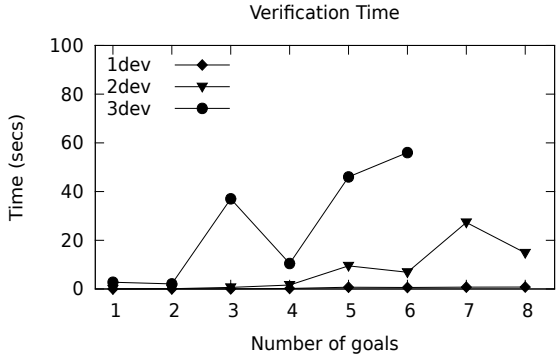

Figure 2: Verification time for plans with temporal flexibility 30.

Temporal flexibility seems not to affect the performances of UPPAAL-TIGA while verifying plans and generating execution strategies. In fact, comparing the running times for the problems with temporal flexibility 10 and 30, the average difference in speed is about 6 seconds. This is mainly due to the fact that UPPAAL-TIGA is specifically tailored for verifying temporal properties on timed game automata.

On the other hand, while the planning system seems to be negatively influenced by the presence of many tokens on the same timeline (due to different scientific requests on the same device), the verification process is strongly affected by the number of devices, determining in turn the number of timelines to be checked, i.e. the automata making up the network. Indeed, UPPAAL-TIGA is able to promptly check plans in the case of 1 or 2 devices regardless the number of goals. While its performances degrade when addressing scenarios with more devices, up to the case of 4 devices, where only two plans were successfully verified.

These preliminary experiments show that the performances of the verification process are comparable with planning costs when considering few devices for the satellite (i.e. up to 3) and, thus, also compatible with its deployment in on-line planning and execution control architectures. In general, the tool can be exploited to support off-line planning tools where a formal certification of

robust execution properties (i.e. dynamic controllability, and the consequent generation of an execution strategy) is more important than fast computation times.

7 Conclusions

This paper presents a formal account of flexible timelines and plans in the presence of uncertain and exogenous events, specifically addressing the dynamic controllability issue. Carrying on the work started in [12], the definition of flexible plans is extended introducing quantitative temporal relations as well as taking into account the difference between controllable and uncontrollable activities. Beyond formally defining the related main notions for flexible plans, a formal semantics is given of flexible plans in terms of *Timed Game Automata* (TGA). Similarly to [9, 29], UPPAAL-TIGA is exploited to verify whether a flexible plan is dynamically controllable and to generate a dynamic execution strategy by solving a *reachability game*. An initial empirical assessment shows the feasibility of the proposed approach when deployed in a benchmark domain derived from a real world context.

The contribution presented in this paper advances the work in [9, 29] on different perspectives. First of all, a more comprehensive approach is presented here, since controllability information is included in the description of the plan itself, thus avoiding the need of considering additional information derived from the specific execution contexts. Moreover, the encoding into TGA does not require to consider also the specification of the planning domain (like in the previously cited works) allowing for a more compact and straightforward translation of plans in terms of TGA. In summary, the whole information needed to encode and control the flexible plan is contained in its description. Furthermore, the present methodology allows also for the encoding of partially specified plans, thus making it possible to exploit the controllability check also during the planning process: during plan construction, the planner could query the TGA verifier about the current partial plan and check whether some planning decisions are affecting its dynamic controllability. The verifier can then provide suitable feedback to the planner so as to allow it to discard potential solution plans that, though consistent (with respect to the planning domain and problem), do not guarantee dynamic controllability.

In order to assess the feasibility of the proposed methodology, the preliminary empirical investigation presented and discussed in the paper is to be carried on more systematically, considering multiple and more complex domains and execution scenarios. A thorough experimental analysis can be of help in figuring out which are the crucial features of both planning problems and solution plans affecting the performances of the verification process, thus giving general guidelines for modeling planning domains. Moreover, the implementation of a tool translating the UPPAAL-TIGA strategy into executable code and the actual deployment of the whole approach in a real P&S control architecture (e.g. APSI-TRF) would constitute a natural and valuable future work.

Acknowledgments. The authors wish to thank Alessandro Umbrico for having endowed the EPSL system with the capability to generate its plans in the text format required by the encoding tool. Andrea Orlandini is partially funded by MIUR Flagship Initiative [FdF-SP1-T2.1] Project Generic Evolutionary Control

Knowledge-based Module (GECKO).

References

[1] Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.

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

[3] J. Bengtsson and W. Yi. Timed automata: Semantics, algorithms and tools. In *Lectures on Concurrency and Petri Nets*, volume 3098 of *LNCS*, pages 87–124. Springer, 2004.

[4] F. Cassez, A. David, E. Fleury, K. G. Larsen, and D. Lime. Efficient on-the-fly algorithms for the analysis of timed games. In *Proc. of 16th Int. Conf. on Concurrency Theory (CONCUR-05)*, volume 3653 of *LNCS*, pages 66–80. Springer, 2005.

[5] Franck Cassez, Alexandre David, Emmanuel Fleury, Kim G. Larsen, and Didier Lime. Efficient on-the-fly algorithms for the analysis of timed games. In *CONCUR 2005*, pages 66–80. Springer-Verlag, 2005.

[6] A. Cesta, G. Cortellessa, S. Fratini, and A. Oddi. Developing an End-to-End Planning Application from a Timeline Representation Framework. In *IAAI-09. Proc. of the 21st Innovative Application of Artificial Intelligence Conference, Pasadena, CA, USA*, 2009.

[7] A. Cesta, G. Cortellessa, S. Fratini, and A. Oddi. MrSPOCK: Steps in Developing an End-to-End Space Application. *Computational Intelligence*, 27(1), 2011.

