Flexible Timeline-Based Plan Verification

A. Cesta[1], A. Finzi[2], S. Fratini[1], A. Orlandini[3], and E. Tronci[4]

[1] ISTC-CNR, Via S.Martino della Battaglia 44, I-00185 Rome, Italy
[2] DSF "Federico II" University, Via Cinthia, I-80126 Naples, Italy
[3] DIA "Roma TRE" University, Via della Vasca Navale 79, I-00146 Rome, Italy
[4] DI "La Sapienza" University, Via Salaria 198, I-00198 Rome, Italy

Abstract. Flexible temporal planning is a general technique that has demonstrated wide applications possibilities in heterogeneous domains. A key problem for widening applicability of these techniques is the robust connection between plan generation and execution. This paper describes how a model-checking verification tool, based on UPPAAL-TIGA, is suitable for verifying flexible temporal plans. Moreover, we further investigate a particular perspective, i.e., the one of verifying dynamic controllability before actual plan execution.

1 Introduction

Timeline-based planning has been shown well suited for applications, e.g., [9]. A problem for a wider diffusion of such technology stems in the limited community that has been studying formal properties of this approach to planning. In a preliminary work [1], we have listed several directions for contamination between timeline-based planning and standard techniques for formal validation and verification, then we have started addressing properties to develop a robust connection between plan generation and execution. In this context, plan verification is a crucial verification task most of the V&V tasks considered in [1] rely on. In particular, our approach allows to deploy general purpose techniques, such as model checking, to this aim. Moreover, in real domains, a generated temporally flexible plan is to be executed by an *executive* system that manages controllable processes in the presence of exogenous events. In this scenario, the duration of the execution process is not completely under the control of the executive. Thus, verifying a temporal flexible plan before its actual execution requires particular attention and major efforts. In this paper, we present a formalization used for addressing the flexible plan verification problem that makes use of Timed Game Automata [6] and UPPAAL-TIGA [4], a well known model-checking tool for verification. Other works addressed similar problems [10,2] but, up to our knowledge, the present paper is the first dealing with flexible plan verification. Moreover, we show how plan verification can also be exploited to address the *controllability problem* [11,7]. Finally, we discuss some experimental results collected using the verification tool.

2 Timeline-Based Planning and Execution

Timeline-based planning is an approach to temporal planning [9] where the generated plans are represented by sets of timelines. Each timeline denotes the evolution of a

B. Mertsching, M. Hund, and Z. Aziz (Eds.): KI 2009, LNAI 5803, pp. 49–56, 2009.

particular feature in a dynamic system. A planning domain encodes the possible evolutions of the timelines whose time points have to satisfy temporal constraints, usually represented as Simple Temporal Problem (STP) restrictions.

