Proceedings of the
1st Workshop on Artificial Intelligence and Formal Verification, Logics, Automata and Synthesis (OVERLAY),
Rende, Italy, November 19–20, 2019

A Prototype for the Robust Execution of Flexible Plans

Annarita Lanzilli, Marta Cialdea Mayer[1] and Amedeo Cesta, Andrea Orlandini, Alessandro Umbrico[2]

[1]ROMA TRE University, Rome, Italy
[2]ISTC-CNR, Rome, Italy

Abstract

Timeline-based Planning and Scheduling (P&S) usually deals with two main sources of uncertainty: some components may depend on an external environment and cannot be planned; there may be tasks whose duration cannot be exactly foreseen in advance. Such uncertainties are formally defined and consequent controllability issues have been addressed, focusing on dynamic controllability. In this work, we present a new software prototype, TIGA2EXEC, for dynamic controllable execution of timeline-based plans leveraging recent results gathered from the integration of P&S and Model Checking techniques. TIGA2EXEC is deployed in a timeline-based planning system to control plan execution guaranteeing dynamic controllability. A preliminary experimental evaluation is also presented.

1 Introduction

Robust plan execution in uncertain and dynamic environments is a critical issue for plan-based autonomous systems (see, e.g., [9]). Once a planner has generated a temporal plan, it is up to an 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. 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 (see, e.g., [12, 8, 3]). Timeline-based planning has been introduced in [12] under a modeling assumption inspired by classical control theory. 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*). 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. They can be exploited by an executive system for robust on-line execution. This work carries on the work started in [6] and further advanced in [7, 5] where a formal account of flexible timelines and plans is given addressing controllability issues. A P&S framework was provided to define flexible plans, considering 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 [11]. Moreover, a formal semantics of flexible plans has been given in terms of *Timed Game Automata* (TGA) [10]. Following the same approach presented in [4, 13], a verification tool, UPPAAL-TIGA [2], is exploited to verify whether a flexible plan is dynamically controllable and to generate a dynamic execution strategy by solving a *reachability game*. The above notions have also been extended to a timeline-based framework [4], 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 [13, 14]. In this work, we present a software prototype, TIGA2EXEC, for dynamic controllable execution of timeline-based plans pursuing the same approach presented in [5]. TIGA2EXEC implements a work chain that i) encodes flexible plans as a network of TGA, ii) exploits UPPAAL-TIGA to check dynamic controllability property for flexible plans and generate dynamic execution strategies and iii) implements such strategies as robust plan execution controllers. TIGA2EXEC is deployed in a P&S system, i.e., PLATINUM, to control plan execution while guaranteeing dynamic controllability. A preliminary experimental evaluation is also presented considering a benchmark domain introduced in [4].

2 Timeline-based Planning and Execution

A timeline-based planning domain contains the characterization of a set of *state variables*, representing the components of a system. A *state variable* x is characterized by the set of values it may assume, denoted by values(x), possible upper and lower bounds on the duration of each value, and rules governing the correct sequencing of such values. A *timeline* for a state variable is made up of a finite sequence of valued intervals, called *tokens*, each of which represents a time slot where the variable assumes a given value. In general, timelines may be *flexible*, i.e., the start and end times of each of its tokens are not necessarily fixed time points, but may range in given intervals. Tokens in a timeline for the state variable x are denoted by expressions of the form x^i, where the superscript indicates the position of the token in the timeline. Each token x^i is characterized by a value $v_i \in \text{values}(x)$, an end time interval $[e_i, e'_i]$ referred to as end_time(x^i), and a duration interval $[d_i, d'_i]$ (as usual, the notation $[x, y]$ denotes the closed interval $\{t \mid x \leq t \leq y\}$). The start time interval start_time(x^i) of the token x^i is $[0, 0]$ if x^i is the first token of the timeline (i.e. $i = 1$), otherwhise, if $i > 1$, start_time(x^i) = end_time(x^{i-1}). So, a token has the form $x^i = (v_i, [e_i, e'_i], [d_i, d'_i])$ and a timeline is a finite sequence of tokens $x^1, \ldots, x^k$. The metasymbol FTL (FTL_x) will henceforth be used to denote a timeline (for the state variable x), and $\mathbf{FTL}$ to denote a set of timelines. Being tokens flexible, their exact start end end times will be decided at execution time. Tokens can be either *controllable* (the controller can decide both their start and end time), or *uncontrollable* (both start and end time depend on the environment's choices), or *partially controllable* (the controller can decide when to start them, but their exact duration is outside the system's control). Each token is consequently equipped also with a *controllability tag*, identifying the class it belongs to. The presented theoretical background follows the formal framework presented in [7, 5].

