A TGA-based Method for Safety Critical Plan Execution

Andrea Orlandini, Marco Suriano, Amedeo Cesta
CNR – National Research Council of Italy
Institute for Cognitive Science and Technology
Rome, Italy – {name.surname}@istc.cnr.it

Alberto Finzi
Università di Napoli "Federico II"
DIETI
Naples, Italy – alberto.finzi@unina.it

Abstract

Safety critical planning and execution is a crucial issue in autonomous systems. This paper proposes a methodology for controller synthesis suitable for timeline-based planning and demonstrates its effectiveness in a space domain where robustness of execution is a crucial property. The proposed approach uses Timed Game Automata (TGA) for formal modeling and the UPPAAL-TIGA model checker for controllers synthesis. An experimental evaluation is performed using a real-world control system.

Introduction

The design of safety critical and dependable systems is becoming increasingly important as technology advances. In fact, safety critical systems need to be certified and, thus, crucial properties, e.g., a bounded maximum execution time, require to be enforced. On the other hand, the need of meeting operational requirements in challenging domains as, for instance, in several space applications, often leads to the use of highly efficient software modules that address specific sub-parts of a larger problem with ad-hoc solving algorithms that cannot be easily verified. In this regard, the authors are working at the integration of Planning and Scheduling (P&S) technology with Validation and Verification (V&V) techniques to synthesize safety critical systems in space robotics. In particular, our current goal consists in cascading a timeline-based planner (OMPS (Fratini, Pecora, and Cesta 2008)) and a V&V technique based on Timed Game Automata (TGA) (Maler, Pnueli, and Sifakis 1995) to automatically synthesize a robot controller that guarantees certain properties.

More specifically, we are addressing the *dynamic controllability* issue (e.g., see (Morris and Muscettola 2005)): once a planner 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 capability is even more crucial when the generated plan is temporally flexible, as it 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]

A previous paper (Cesta et al. 2010a) introduces a technique for the P&S and V&V integration, while (Orlandini et al. 2011) first uses such integration for controller synthesis. The present paper integrates the technology in a real control architecture result from the GOAC project [2] (Ceballos et al. 2011) and explores the applicability in a set of experiments based on a real robot. The experimental evaluation shows the practical feasibility of the on-line deployment of such TGA-based approach in different operative modalities considering increasingly complex instances of a real-world robotics case study. In all the considered settings, robust plan execution is formally enforced maintaining plans as dynamically controllable. It is worth underscoring that, even though the running example is taken from a specific project, the work described in this paper is valid for any generic layered control architecture (e.g., (Gat 1997)) that integrates a temporal planning system.

Plan of the Paper. A first section introduces timeline-based planning and execution to set the context of the work. The second presents the integration with the TGA-based method. The real-world robotic scenario is illustrated in the subsequent section followed by the outcome of the associated empirical evaluation. Some conclusions end the paper.

Planning and Execution with Timelines

Timeline-base planning has been introduced in (Muscettola 1994) and has demonstrated successful in a number of space applications (Muscettola 1994; Jonsson et al. 2000; Cesta et al. 2007). The modeling assumption underlying this approach is inspired by the classical Control Theory: the problem is modeled by identifying a set of relevant *components* whose temporal evolutions need to be controlled to obtain a desired behavior. Components are primitive entities for knowledge modeling, and represent logical or physical

The present paper has been already accepted for publication at the IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2013).

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

[2] GOAC (Goal Oriented Autonomous Controller) has been a multi-institutional effort within the activities on robotics funded by the European Space Agency (ESA)

subsystems whose properties may vary in time. In this respect, the set of domain features under control are modeled as a set of temporal functions whose values have to be decided over a time horizon. Such functions are synthesized during problem solving by posting planning decisions. The evolution of a single temporal feature over a time horizon is called the *timeline* of that feature. In particular, for the purpose of this paper multi-valued *state variables* are considered as the basic type of time varying features (Cesta and Oddi 1996). As in classical control theory, the evolution of those features are described by some causal laws which determine legal temporal evolutions of timelines. For the state variables, such causal laws are encoded in a *Domain Theory* which determines the operational constraints of a given domain. Task of a planner is to find a sequence of control decisions that brings the variables into a final set of desired evolutions (i.e., the *Planning Goals*) always satisfying the domain specification.

In this area of research, a lot of effort has been dedicated to build software development environments, like EUROPA (Barreiro et al. 2012), ASPEN (Chien et al. 2010), and APSI-TRF (Cesta et al. 2009), aiming to facilitate the synthesis of timeline-based P&S applications. Nevertheless, a crucial issue in real applications is the tight integration of planning and execution.