Here, we assume that the timelines in a planning domain are incarnations of multi-valued *state variables* as in [9]. A state variable is characterized by a finite set of values describing its temporal evolutions, and by minimal and maximal duration for each value. More formally, a state variable is defined by a tuple $\langle \mathcal{V}, \mathcal{T}, \mathcal{D} \rangle$ where: (a) $\mathcal{V} = \{v_1, \ldots, v_n\}$ is a finite set of *values*; (b) $\mathcal{T} : \mathcal{V} \rightarrow 2^{\mathcal{V}}$ is the *value transition* function; (c) $\mathcal{D} : \mathcal{V} \rightarrow \mathbb{N} \times \mathbb{N}$ is the *value duration* function, i.e. a function that specifies the allowed duration of values in $\mathcal{V}$ (as an interval $[lb, ub]$). Given a state variable, its associated *timeline* is represented as a sequence of values in the temporal interval $\mathcal{H} = [0, H)$. Each value satisfies previous (a-b-c) specifications and is defined on a set of not overlapping time intervals contained in $\mathcal{H}$. We suppose that adjacent intervals present different values. A timeline is said *completely specified* over the temporal horizon $\mathcal{H}$ when a sequence of non-overlapping valued intervals exists and its union is equal to $\mathcal{H}$. A timeline is said *time-flexible* when is completely specified and transition events are associated to temporal intervals (lower and upper bounds are given for them), instead of exact temporal occurrences. In other words, a time-flexible timeline represents a set of timelines, all sharing the same sequence of values. It is worth noting that not all the timelines in this set are valid (satisfies a-b-c). The process of *timeline extraction* from a time-flexible timeline is the process of computing (if exists) a valid and completely specified timeline from a given time-flexible timeline. In timeline-based planning, a *planning domain* is defined as a set of state variables $\{\mathcal{SV}_1, \ldots, \mathcal{SV}_n\}$ that cannot be considered as reciprocally decoupled. Then, a *domain theory* is defined as a set of additional relations, called *synchronizations*, that model the existing temporal constraints among state variables. A synchronization has the form $\langle \mathcal{TL}, v \rangle \longrightarrow \langle \{\mathcal{TL}'_1, \ldots, \mathcal{TL}'_n\}, \{v'_1, \ldots, v'_{|\mathcal{TL}'|}\}, \mathcal{R} \rangle$ where: $\mathcal{TL}$ is the reference timeline; v is a value on $\mathcal{TL}$ which makes the synchronization applicable; $\{\mathcal{TL}'_1, \ldots, \mathcal{TL}'_n\}$ is a set of target timelines on which some values v'_j must hold; and $\mathcal{R}$ is a set of *relations* which bind temporal occurrence of the *reference* value v with temporal occurrences of the *target* values $v'_1, \ldots, v'_{|\mathcal{TL}'|}$. A *plan* is defined as a set of timelines $\{\mathcal{TL}_1, \ldots, \mathcal{TL}_n\}$ over the same interval for each state variable. A plan is *valid* with respect to a domain theory if every temporal occurrence of a reference value implies that the related target values hold on target timelines presenting temporal intervals that satisfy the expected relations. A plan is *time flexible* if $\exists \mathcal{TL}_i \in \{\mathcal{TL}_1, \ldots, \mathcal{TL}_n\}$ such that $\mathcal{TL}_i$ is time flexible.

At execution time, an executive cannot completely predict the behavior of the controlled physical system because the duration of certain processes or the timing of exogenous events is outside of its control. In these cases, the values for the state variables that are under the executive scope should be chosen so that they do not constrain uncontrollable events. This *controllability problem* is defined, e.g. in [11] where *contingent* and *executable* processes are distinguished. The contingent processes are not controllable, hence with uncertain durations, instead the executable processes are started and ended by the executive system. Controllability issues have been formalized and investigated for the Simple Temporal Problems with Uncertainty (STPU) in [11] where basic

formal notions are given for *dynamic* controllability (see also [8]). In the timeline-based framework, we introduce the same controllability concept defined on STNU as follows. Given a plan as a set of flexible timelines $\mathcal{PL} = \{\mathcal{TL}_1, \ldots, \mathcal{TL}_n\}$, we call *projection* the set of flexible timelines $\mathcal{PL}' = \{\mathcal{TL}'_1, \ldots, \mathcal{TL}'_n\}$ derived from $\mathcal{PL}$ setting to a fixed value the temporal occurrence of each uncontrollable timepoint. Considering N as the set of controllable flexible timepoints in $\mathcal{PL}$, a *schedule* T is a mapping $T : N \rightarrow \mathbb{N}$ where $T(x)$ is called *time* of timepoint x. A *schedule* is *consistent* if all value durations and synchronizations are satisfied in $\mathcal{PL}$. The *history* of a timepoint x w.r.t. a schedule T, denoted by $T\{\prec x\}$, specifies the time of all uncontrollable timepoints that occur prior to x. An *execution strategy* S is a mapping $S : \mathcal{P} \rightarrow \mathcal{T}$ where $\mathcal{P}$ is the set of projections and $\mathcal{T}$ is the set of schedules. An execution strategy S is viable if $S(p)$ (denoted also S_p) is consistent for each projection p. Thus, a flexible plan $\mathcal{PL}$ is *dynamically controllable* if there exists a viable execution strategy S such that $S_{p1}\{\prec x\} = S_{p2}\{\prec x\} \Rightarrow S_{p1}(x) = S_{p2}(x)$ for each controllable timepoint x and projections $p1$ and $p2$.