The behavior of state variables may be restricted by requiring that time intervals with given state variable values satisfy some temporal constraints. Such constraints are stated as a set of *synchronization rules* which relate tokens on possibly different timelines through temporal relations between intervals or between an interval and a time point. These temporal relations refer to token start or end points, that will henceforth be called *events*. Other relations between tokens [1] can be defined in terms of the primitive ones. Temporal relations are also used to state the synchronization rules of the planning domain. A *flexible plan* Π is a pair $(\mathbf{FTL}, \mathcal{R})$, where $\mathbf{FTL}$ is a set of timelines and $\mathcal{R}$ is a set of temporal relations, involving tokens in some timelines in $\mathbf{FTL}$. An *instance* of the flexible plan $\Pi = (\mathbf{FTL}, \mathcal{R})$, is any schedule of $\mathbf{FTL}$ (i.e., set of timelines with fixed timings) that satisfies every relation in $\mathcal{R}$. In order for a flexible plan $\Pi = (\mathbf{FTL}, \mathcal{R})$ to satisfy a synchronization rule it must be the case that $\mathcal{R}$ contains temporal relations guaranteeing what the rule requires. During the execution, not any possible instance of a flexible solution plan can be safely executed, depending on how the environment will decide to schedule external variables and uncontrollable tokens. Thus, a control strategy is needed to determine how to schedule controllable tasks. A flexible solution plan is said to be *dynamically controllable* if there exists a control strategy that is able to schedule all the controllable time points such that (i) all problem rules are satisfied and (ii) only past events are considered to decide the schedule of subsequent ones. We focus on dynamic controllability since it is the property that most often captures the needs of real world scenarios. In [5], a semantics of flexible plans in terms of TGA networks is also defined, showing how flexible plans can be encoded into such formalism (see also PLAN2TIGA `http://cialdea.dia.uniroma3.it/plan2tiga/`). This allows to exploit existing verification tools for TGA to check dynamic controllability property for flexible plans and, possibly, generate a dynamic execution strategy that can be used for robust plan execution.

3 A new prototype for robust timeline-based plan execution

TIGA2EXEC is a new software prototype composed by a set of modules for representing a strategy generated by UPPAAL-TIGA and making it accessible (via a suitable software interface) for timeline-based P&S systems to provide plan dispatching strategy, i.e., a dynamic decision mechanism for token activation according to a given system status and execution time. Three main features are considered for prototype design: strategy representation, creation and access. Strategy representation is based on the format of the output provided by UPPAAL-TIGA. Namely, a winning strategy generated by the TGA model checker is a set of rules, each of which is composed by a set of state formulae (a pair timeline-token) associated to actions to be performed. Each state usually has two possible actions: one associated to an actual transition (an action leading to a value change on a state variable) with some temporal constraints derived from guards on transitions in the associated TGA network and one special action that requires the system to wait. If a transition is uncontrollable, only a wait action is represented as the value change is supposed to be executed by the environment. In order to enable its actions, each strategy is associated to temporal guards (temporal constraints) and to clock updates (for checking temporal constraints). The software structure of TIGA2EXEC replicates this hierarchy and has been deployed within PLATINUM [15].

TIGA2EXEC workflow is the following: once a plan is generated by PLATINUM, TIGA2EXEC triggers PLAN2TIGA, an encoder capable to translate a flexible plan in a TGA network and provided as input for UPPAAL-TIGA. The model checker verifies the TGA network in order to check dynamic controllability and, if possible, generates a winning strategy. TIGA2EXEC then analyzes the output to check whether the plan is DC or not (i.e., plan either dynamically controllable or not controllable). The strategy is then encoded as a set of software objects triggered by specific conditions (i.e., states and timings) and issuing the corresponding actions defined in the winning strategy. Indeed, a P&S system can access the strategy querying step-by-step TIGA2EXEC about which are the actions to be performed. During each execution step, TIGA2EXEC will perform a cycle to extract actions from the strategy. For each cycle, TIGA2EXEC performs a synchronization phase and a dispatching phase. The synchronization phase is for collecting data from the P&S system and the environment (via external observations) and to check whether the actual plan execution is coherent with the plan specifications. The dispatching phase is about tokens selection and dispatching. Some variables representing the status of the execution are added to the strategy representation as well as some internal clocks for supervising the execution. The last status of the system is considered to track uncontrollable transitions that may occur and, in case, update the internal system status with post conditions for these transitions. In general, TIGA2EXEC can return a list of tokens to be dispatched (dispatching actions until a wait action is found). The internal logic to implement these steps is the following: a) The actual status perceived by the P&S system is compared with the expected status by TIGA2EXEC; b) internal plan clocks are updated; c) Each internal state is compared to the actual status and when they match, an *action trigger* is searched between the rules in the strategy; d) once a match is found, the action is added to the list of actions to be dispatched and post conditions are applied to the internal status. If a match is not found, an action wait is added to the action list and the list is returned. TIGA2EXEC is fully integrated within PLATINUM and can be integrated in any timeline-based P&S system implemented according to the framework given in [7].