Previous works have tackled the robust execution issue within a Constraint-based Temporal Planning (CBTP) framework deploying specialized techniques based on temporal-constraint networks. Several authors (Morris, Muscettola, and Vidal 2001; Morris and Muscettola 2005; Shah and Williams 2008; Hunsberger 2010) have proposed a *dispatchable execution* approach where a flexible temporal plan is then used by a plan executive that schedules activities on-line while guaranteeing constraint satisfaction. This general line of research has concerned specifically the use of timeline-based planning and their temporal constraint networks implementation for an homogeneous synthesis of controllers. Among the architectures that use a uniform representation for the continuous planning and execution task are IDEA (Muscettola et al. 2002), T-REX (Py, Rajan, and McGann 2010) and, more recently, GOAC (Ceballos et al. 2011). In particular, the GOAC effort combines several technologies: (a) a timeline-based deliberative layer which integrates a planner, called OMPS (Fratini, Pecora, and Cesta 2008), built on top of APSI-TRF to synthesize timelines and revise them according to execution needs, and an executive *a la* T-REX (Py, Rajan, and McGann 2010); (b) a functional layer (Bensalem et al. 2010) which combines a state of the art tool for developing functional modules of robotic systems (G$^{en}{}_{o}$M) with a component based framework for implementing embedded real-time systems (BIP).

In this context, the present paper particularly focuses on a timeline-based, domain independent deliberative control system, called APSI *Deliberative Reactor* (ADR) [3] (Cesta et al. 2012), proposed in GOAC. The ADR has been designed to address a set of open issues in planning and execution with timelines, i.e., the dynamic management of goals during planning and execution, the assessment of the status of partially executed goals and the dynamic dispatching of commands. More in detail, the ADR is an instance of a proactive control system entirely based on APSI-TRF technology and is constituted by (i) an execution module, to dispatch planned timelines, to supervise their execution status and to entail continuous planning and re-planning, (ii) a timeline-based planning module, i.e., OMPS, to model and solve planning problems.

The ADR is designed to be domain independent, i.e., once provided a suitable timeline-based description model of the system to be controlled and a set of temporal goals to be achieved it fully implements all the required functionalities to plan for goals, dispatch planned values to the controlled system and supervise plan execution collecting the telemetry of the controlled system. One of the main advantage of domain independence is the capability of the deliberative reactor to both plan for user goals and dynamically react to off-nominal conditions detected from the controlled system telemetry. Additionally, it allows flexibility in two direction: it can achieve different classes of user goals in the same system by substituting the controller model and it can be deployed to control different systems by substituting the domain description of the controlled system.

Finally, it is worth pointing out that, following a T-REX-like approach (Py, Rajan, and McGann 2010), the use of reactors allows to implement controller systems by means of hierarchical compositions of various deliberative reactors. In fact, reactors are differentiated on the basis of whether they need to deliberate in abstraction (reasoning on the highest level of representation) or they need to be responsive to the inputs from the lower levels closer to the robotic hardware. In the former case the planner a have larger planning horizon to deliberate and return partial plan for dispatching to other reactors. In the latter the planner has a no time to reason, hence it implements simple reactive policies. Such gradation allows the entire system to be both deliberative and reactive over its temporal scope. The GOAC architecture, whose executive component is based on T-REX, uses such a hierarchical configuration of reactors (see further details in (Ceballos et al. 2011)).

TGA-based controller synthesis

This section presents the integration in APSI-TRF of an alternative and novel approach to flexible plan dispatching/execution proposed in (Orlandini et al. 2011), where robust plan execution is pursued by relying on Timed Game Automata (TGA) formal modeling and controller synthesis. The technique used to synthesize plan controllers is a direct consequence of the formalization proposed in (Cesta et al. 2010a) in which plan correctness as well as dynamic controllability are checked by means of TGA model checking. Analogously to that work, the dynamic P&S domain and the generated flexible temporal plan are encoded into TGA models. However, a different perspective is exploited through the use of a model checker (i.e., UPPAAL-TIGA

[3] The term *Reactor* is a legacy from T-REX. It is also worth saying that the initial motivation of our work is to design a smooth integration with the T-REX executive that in its original implementation uses a different timeline-based planner.

(Behrmann et al. 2007)) to directly synthesize a real-time plan controller for the flexible plan. Such controller guarantees robust plan execution along with dynamic controllability.

TGA-based controllers for flexible plan execution

Timed Game Automata (Maler, Pnueli, and Sifakis 1995) (TGA) allow 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 (Cesta et al. 2010a) (and briefly discussed above), flexible timeline-based plan verification can be performed by solving a *Reachability Game* using UPPAAL-TIGA (Cassez et al. 2005). To this end, flexible timeline-based plans, state variables, and domain theory descriptions are compiled into a network of TGA (nTGA). This is obtained by means of through following steps: (1) a flexible timeline-based plan $\mathcal{P}$ is mapped into a nTGA *Plan*. Each timeline is encoded as a sequence of locations (one for each timed interval), while transition guards and location invariants are defined according to (respectively) lower and upper bounds of flexible timed intervals; (2) the associated set of state variables *SV* is mapped into a nTGA *StateVar*. Basically, a one-to-one mapping is defined between state variables descriptions and TGA. In such encoding, value transitions are partitioned into controllable and uncontrollable according to their actual execution profile; (3) an *Observer* automaton is introduced to check for violations of both value constraints and Domain Theory. In particular, two locations are defined: an Error location, to state constraint violations, and a Nominal (OK) location, to state that the plan behavior is correct. The *Observer* is defined as fully uncontrollable. (4) the compound nTGA $\mathcal{PL}$ = *StateVar* $\cup$ *Plan* $\cup$ $\{$*Observer*$\}$ encapsulates flexible plan, state variables and domain theory descriptions.