3 Timed Game Automata

A fundamental concept in Timed Automata is time. First, we give the formal definition of clocks and relations that can be defined over them. We call *clock* a nonnegative, real-valued variable. Let X be a finite set of clocks. We denote with $C(X)$ the set of constraints Φ generated by the grammar: $\Phi ::= x \sim c \mid x - y \sim c \mid \Phi \wedge \Phi$, where $c \in Z$, $x, y \in X$, and $\sim \in \{<, \leq, \geq, >\}$. We denote by $B(X)$ the subset of $C(X)$ that uses only constraints of the form $x \sim c$.

Definition 1. *A* **Timed Automaton (TA)** *[3] is a tuple* $\mathcal{A} = (Q, q_0, \text{Act}, X, \text{Inv}, E)$*, where:* Q *is a finite set of* locations*,* $q_0 \in Q$ *is the* initial location*,* Act *is a finite set of* actions*,* X *is a finite set of clocks,* $\text{Inv} : Q \rightarrow B(X)$ *is a function associating to each location* $q \in Q$ *a constraint* $\text{Inv}(q)$ *(the* invariant *of* q*),* $E \subseteq Q \times B(X) \times \text{Act} \times 2^X \times Q$ *is a finite set of* transitions *and each transition* (q, g, a, Y, q') *is noted* $q \overset{g,a,Y}{\rightarrow} q'$.

A *valuation* of the variables in X is a mapping v from X to the set $R_{\geq 0}$ of nonnegative reals. We denote with $R^X_{\geq 0}$ the set of valuations on X and with $\mathbf{0}$ the valuation that assigns the value 0 to each clock. If $Y \subseteq X$ we denote with $v[Y]$ the valuation (on X) assigning the value 0 ($v(z)$) to any $z \in Y$ ($z \in (X - Y)$). For any $\delta \in R^{\geq 0}$ we denote with $(v + \delta)$ the valuation such that, for each $x \in X$, $(v + \delta)(x) = v(x) + \delta$. Let $g \in C(X)$ and v be a valuation. We say that g satisfies v, notation $v \models g$ if constraint g evaluated on v returns true. A *state* of TA $\mathcal{A}$ is a pair (q, v) that $q \in Q$ and v is a valuation (on X). We denote with S the set of states of $\mathcal{A}$. An *admissible* state for $\mathcal{A}$ is a state (q, v) that $v \models \text{Inv}(q)$. A *discrete transition* for $\mathcal{A}$ is 5-tuple $(q, v) \overset{a}{\rightarrow} (q', v')$ where $(q, v), (q', v') \in S$, $a \in \text{Act}$ and there exists a transition $q \overset{g,a,Y}{\rightarrow} q' \in E$ that $v \models g$, $v' = v[Y]$ and $v' \models \text{Inv}(q')$. A *time transition* for $\mathcal{A}$ is 4-tuple $(q, v) \overset{\delta}{\rightarrow} (q, v')$ where $(q, v) \in S$, $(q, v') \in S$, $\delta \in R_{\geq 0}$, $v' = v + \delta$, $v \models \text{Inv}(q)$ and $v' \models \text{Inv}(q)$. A *run* of a TA $\mathcal{A}$ is a finite or infinite sequence of alternating time and discrete transitions of $\mathcal{A}$. We denote with $\text{Runs}(\mathcal{A}, (q, v))$ the set of runs of $\mathcal{A}$ starting from state (q, v) and write