[8] A. Cesta, G. Cortellessa, S. Fratini, A. Oddi, and N. Policella. An Innovative Product for Space Mission Planning: An A Posteriori Evaluation. In *ICAPS-07*, pages 57–64, 2007.

[9] A. Cesta, A. Finzi, S. Fratini, A. Orlandini, and E. Tronci. Analyzing Flexible Timeline Plan. In *ECAI 2010. Proceedings of the 19th European Conference on Artificial Intelligence*, volume 215. IOS Press, 2010.

[10] Amedeo Cesta, Andrea Orlandini, and Alessandro Umbrico. Toward a general purpose software environment for timeline-based planning. In *20th RCRA International Workshop on "Experimental Evaluation of Algorithms for solving problems with combinatorial explosion"*, 2013.

[11] M. Cialdea Mayer and A. Orlandini. Plan2tiga User Guide. available at http://cialdea.dia.uniroma3.it/plan2tiga, 2015.

[12] M. Cialdea Mayer, A. Orlandini, and A. Umbrico. A formal account of planning with flexible timelines. In *21st Int. Symp. on Temporal Representation and Reasoning (TIME 2014)*, pages 37–46, 2014.

[13] A. Cimatti, L. Hunsberger, A. Micheli, R. Posenato, and M. Roveri. Sound and complete algorithms for checking the dynamic controllability of temporal networks with uncertainty, disjunction and observation. In *21st International Symposium on Temporal Representation and Reasoning (TIME), 2014*, pages 27–36, 2014.

[14] A. Cimatti, L. Hunsberger, A. Micheli, and M. Roveri. Using timed game automata to synthesize execution strategies for simple temporal networks with uncertainty. In *Proc. of the 28th AAAI Conference on Artificial Intelligence*, 2014.

[15] A. Cimatti, A. Micheli, and M. Roveri. Timelines with temporal uncertainty. In *Proc. of the 27th AAAI Conference on Artificial Intelligence*, pages 195–201, 2013.

[16] G. Gerd Behrmann, A. Cougnard, A. David, E. Fleury, K. Larsen, and D. Lime. UPPAAL TIGA user-manual. available at `http://people.cs.aau.dk/~adavid/tiga/`.

[17] Robert P. Goldman, David J. Musliner, and Michael J. Pelican. Exploiting implicit representations in timed automaton verification for controller synthesis. In *HSCC-02. Proc. of the Fifth Int. Workshop on Hybrid Systems: Computation and Control*, 2002.

[18] Luke Hunsberger. Fixing the semantics for dynamic controllability and providing a more practical characterization of dynamic execution strategies. In *Proceedings of the 16th International Symposium on Temporal Representation and Reasoning (TIME-2009)*, pages 155–162. IEEE Computer Society, 2009.

[19] Luke Hunsberger. A fast incremental algorithm for managing the execution of dynamically controllable temporal networks. In *Proc. of the 17th Int. Symp. on Temporal Representation and Reasoning (TIME 2010)*, pages 121–128, 2010.

[20] A.K. Jonsson, P.H. Morris, N. Muscettola, K. Rajan, and B. Smith. Planning in Interplanetary Space: Theory and Practice. In *AIPS-00. Proceedings of the Fifth Int. Conf. on AI Planning and Scheduling*, 2000.

[21] S. Lemai and F. Ingrand. Interleaving temporal planning and execution in robotics domains. In *AAAI-04*, pages 617–622, 2004.

[22] O. Maler, A. Pnueli, and J. Sifakis. On the Synthesis of Discrete Controllers for Timed Systems. In *STACS*, LNCS, pages 229–242. Springer, 1995.

[23] R. Milner. *Communication and concurrency*. Prentice-Hall, Inc., 1989.

[24] Paul Morris. Dynamic controllability and dispatchability relationships. In Helmut Simonis, editor, *Integration of AI and OR Techniques in Constraint Programming*, volume 8451 of *Lecture Notes in Computer Science*, pages 464–479. Springer International Publishing, 2014.

[25] Paul H. Morris and Nicola Muscettola. Temporal Dynamic Controllability Revisited. In *Proc. of AAAI 2005*, pages 1193–1198, 2005.

[26] Paul H. Morris, Nicola Muscettola, and Thierry Vidal. Dynamic control of plans with temporal uncertainty. In *Proc. of the 17th International Joint Conference on Artificial Intelligence (IJCAI '01)*, pages 494–499. Morgan Kaufmann, 2001.

[27] N. Muscettola. HSTS: Integrating Planning and Scheduling. In Zweben, M. and Fox, M.S., editor, *Intelligent Scheduling*. Morgan Kauffmann, 1994.

[28] A. Orlandini, M. Suriano, A. Cesta, and A. Finzi. Controller synthesis for safety critical planning. In *IEEE 25th International Conference on Tools with Artificial Intelligence (ICTAI 2013)*, pages 306–313. IEEE, 2013.

[29] Andrea Orlandini, Alberto Finzi, Amedeo Cesta, and Simone Fratini. TGA-based controllers for flexible plan execution. In *KI 2011: Advances in Artificial Intelligence, 34th Annual German Conference on AI.*, volume 7006 of *Lecture Notes in Computer Science*, pages 233–245. Springer, 2011.

[30] Alessandro Umbrico, Andrea Orlandini, and Marta Cialdea Mayer. Enriching a temporal planner with resources and a hierarchy-based heuristic. In *Proc. of the 14th Conference of the Italian Association for Artificial Intelligence (AIxIA)*, 2015. In press.

[31] Thierry Vidal and Helene Fargier. Handling contingency in temporal constraint networks: from consistency to controllabilities. *Journal of Experimental and Theoretical Artificial Intelligence*, 11:23–45, 1999.