Then, considering a Reachability Game $RG(\mathcal{PL}$, *Init*, *Safe*, *Goal*) where *Init* represents the set of the initial locations of each automaton in $\mathcal{PL}$, *Safe* is the OK location of the Observer automaton, and *Goal* is the set of goal locations (one for each automaton in *Plan*), plan verification can be performed solving/winning the $RG(\mathcal{PL}$, *Init*, *Safe*, *Goal*) defined above. In order to win/solve the reachability game RG, UPPAAL-TIGA is exploited as verification tool checking a suitable CTL formula, i.e., Φ = *A [Safe U Goal]* in $\mathcal{PL}$. In fact, the formula Φ states that along all its possible temporal evolutions, $\mathcal{PL}$ remains in *Safe* states until *Goal* states are reached. That is, in all the possible temporal evolutions of the timeline-based plan $\mathcal{P}$ all the constraints are fulfilled and the plan is completed. Thus, if the solver verifies the above property, then the flexible temporal plan is valid. Whenever the flexible plan is not verified, UPPAAL-TIGA produces an execution trace showing one temporal evolution that leads to a fault. Such a strategy can be analyzed in order to check either for plan weaknesses or for the presence of flaws in the planning model.

Furthermore, a mapping between flexible temporal behaviors defined by $\mathcal{P}$ over the temporal horizon $[0, H]$ and the automata behaviors defined by $\mathcal{PL}$ can be shown: for each partial temporal behavior $pb \in \mathcal{P}$ defined over $H' < H$, it there exists a unique temporal evolution ρ_{pb} of $\mathcal{PL}$ such that ρ_{pb} represents the partial temporal behavior pb over the same horizon H'. That is, ρ_{pb} represents the same valued intervals sequence in $\mathcal{P}$ limited to H' and the duration of ρ_{pb} is exactly the horizon H'. As a consequence, the winning strategy f generated by UPPAAL-TIGA solving the reachability game on $\mathcal{PL}$ represents a flexible plan controller $\mathcal{C}_f$ that achieves the planning goals maintaining the dynamic controllability during the overall plan execution. In (Orlandini et al. 2011), the reader may find a formal account of the generation of a plan controller $\mathcal{C}_f$ derived from a winning strategy f generated by UPPAAL-TIGA.

Integrating the controller in the deliberative reactor

Here, the integration in the ADR of the TGA-based method discussed above is presented. In particular, a suitable embedding of the UPPAAL-TIGA tool within the ADR planning and execution cycle is shown and, then, the advantages in terms of plan correctness and robust execution enforcement (i.e., dynamic controllability) are discussed.

The integration schema is shown in Figure 1. The left part of the figure shows the APSI-TRF general architecture. The domain and problem models are encoded as Domain Definition Language (DDL) and Problem Definition Language (PDL) input files. Then, both DDL and PDL files are parsed and managed by the *Component-based Domain Modeling Engine* and a *Current Plan* (i.e., the initial planning problem) is created to be manipulated by a *Problem Solver*. Indeed, the Current Plan is specialized as a data structure called Decision Network in APSI-TRF. Then, a generic problem solver, e.g., OMPS, applies a solving procedure until the Current Plan satisfies all the planning goals (or fails in finding a solution plan).

The right part of Fig. 1 depicts a simplified view of the APSI Deliberative Reactor with two relevant services, i.e., the *Dispatch services* and the *Execution Feedback* modules, in charge of (respectively) dispatching suitable commands for the controlled system and collecting feedback from the field. The new APSI Deliberative Reactor architecture still reflects the structure of a T-REX reactor (as defined in (Cesta et al. 2012)) as well as it introduces two new components, i.e., the *TGA-based Controller* (TC) and the *Strategy Manager* (SM), enabling robust plans execution through the use of strategies generated by UPPAAL-TIGA.

The TC is in charge of managing plans in order to (i) verify plan correctness and (ii) generate a dynamically controllable execution strategy: once a solution plan $\mathcal{P}$ is generated by the problem solver (i.e., the stored Current Plan is actually the valid plan to be executed), the TC automatically generates the associated TGA encoding ($\mathcal{PL}$) and, then, invokes UPPAAL-TIGA in order to verify the correctness of the plan as well as to check for the existence of (at last) one temporal plan execution guaranteeing the correct achievement of the plan goals, independently from the exogenous events generated by the environment (i.e., enforcing the dynamic controllability). If the verifier finds one of these sequences, then a strategy for the plan execution is generated. Namely, a strategy generated by UPPAAL-TIGA is a set of *tempo-*