Runs($\mathcal{A}$) for Runs($\mathcal{A}, (q, \mathbf{0})$). If ρ is a finite run we denote with last(ρ) the last state of run ρ. A ***network* of TA (nTA)** is a finite set of TA evolving in parallel with a CSS style semantics for parallelism. Formally, let $\mathcal{F} = \{\mathcal{A}_i \mid i = 1, \ldots n\}$ be a finite set of automata with $\mathcal{A}_i = (Q_i, q_i^0, \text{Act}, X, \text{Inv}_i, E_i)$ for $i = 1, \ldots n$. Note that the automata in $\mathcal{F}$ have all the same set of actions and clocks and disjoint sets of locations. The *network* of $\mathcal{F}$ (notation $||\mathcal{F}$) is the TA $\mathcal{P} = (Q, q^0, \text{Act}, X, \text{Inv}, E)$ defined as follows. The set of locations Q of $\mathcal{P}$ is the Cartesian product of the locations of the automata in $\mathcal{F}$, that is $Q = Q_1 \times \ldots Q_n$. The *initial state* q^0 of $\mathcal{P}$ is $q^0 = (q_1^0, \ldots q_n^0)$. The *invariant* Inv for $\mathcal{P}$ is $\text{Inv}(q_1, \ldots q_n) = \text{Inv}_1(q_1) \wedge \ldots \text{Inv}_n(q_n)$. The *transition relation* E for $\mathcal{P}$ is the synchronous parallel of those of the automata in $\mathcal{F}$. That is, E consists of the set of 5-tuples (q, g, a, Y, q') satisfying the following conditions: 1. $q = (q_1, \ldots q_n)$, $q' = (q_1', \ldots q_n')$; 2. There are $i \leq j \in \{1, \ldots n\}$ such that for all $h \in \{1, \ldots n\}$, if $h \neq i, j$ then $q_h = q_h'$. Furthermore, if $i = j$ then action a occurs only in automaton $\mathcal{A}_i$ of $\mathcal{F}$. 3. Both automata $\mathcal{A}_i$ and $\mathcal{A}_j$ can make a transition with action a. That is, $q_i \overset{g_i, a, Y_i}{\rightarrow} q_i' \in E_i$, $q_j \overset{g_j, a, Y_j}{\rightarrow} q_j' \in E_j$, $g = g_i \wedge g_j$, $Y = Y_i \cup Y_j$.

Definition 2. *A* **Timed Game Automaton (TGA)** *is a TA where the set of actions* Act *is split in two disjoint sets:* Act_c *the set of controllable actions and* Act_u *the set of* uncontrollable *actions.*

The notions of network of TA, run, configuration are defined in a similar way for TGA.

Given a TGA $\mathcal{A}$ and three symbolic configurations *Init*, *Safe*, and *Goal*, the *reachability control problem* or reachability game $RG(\mathcal{A}, \mathit{Init}, \mathit{Safe}, \mathit{Goal})$ consists in finding a *strategy* f such that starting from *Init* and executing f, $\mathcal{A}$ stays in *Safe* and reaches *Goal*. More precisely, a strategy is a partial mapping f from the set of runs of $\mathcal{A}$ starting from *Init* to the set $\text{Act}_c \cup \{\lambda\}$ (λ is a special symbol that denotes "do nothing and just wait"). For a finite run ρ, the strategy $f(\rho)$ may say (1) no way to win if $f(\rho)$ is undefined, (2) do nothing, just wait in the last configuration ρ if $f(\rho) = \lambda$, or (3) execute the discrete, controllable transition labeled by l in the last configuration of ρ if $f(\rho) = l$. A strategy f is *state-based* or *memory-less* whenever its result depends only on the last configuration of the run.

4 Building TGA from Timeline-Based Planning Specifications

The main contribution of this work is the description of how flexible timeline-based plan verification can be performed solving a Reachability Game using UPPAAL-TIGA. To this end, this section describes how a flexible timeline-based plan, state variables and domain theory can be formalized as an adequate nTGA. Timelines and state variables are mapped into TGA. In addition, a *Observer* TGA checks for both illegal values occurrences and synchronizations violations. Here, we distinguish between controllable and uncontrollable state variables/timelines to simplifying the formalization.