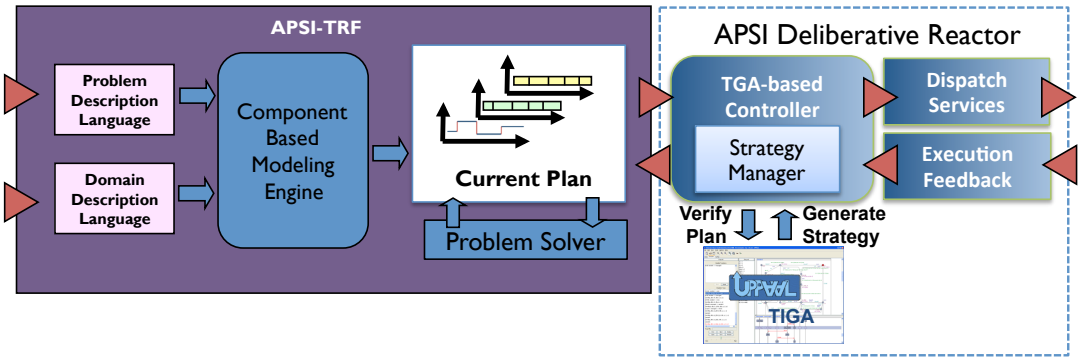

Figure 1: *Integration of TGA-based controller in the* APSI *Deliberative Reactor*

ral rules that should guide the controlled system through the *execution space* avoiding plan failures during its execution. More formally, an UPPAAL-TIGA strategy is a set of rules $f(t, s)$ defined as follows:

$$f(t,s) = \begin{cases} t_{wl} < t < t_{wu} & \text{Wait} \\ t_{al} < t < t_{au} & \text{Action } a_n \\ t > t_{err} & \text{Error} \end{cases}$$

where t is the execution time, s is one of the possible state of the system, t_{wu} and t_{wl} represent, respectively, lower and upper bounds of a time interval in which the system must wait for the environment to act, t_{al} and t_{au} represent lower and upper bounds of a time interval in which the system should perform the action a_n (i.e., one of the timeline should change value) and t_{err} is a time limit beyond which the system generates an error. The latter represents the case in which the execution strategy is coping with exogenous events that are not properly modeled in the planning domain, e.g., the actual duration of an uncontrollable event is shorter/longer than the minimal/maximal duration stated in the domain model. This implies that the planning model is inconsistent with the actual behavior of the controlled system and, thus, a revision of that model (and the TGA encoding) is required.

The SM is the module in charge of implementing the concrete dispatching policy relying on the UPPAAL-TIGA strategy. In fact, once generated, the SM exploit such strategy to to choose the more suitable $f(t, s)$ rule to be executed, thus, extracting the associated action to be dispatched (or to *wait* while the controlled system is evolving) as well as to continuously monitor the internal status of the reactor timelines and the execution feedback received from the field.

Given the above, the new integrated reactor architecture guarantees plans correctness as well as the robust execution of the generated plans, thus, increasing the probability of successfully performing the temporal plan.

Testing the Synthesis on a Robot Controller

This section describes a robotic scenario related to the GOAC project exploited as case study for the experimental assessment presented in the next section. First, we describe the DALA platform, i.e., the real robotic platform deployed within the GOAC project. Then, we exploit the same scenario in order to show a possible configuration of a control system implemented by means of an APSI Deliberative Reactor.

The Robotic Platform

The DALA rover is one of the LAAS-CNRS robotic platforms that can be used for autonomous exploration experiments. In particular, it is an iRobot ATRV robot that provides a large number of sensors and effectors. It can use vision based navigation (such as the one used by the Mars Exploration Rovers Spirit and Opportunity), as well as indoor navigation based on a Sick laser range finder. Then, the use of DALA in the GOAC project was to simulate a robotic scenario as close as possible to a planetary exploration rover.

In this regard, DALA can be considered as a fair representative for a planetary rover equipped with a Pan-Tilt Unit (PTU), two stereo cameras (mounted on top of the PTU), a panoramic camera and a communication facility. The rover is able to autonomously navigate the environment, move the PTU, take high-resolution pictures and communicate images to a Remote Orbiter. During the mission, the Orbiter may be not visible for some periods. Thus, the robotic platform can communicate only when the Orbiter is visible. The mission goal is a list of required pictures to be taken in different locations with an associated PTU configuration. A possible mission actions sequence is the following: navigate to one of the requested locations, move the PTU pointing at the requested direction, take a picture, then, communicate the image to the orbiter during the next available visibility window, put back the PTU in the safe position and, finally, move to the following requested location. Once all the locations have been visited and all the pictures have been communicated, the mission is considered successfully completed. The rover must operate following some operative rules to maintain safe and effective configurations. Namely, the following conditions must hold during the overall mission: **(C1)** While the robot is moving the PTU must be in the safe position (pan and tilt at 0); **(C2)** The robotic platform can take a picture only if the robot is still in one of the requested locations while the PTU is pointing at the related direction; **(C3)** Once a picture has been taken, the rover has to communicate the picture to the base station; **(C4)** While communicating, the rover has to be still; **(C5)** While communicating, the orbiter has to be visible. The reader may refer to (Ceballos et al. 2011) for further details.

The Figure 2 shows a timeline-based plan and the associated temporal constraints implementing the operative rules given above. The depicted constraints are: (**C1**) *GoingTo(x,y)* must occur during *PointingAt(0,0)*; (**C2**) *TakingPicture(pic,x,y,pan,tilt)* must occur during *At(x,y)* and *PointingAt(pan,tilt)*; (**C3**) *TakingPicture(pic,x,y,pan,tilt)* must occur before *Communicating(pic)*; (**C4**) *Communicating(file)* must occur during *At(x,y)*; (**C5**) *Communicating(file)* must occur during *Visible*.

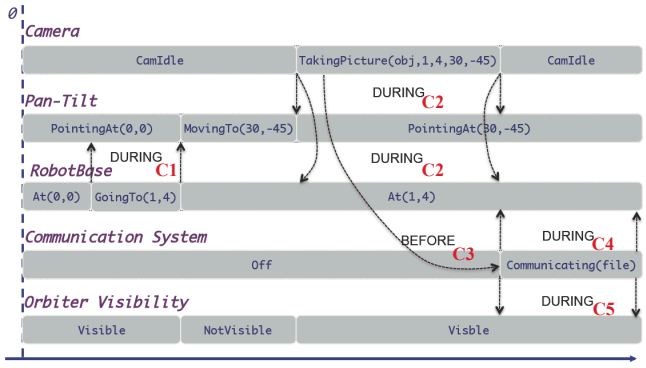

Figure 2: *An example of timeline-based plan with constraints.*

Control system configuration

According to T-REX design approach (Py, Rajan, and McGann 2010), a GOAC control system configuration has been designed considering an analogy between human control responsibilities for the mentioned rover, thus, implementing a suitable control system as the composition of a set of different deliberative reactors. Namely, some different personnel acting specific roles can be considered as involved in control tasks (Ceballos et al. 2011). As the goal of this paper is to evaluate the new architecture for the APSI Deliberative Reactor, taking advantage of the flexibility provided by the GOAC framework, a control system is here defined considering only two different reactors i.e., a *Mission Manager* responsible to perform all the deliberative tasks and a *Command Dispatcher* in charge of executing commands and collecting execution feedback.

More in detail, the Mission Manager reactor is designed to provide plans for user requested goals, i.e., requests for (i) scientific pictures in desired locations, (ii) reaching a certain position and (iii) monitoring a certain area. Then, the timelines planned by the Mission Managers are dispatched for execution to the Command Dispatcher reactor that, in turn, encodes the planned values into actual commands for the rover and uses the replies provided by the functional layer to produce observations on the low-level timelines. Thus, each reactor has a specific functional role over different temporal scopes during the mission: the Mission Manager's temporal scope is the entire mission and potentially can take minutes to deliberate; the Command Dispatcher interfaces to the DALA functional layer and requires minimal latency with no deliberation. It is also worth underscoring that the Mission Manager is the only APSI Deliberative Reactor in this use of the GOAC architecture. The Command Dispatcher is a fully reactive system that interacts with the actual controlled system with no deliberation task involved.

Empirical Evaluation

This section illustrates the assessment of the new APSI Deliberative Reactor performance considering the control system configuration presented in the previous section. Here, the aim is to assess the on-line TGA-based Controller synthesis performance in a real world scenario in order to show its viability with respect to actual execution requirements, i.e., the latencies of an on-line planning and execution cycle. Therefore, similarly to (Orlandini et al. 2011), different planning/execution scenarios are considered by varying the complexity of the robotic planning problem dimensions:
(1) *Plan Length.* Problem instances are considered with an increasing number of requested pictures (from 1 to 3). At the same time, flexible plans are generated over a horizon length ranging from 150 to 400 seconds.
(2) *Plan Flexibility.* For each uncontrollable activity (i.e., robot and PTU movements as well as camera and communication tasks), a minimal duration is set, but temporal flexibility on activity termination is considered, i.e., the end of each activity presents a tolerance ranging from 10 to 30 seconds. This interval represents the degree of temporal flexibility/uncertainty that we introduce in the system.
(3 *Plan Choices.* We define from 1 to 3 visibility windows that can be exploited to communicate picture contents. Notice that an increasing number of communication opportunities raises the complexity of the planning problem with a combinatorial effect.

More in general, among all the generated problems instances, the ones with higher number of required pictures, higher temporal flexibility, and higher number of visibility windows result as the hardest ones. In these scenarios, we analyzed the performance of the APSI Deliberative Reactor considering costs for planning, TGA model generation, plan verification-strategy synthesis and actual plan execution. The OMPS tool has been exploited as CBTP Domain Independent Planner. The DALA rover has been simulated by means of a software environment [4] used for testing the control system during the GOAC project and offering the same robotic functional interface as well as fully replicating the physical rover behaviors (i.e., random temporal durations for uncontrollable tasks). The experiments have been ran on a PC endowed with an Intel Core i7 CPU (2.93GHz) and 4GB RAM and, for each setting, 10 runs have been performed, and in tables, average timings are reported in milliseconds.