Given a flexible plan $\mathcal{P} = \{\mathcal{TL}_1, \ldots, \mathcal{TL}_n\}$, we define a TGA for each $\mathcal{TL}_i$. For each valued interval in the timeline (also called plan step), we consider a location in the automaton. An additional final location, labeled *goal*, is considered. We consider a unique *plan clock* c_p over all the timelines automata. Then, for each planned flexible

timeline $\mathcal{TL}$, we define a Timed Game Automaton $\mathcal{A}_{\mathcal{TL}} = (Q_{\mathcal{TL}}, q_0, \text{Act}_{\mathcal{TL}}, X_{\mathcal{TL}}, \text{Inv}_{\mathcal{TL}}, E_{\mathcal{TL}})$ as follows. For each i-th valued interval in $\mathcal{TL}$, we consider l_i in $Q_{\mathcal{TL}}$, plus the final location l_{goal} (q_0 is l_0). For each allowed value $v \in \mathcal{SV}_i$, we consider an action a_v. If the related state variable is controllable (uncontrollable) we add a_v in $\text{Act}_{c\mathcal{TL}}$ ($\text{Act}_{u\mathcal{TL}}$). The overall plan clock c_p is considered in $X_{\mathcal{TL}}$. For each i-th valued interval in $\mathcal{TL}$ and the related value v_p associated with the flexible interval timepoint $[lb, ub]$, we define $\text{Inv}_{\mathcal{TL}}(l_i) := c_p \leq ub$ and we define a transition $e = q \overset{g,a,Y}{\rightarrow} q'$ in $E_{\mathcal{TL}}$, where $q = l_i$, $q' = l_{i+1}$, $g = c_p \geq lb$, $a = v_p!$, $Y = \emptyset$. Finally, we define a final transition $e = q \overset{g,a,Y}{\rightarrow} q'$ in $E_{\mathcal{TL}}$, where $q = l_{pl}$ (where pl is the plan length), $q' = l_{goal}$, $g = \emptyset$, $a = \emptyset$, $Y = \emptyset$. The set $Plan = \{\mathcal{A}_{\mathcal{TL}_1}, ..., \mathcal{A}_{\mathcal{TL}_n}\}$ represents the planned flexible timelines description as a nTGA.

For each state variable $\mathcal{SV}$, we have a one-to-one mapping into a Timed Game Automaton $\mathcal{A}_{SV} = (Q_{SV}, q_0, \text{Act}_{SV}, X_{SV}, \text{Inv}_{SV}, E_{SV})$. In fact, for each allowed value v in $\mathcal{V}$, we consider a location l_v in Q_{SV} (q_0 is set according to the initial value of the related planned timeline). Then, for each allowed value $v \in \mathcal{V}$, we consider an action a_v. If the state variable is controllable (uncontrollable), we consider a_v in Act_{cSV} (Act_{uSV}). A local clock c_{sv} is considered in X_{SV}. Finally, for each allowed value $v \in \mathcal{V}$ and both the associated $\mathcal{T}(v) = \{vs_1, ..., vs_n\}$ and $\mathcal{D}(v) = [l_b, u_b]$, we define $\text{Inv}_{SV}(v) := c_{sv} \leq u_b$ and we define a transition $e = q \overset{g,a,Y}{\rightarrow} q'$, where $q = l_v$, $q' = l_{vs_i}$ in E_{SV}, $g = c_{sv} \geq l_b$, $a = a_{vs_i}?$, $Y = \{c_{sv}\}$. The set $SV = \{\mathcal{A}_{SV_1}, ..., \mathcal{A}_{SV_n}\}$ represents the State Variables description. Note that the use of actions as transitions label implements the synchronization between state variables and planned timelines.