In Table 1, the performance of the APSI Deliberative Reactor during the whole planning and execution cycle are reported. Such execution settings seem to be suitable only in problems with one picture while, with 2 pictures, verification costs are rather dominating both deliberative and execution costs. For instance, with 3 communication windows and 30 seconds flexibility (i.e., the most complex scenario),

[4]DALA software simulator courtesy of Felix Ingrand and Lavindra De Silva from LAAS-CNRS.

Table 1: Performance with verification and strategy generation performed on a complete TGA model (timings in secs).

	1 Comm Window			2 Comm Windows			3 Comm Windows		
flex	**10**	**20**	**30**	**10**	**20**	**30**	**10**	**20**	**30**
PLANNING									
TP1	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3
TP2	1.0	1.1	1.0	1.1	1.1	1.1	1.1	1.1	1.1
TGA ENCODING									
TP1	0.008	0.007	0.006	0.007	0.009	0.006	0.009	0.007	0.006
TP2	0.007	0.007	0.008	0.008	0.006	0.007	0.007	0.007	0.008
PLAN VERIFICATION & STRATEGY GENERATION ON COMPLETE TGA MODEL									
TP1	0.6	0.6	0.8	0.6	0.6	0.8	0.6	0.8	2.2
TP2	137.8	152.8	149.9	137.4	149.4	150.5	139.4	150.3	151.5
PLAN EXECUTION									
TP1	27.9	38.3	42.7	30.8	36.5	44.5	31.7	36.4	40.4
TP2	65.5	77.2	103.5	60.4	78.7	89.0	66.0	78.0	106.1

even the execution costs are comparable with the time spent by UPPAAL-TIGA in verifying the plan and generating the strategy. Moreover, in the case of 3 pictures, UPPAAL-TIGA has been always terminated after 500 seconds with no suitable generated strategy.

Table 2: Performance with verification performed on a complete TGA model and strategy generation on a reduced TGA model (timings in msecs).

	1 Comm Window			2 Comm Windows			3 Comm Windows		
flex	**10**	**20**	**30**	**10**	**20**	**30**	**10**	**20**	**30**
PLANNING									
TP1	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3
TP2	1.0	1.0	0.9	1.0	1.1	1.0	1.1	1.1	1.1
TP3	14.9	15.2	14.1	15.4	15.7	15.9	16.4	16.4	16.5
TGA ENCODING									
TP1	0.006	0.006	0.006	0.006	0.006	0.007	0.007	0.006	0.006
TP2	0.006	0.005	0.005	0.005	0.006	0.005	0.006	0.005	0.005
TP3	0.004	0.004	0.003	0.005	0.004	0.004	0.005	0.005	0.004
PLAN VERIFICATION (COMPLETE) & STRATEGY GENERATION (REDUCED)									
TP1	0.6	0.5	0.5	0.6	0.5	0.5	0.5	0.5	0.5
TP2	10.8	10.8	10.7	10.8	10.8	10.6	10.8	10.6	10.6
TP3	70.4	70.7	70.1	70.3	70.4	71.1	70.6	70.5	70.5
PLAN EXECUTION									
TP1	37.6	46.4	49.8	38.8	47.6	50.2	39.4	49.6	50.8
TP2	89.4	100.8	120.6	87.0	108.6	117.4	90.8	110.8	124.0
TP3	142.2	165.6	169.6	137.8	166.6	182.6	135.8	172.8	180.2

Then, considering the different performance of the verification tool in checking plan correctness only (see (Orlandini et al. 2011)) and taking advantage of the flexibility of the TGA method, a slightly modified approach has been deployed and tested. The TC in the APSI deliberative reactor has been modified in order to invoke first the UPPAAL-TIGA tool to check the plan correctness on the complete TGA model $\mathcal{PL}$ without generating the winning strategy and, afterward, to ask the verification tool for generating a strategy on a *reduced TGA model*. Namely, the TC produces a reduced TGA model considering only the plan *Plan* and the *Observer* automata (i.e., focusing the strategy generation on the plan and the domain theory descriptions only) relying on the fact that the plan validity is guaranteed by the previous verification step.

Such approach leads to verification performance more compatible with the considered on line execution scenarios even though, yet, plan verification and strategy generation costs can not be neglected with respect to planning and execution costs (see Table 2). Furthermore, considering the average timing values among all the execution settings (i.e., the average values of each rows in the table), Figure 3 depicts a radar chart showing how each phase is affecting the whole planning and execution cycle. The plan verification and strategy generation task is always greater of almost one order of magnitude (axis are in logarithmic scale) and, in the 3 pictures settings, such cost is comparable even with plan execution cost.

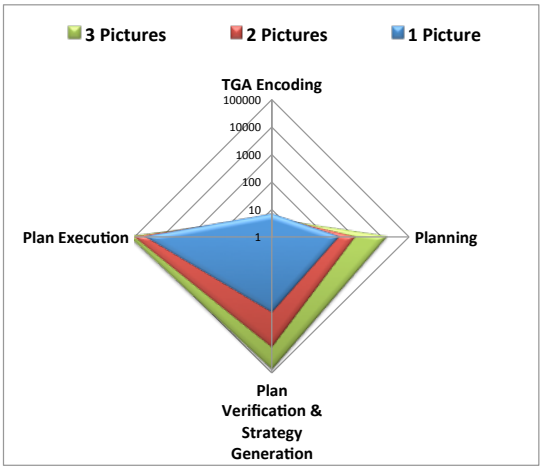

Figure 3: Performance related to full execution with plan verification and strategy generation performed in two different steps (timings in msecs).