A last TGA is the *Observer* that monitors synchronizations and values over SV and $Plan$. Basically, two locations are considered to represent *correct* and *error* status. For each possible cause of error, an appropriate transition is defined, forcing the Observer to hold the error location. In this sense, we define a TGA $\mathcal{A}_{Obs} = (Q_{Obs}, q_0, \text{Act}_{Obs}, X_{Obs}, \text{Inv}_{Obs}, E_{Obs})$ as follows. We consider $Q_{Obs} = \{l_{ok}, l_{err}\}$ (q_0 is l_{ok}), $\text{Act}_{uObs} = \{a_{fail}\}$, $X_{Obs} = \{c_p\}$. Inv_{Obs} is undefined. For each pair plan step and associated planned value (s_p, v_p) for each timeline $\mathcal{TL}$ and the related variable $\mathcal{SV}$, we define an uncontrollable transition $e = q \overset{g,l,r}{\rightarrow} q'$ in E_{Obs}, where $q = l_{ok}$, $q' = l_{err}$, $g = \mathcal{TL}_{s_p} \wedge \neg SV_{v_p}$, $l = a_{fail}$, $r = \emptyset$. Moreover, for each synchronization $\langle \mathcal{TL}, v \rangle \longrightarrow \langle \{\mathcal{TL}'_1, \ldots, \mathcal{TL}'_n\}, \{v'_1, \ldots, v'_n\}, \mathcal{R} \rangle$, we define an uncontrollable transition $e = q \overset{g,a,Y}{\rightarrow} q'$ in E_{Obs} where $q = l_{ok}$, $q' = l_{err}$, $g = \neg\mathcal{R}(\mathcal{TL}_v, \mathcal{TL}'_{1v'_1}, \ldots, \mathcal{TL}'_{nv'_n})$, $a = a_{fail}$, $Y = \emptyset$.

The nTGA $\mathcal{PL}$ composed by the set of automata $PL = SV \cup Plan \cup \{\mathcal{A}_{Obs}\}$ encapsulates Flexible plan, State Variables and Domain Theory descriptions.

5 Verifying Time Flexible Plans

Given the nTGA $\mathcal{PL}$ defined above, we can define a Reachability Game that ensures, if successfully solved, plan validity. In particular, we define $RG(\mathcal{PL}, \textit{Init}, \textit{Safe}, \textit{Goal})$ by considering *Init* as the set of initial locations of each automaton in $\mathcal{PL}$, *Safe* = $\{l_{ok}\}$ and *Goal* as the set of goal locations of each $\mathcal{TL}_i$ in $\mathcal{PL}$. By construction, it is possible to show that we use a one-to-one mapping between flexible behaviors defined by $\mathcal{P}$ and

automata behaviors defined by *Plan* ∪ *SV*. While, the Observer holds the error location iff either an illegal value occurs or a synchronization is violated. Then, $\mathcal{PL}$ adequately represents all and only the behaviors defined by the flexible plan $\mathcal{P}$.

In order to solve such a reachability game, we use UPPAAL-TIGA [4]. This tool extends UPPAAL [5] providing a toolbox for the specification, simulation, and verification of real-time games. If there is no winning strategy, UPPAAL-TIGA gives a counter strategy for the opponent (environment) to make the controller lose. Given a nTGA, a set of goal states (*win*) and/or a set of bad states (*lose*), four types of winning conditions can be issued [4]. We ask UPPAAL-TIGA to solve the $RG(\mathcal{PL}$, *Init*, *Safe*, *Goal*) checking the formula $\Phi = A$ *[Safe U Goal]* in $\mathcal{PL}$. In fact, this formula means that along all the possible paths, $\mathcal{PL}$ stays in *Safe* states until *Goal* states are reached. In other words, winning the game corresponds to ask UPPAAL-TIGA to find a strategy that, for each possible evolution of uncontrollable state variables, ensures goals to be reached and errors to be avoided. Thus, verifying with UPPAAL-TIGA the above property implies validating the flexible temporal plan.

Moreover, we show the feasibility and effectiveness of our verification method by addressing the relevant issue of plan controllability. In fact, we may notice that each possible evolution of uncontrollable automata corresponds to a timeline projection p. Each strategy/solution for the RG corresponds to a schedule T. And a set of strategy represents an execution strategy S. Thus, the winning strategies produced by UPPAAL-TIGA constitute a viable execution strategy S for the flexible timelines. The use of forward algorithms [4] guarantees that S is such that $S_{p1}\{\prec x\} = S_{p2}\{\prec x\} \Rightarrow S_{p1}(x) = S_{p2}(x)$ for each controllable timepoint x and projections $p1$ and $p2$. That is, the flexible plan is dynamically controllable.

6 Case Study and Preliminary Experiments

Figure 1 sketches a generic space domain in which the main timeline for a remote space agent should be synthesized. The planning goal is to allocate the temporal occurrences of science and maintenance operations as well as the agent's ability to communicate. There are two uncontrollable state variables pos and ava both representing orbit events. Variable pos models the position of the spacecraft with respect to a (given) deep space planet and takes values: "PERI" (pericentre, i.e. the orbital position closest to the planet) and "APO" (apocentre, i.e. the orbital position more far away from the planet). Variable ava (taking values *Available* and *Unavailable*) models the availability of the ground station opportunity windows, i.e., the visibility of the spacecraft with respect to Earth. The controllable state variable modeling the timeline should be synchronized with the above described uncontrollable variables.

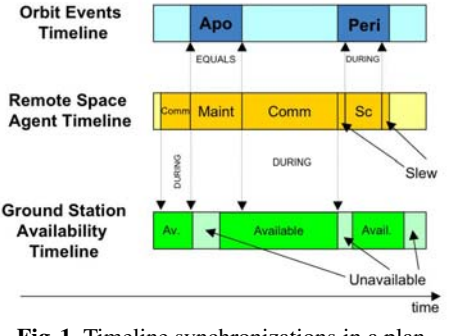

Fig. 1. Timeline synchronizations in a plan

This domain is a nondeterministic version of a real domain authors are working with [1], where the uncertainty is associated with the ground station availability. Any valid plan needs synchronizations among the agent controllable timeline (Figure 1, middle) and the uncontrollable timelines (see dotted arrows in Figure 1): science operations must occur during Pericentres, maintenance operations must occur in the same time interval as Apocentres and communications must occur during ground station visibility windows. In addition to those synchronization constraints, the operative mode timeline must respect transition constraints among values and durations for each value specified by the domain for the agent.

Here, we present some experimental results on preliminary tests focusing on the analysis of the dependency of plan verification performance from the degree of *flexibility*. We generate a flexible plan by introducing flexibility into a completely instantiated plan. This is done by replacing a time point $t = \tau$ in the instantiated plan with a time interval $t \in [\tau - \Delta, \tau + \Delta]$ in the flexible plan. The main parameters we consider are: the number Φ of time points that are replaced with time intervals and the width (*duration*) Δ of such intervals.

Φ \ plan size	10	20	35
3	35.6 ±0.8	36.6±1.7	37.4 ±0.5
6	35.2 ±0.4	36 ±0	37.4 ±0.5
9	36 ±1.8	36.2 ±0.4	39.2 ±1.9
12	34.8 ±0.4	36.4 ±0.5	37.8 ±0.4
15	35 ±0	36.2 ±0.4	43.6 ±10.2
18	35 ±0	40 ±8	39 ±0

Fig. 2. Experimental results collected varying plan length and the number of flexible time points (Timings in msecs)

Φ \ Δ	1	5	10	15	20
3	40±6	37.4±0.5	37.8±0.4	51±7.8	37.8±1
6	38.4±0.5	38.6±1.2	38±0	44.4±8.5	38.2±0.4
9	38.4±0.5	38±0	39.2±1.9	39±0	38.8±0.4
12	52.4±10.3	38.8±0.4	38.4±0.5	39±0	39.4±0.5
15	39.2±0.4	52±13	39.2±0.4	39.2±0.4	39.8±0.4
18	39.6±0.5	39.6±0.8	40.4±1.5	48.8±9.1	40±0.6

Fig. 3. Experimental results collected with a fixed plan length (Timing in msecs.)

We perform two kind of experiments. First, keeping Δ constant ($\Delta = 10$), we study how plan verification time depends on the plan size (i.e., the number of plan time points) and on the number of flexible time points Φ. Second, keeping constant the plan size (to 35 time points), we study how plan verification time depends on the number of flexible time points Φ and on the duration Δ. Given Φ and Δ, an experiment consists in choosing at random Φ plan time points, replacing such chosen time points with time intervals of duration Δ, running the UPPAAL-TIGA verifier and, finally, measuring the verification time. For each configuration we repeat our experiment 5 times and compute the mean value (in msecs.) and variance ($\pm var$) for the verification time[1]. We note that not all experiments relative to given values for Φ and Δ yield a satisfiable flexible temporal plan. In fact, since the plan is only flexible at certain time points, the degrees of freedom may not suffice to recover from previously delayed (or anticipated) actions. Of course this is particularly the case when Φ is small with respect to the plan size. Accordingly, our verification times refer to passing (i.e., the given flexible temporal plan is dynamically controllable) as well as failing (i.e., the given flexible temporal plan is not dynamically controllable) experiments. Figure 2 shows our results for the first kind of experiments depicting the homogeneous performances of the verification tool over