Then, a further modification of the TC has been deployed where strategy generation is performed on the reduced TGA model without checking plan correctness. This option entails the strong assumption that every plan generated by the problem solver, in this case OMPS, is supposed to be valid, i.e., off-line plan verification is requested. In Table 3, the reported performance shows that planning and strategy generation costs are equivalent and fully compatible in all the plan execution scenarios. This is shown also in Figure 4 in which, again, average values are considered in the radar charts.

Table 3: Performance with strategy generation only on a reduced TGA model (timings in msecs).

	1 Comm Window			2 Comm Windows			3 Comm Windows		
flex	**10**	**20**	**30**	**10**	**20**	**30**	**10**	**20**	**30**
PLANNING									
TP1	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3
TP2	1.0	1.0	0.9	1.0	1.1	1.0	1.1	1.1	1.1
TP3	14.9	15.2	14.1	15.4	15.7	15.9	16.4	16.3	16.5
TGA ENCODING									
TP1	0.006	0.006	0.006	0.006	0.006	0.007	0.007	0.006	0.006
TP2	0.006	0.005	0.005	0.005	0.006	0.005	0.006	0.005	0.005
TP3	0.004	0.004	0.003	0.005	0.004	0.004	0.005	0.005	0.004
STRATEGY GENERATION ON REDUCED TGA MODEL									
TP1	0.6	0.5	0.5	0.6	0.5	0.5	0.5	0.5	0.5
TP2	3.9	3.7	3.6	3.8	3.7	3.6	3.8	3.6	3.6
TP3	15.4	15.7	15.1	15.3	15.3	16.1	15.6	15.5	15.5
PLAN EXECUTION									
TP1	37.6	46.4	49.8	38.8	47.6	50.2	39.4	49.6	50.8
TP2	89.4	100.8	120.6	87.0	108.6	117.4	90.8	110.8	124.0
TP3	142.2	165.6	169.6	137.8	166.6	182.6	135.8	172.8	180.2

Discussion The present experimental evaluation shows that the APSI Deliberative Reactor infrastructure allows the

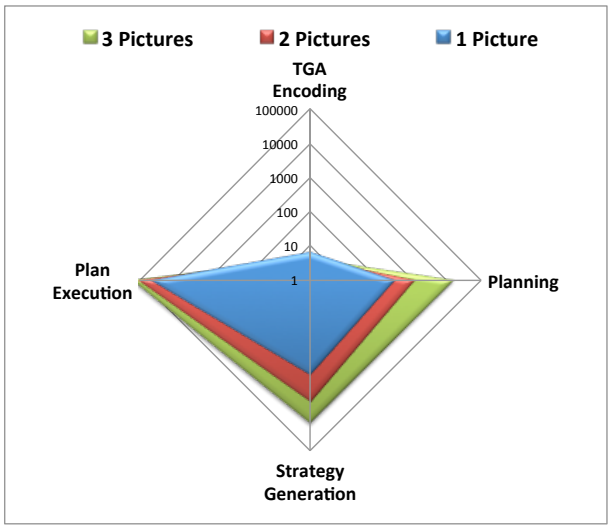

Figure 4: Performance related to full execution with plan verification and strategy generation performed in two different steps (timings in msecs).

deployment of different compositions of verification and strategy generation tasks. In particular, the TGA-based Controller is adaptable to different real contexts allowing the implementation of different suitable controller solutions. The most effective and affordable composition considers only a strategy generation task even though it entails the assumption on the validity of generated plans. Envisaging a use of the technique within a suitable Knowledge Engineering systems (Cesta et al. 2010b; Bernardi et al. 2013) potentially guarantees the deploying of an APSI-based application after an extensive off-line plan verification and testing phases (in addition to the known deterministic behavior of the considered problem solver) and suggests to consider also such composition as a fully reliable solution. More generally, the APSI Deliberative Reactor is open to support any operative modality with a computational load that can be tuned according to the criticality of the controlled system.

Conclusion

In this paper, an extension of the APSI Deliberative Reactor control system has been presented integrating a TGA-based plan controller synthesis approach, thus, enforcing robust plan execution. Then, an experimental evaluation has been reported discussing the practical feasibility of the on-line deployment of such TGA-based approach in different operative modalities and considering increasingly complex instances of a real-world robotics case study derived from a research project funded by the European Space Agency. However, the work described here is valid for any generic layered control architecture (e.g., (Gat 1997)) that integrates a temporal planning and scheduling system.

The reported results show the viability of the approach as well as enforce two main general advantages: the presented methodology relies on off-the-shelf planning/verification tools and, thus, it enables its application to any generic layered control architecture that integrates a temporal P&S system; the possibility of applying different settings for the control system allows to look for trade-off between planning, verification and execution costs, i.e., the control system can be tuned up according to the actual criticality of the controlled system.