[1] We run experiments on a linux workstation with 64-bit AMD Athlon 3.5GHz and 2GB RAM.

all the configurations. In Figure 3, the second kind of experiments shows that the verification tool handles well flexible plan with higher and higher degrees of flexibility both in terms of Φ and Δ.

7 Conclusion

In this paper, we presented a method to represent and verify dynamic controllability of flexible plans using TGA and UPPAAL-TIGA. In particular, the paper describes the verification method, detailing the formal representation and the modeling methodology. Preliminary tests shows the feasibility of the approach. Few related approaches have been proposed in literature. For instance, [10] proposes a mapping from *Contingent Temporal Constraint Networks* (a generalization of STPUs) to TGA which is analogous to the one exploited here. In this work, the use of a model checker is only suggested to obtain a more compact representation and not to verify plan properties. Closely related is the work [2] which proposes a mapping from temporal constraint-based planning problems into UPPAAL-TIGA game-reachability problems and presents a comparison of the two planning approaches. Here, the approach to problem modeling is similar, however, in that work the flexibility issue remains open.

Acknowledgments. Cesta, Fratini, Orlandini and Tronci are partially supported by the EU project ULISSE (Call "SPA.2007.2.1.01 Space Science". Contract FP7.218815).

References

1. Cesta, A., Finzi, A., Fratini, S., Orlandini, A., Tronci, E.: Validation and Verification Issues in a Timeline-based Planning System. In: Knowledge Engineering Review (accepted, 2009)
2. Abdedaim, Y., Asarin, E., Gallien, M., Ingrand, F., Lesire, C., Sighireanu, M.: Planning Robust Temporal Plans: A Comparison Between CBTP and TGA Approaches. In: Proceedings of ICAPS, pp. 2–10 (2007)
3. Alur, R., Dill, D.L.: A Theory of Timed Automata. Theoretical Computer Science 126, 183–235 (1994)
4. Behrmann, G., Cougnard, A., David, A., Fleury, E., Larsen, K.G., Lime, D.: UPPAAL-TIGA: Time for Playing Games! In: Damm, W., Hermanns, H. (eds.) CAV 2007. LNCS, vol. 4590, pp. 121–125. Springer, Heidelberg (2007)
5. Larsen, K.G., Pettersson, P., Yi, W.: UPPAAL in a Nutshell. International Journal on Software Tools for Technology Transfer 1(1-2), 134–152 (1997)
6. Maler, O., Pnueli, A., Sifakis, J.: On the Synthesis of Discrete Controllers for Timed Systems. In: Mayr, E.W., Puech, C. (eds.) STACS 1995. LNCS, vol. 900, pp. 229–242. Springer, Heidelberg (1995)
7. Morris, P.H., Muscettola, N., Vidal, T.: Dynamic Control of Plans with Temporal Uncertainty. In: Proc. of IJCAI 2001, pp. 494–502 (2001)
8. Morris, P.H., Muscettola, N.: Temporal Dynamic Controllability Revisited. In: Proc. of AAAI 2005, pp. 1193–1198 (2005)
9. Muscettola, N.: HSTS: Integrating Planning and Scheduling. In: Zweben, M., Fox, M.S. (eds.) Intelligent Scheduling. Morgan Kaufmann, San Francisco (1994)
10. Vidal, T.: Controllability Characterization and Checking in Contingent Temporal Constraint Networks. In: Proceedings of KR 2000 (2000)
11. Vidal, T., Fargier, H.: Handling Contingency in Temporal Constraint Networks: from Consistency to Controllabilities. JETAI 11(1), 23–45 (1999)