Acknowledgments. Cesta, Orlandini and Suriano are partially funded by the Italian Ministry for University and Research (MIUR) and CNR under the GECKO project (Progetto Bandiera "La Fabbrica del Futuro"). Finzi is partially supported by the EC within the SHERPA FP7 project under grant agreement ICT-600958.

References

Barreiro, J.; Boyce, M.; Do, M.; Frank, J.; Iatauro, M.; Kichkaylo, T.; Morris, P.; Ong, J.; Remolina, E.; Smith, T.; and Smith, D. 2012. 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*.

Behrmann, G.; Cougnard, A.; David, A.; Fleury, E.; Larsen, K.; and Lime, D. 2007. UPPAAL-TIGA: Time for playing games! In *Proc. of CAV-07*, number 4590 in LNCS, 121–125. Springer.

Bensalem, S.; de Silva, L.; Gallien, M.; Ingrand, F.; and Yan, R. 2010. "Rock Solid" Software: A Verifiable and Correct-by-Construction Controller for Rover and Spacecraft Functional Levels. In *i-SAIRAS-10. Proc. of the 10th Int. Symp. on Artificial Intelligence, Robotics and Automation in Space*.

Bernardi, G.; Cesta, A.; Orlandini, A.; and Finzi, A. 2013. A knowledge engineering environment for p&s with timelines. In *ICAPS Workshop on Knowledge Engineering for Planning and Scheduling (KEPS)*.

Cassez, F.; David, A.; Fleury, E.; Larsen, K. G.; and Lime, D. 2005. Efficient on-the-fly algorithms for the analysis of timed games. In *CONCUR 2005*, 66–80. Springer-Verlag.

Ceballos, A.; Bensalem, S.; Cesta, A.; de Silva, L.; Fratini, S.; Ingrand, F.; Ocon, J.; Orlandini, A.; Py, F.; Rajan, K.; Rasconi, R.; and van Winnendael, M. 2011. A Goal-Oriented Autonomous Controller for Space Exploration. In *ASTRA-11. 11th Symposium on Advanced Space Technologies in Robotics and Automation*.

Cesta, A., and Oddi, A. 1996. DDL.1: A Formal Description of a Constraint Representation Language for Physical Domains,. In Ghallab, M., and Milani, A., eds., *New Directions in AI Planning*. IOS Press: Amsterdam.

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

Cesta, A.; Cortellessa, G.; Fratini, S.; and Oddi, A. 2009. 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*.

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

Cesta, A.; Finzi, A.; Fratini, S.; Orlandini, A.; and Tronci, E. 2010b. Validation and Verification Issues in a Timeline-Based Planning System. *Knowledge Engineering Review* 25(3):299–318.

Cesta, A.; Fratini, S.; Orlandini, A.; and Rasconi, R. 2012. Continuous Planning and Execution with Timelines. In *i-SAIRAS-12. Proc. of the 11th Int. Symp. on Artificial Intelligence, Robotics and Automation in Space*.

Chien, S.; Tran, D.; Rabideau, G.; Schaffer, S.; Mandl, D.; and Frye, S. 2010. Timeline-Based Space Operations Scheduling with External Constraints. In *ICAPS-10. Proc. of the 20th Int. Conf. on Automated Planning and Scheduling*.

Fratini, S.; Pecora, F.; and Cesta, A. 2008. Unifying Planning and Scheduling as Timelines in a Component-Based Perspective. *Archives of Control Sciences* 18(2):231–271.

Gat, E. 1997. On Three-Layer Architectures. In *Artificial Intelligence and Mobile Robots*. MIT Press.

Hunsberger, L. 2010. A fast incremental algorithm for managing the execution of dynamically controllable temporal networks. In *Temporal Representation and Reasoning (TIME), 2010 17th International Symposium on*, 121–128.

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

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

Morris, P. H., and Muscettola, N. 2005. Temporal Dynamic Controllability Revisited. In *Proc. of AAAI 2005*, 1193–1198.

Morris, P. H.; Muscettola, N.; and Vidal, T. 2001. Dynamic Control of Plans With Temporal Uncertainty. In *Proc. of IJCAI 2001*, 494–502.

Muscettola, N.; Dorais, G. A.; Fry, C.; Levinson, R.; and Plaunt, C. 2002. Idea: Planning at the core of autonomous reactive agents. In *Proc. of NASA Workshop on Planning and Scheduling for Space*.

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

Orlandini, A.; Finzi, A.; Cesta, A.; and Fratini, S. 2011. 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*, 233–245. Springer.

Py, F.; Rajan, K.; and McGann, C. 2010. A Systematic Agent Framework for Situated Autonomous Systems. In *AAMAS-10. Proc. of the 9th Int. Conf. on Autonomous Agents and Multiagent Systems*.

Shah, J., and Williams, B. C. 2008. Fast Dynamic Scheduling of Disjunctive Temporal Constraint Networks through Incremental Compilation. In *ICAPS-08*, 322–329.