We present some preliminary empirical results considering a benchmark domain for planning and scheduling introduced in [4] modelling a satellite operating around a remote planet and performing science and communication activities with uncertain conditions. This experimental analysis is performed to assess the general performance of TIGA2EXEC. In particular, performance evaluation is crucial as in plan-based control, the frequency of each cycle entails the executive system to be able to provide a set of actions to be dispatched within a fixed time. Therefore, the overhead introduced by TIGA2EXEC for analyzing and processing a high number of tokens and relations may explode and be not fully compatible with such execution latencies. To this aim, we consider three set of batches of tests run on a machine endowed with Intel Core i5 (2.4GHz) processor and 8 GB RAM. Each set corresponds to different settings of the same planning domain. Here, we consider the following specific settings: *Domain1* models a set of three (fully uncontrollable) temporal windows for communications with a given planning horizon; *Domain2* extends the previous setting relaxing the temporal constraints on communication windows, i.e., providing more time for communications and, consequently increasing the opportunities for the planner to allocate communication tasks; *Domain3* further extends the previous settings doubling the size of the temporal horizon, i.e., providing the planner with more time to schedule tasks over time. Then, for each domain, we consider a set of problem instances with an increasing number of goals, i.e., an increasing number of science (and communications) tasks to be performed (spanning from 1 to 5) as well as an increasing temporal uncertainty of uncontrollable tasks (spanning from 5 to 30 time seconds). We perform 5 runs of the same problem and consider average times. It is worth noticing that the most complex planning scenario is the one with higher number of goals (5 goals), and higher temporal uncertainty (30 secs).

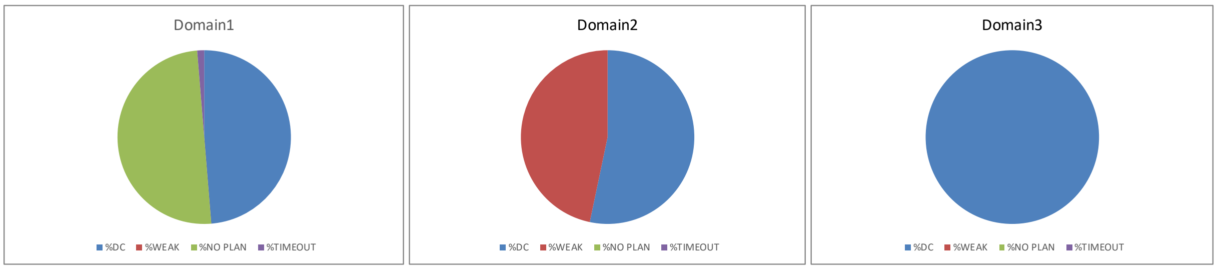

Figure 1: Percentages of controllable plans and failures for the instances on different domains.

Figure 1 depicts for each domain plan generation success rate and, when generated, plan controllability. In *Domain1*, experiments show a high rate of planning failures (50%). The planner is guided by heuristics to minimize plan makespan and, thus, scheduling science and communication tasks as soon as possible. All science/communications are then scheduled during the first temporal window resulting in a plan with small flexibility to deal with temporal uncertainty. Increasing the number of goals, many tasks are planned in a temporally rigid way and often being unable to achieve all the goals (i.e., no valid plan). The planner is able to generate DC plans for problem instances only with a limited number of goals. In *Domain2*, the planner is always able to generate a plan but not fully controllable as almost half of the plans were only weakly controllable. In order to achieve the goals within the given planning horizon, the planner is forced to tightly schedule tasks and, again, only when dealing with a small number of goals the planner was able to keep DC. In the other instances, the planner produces weakly controllable plans. In *Domain3*, a larger planning horizon allows the planner to always generate DC plans and to complete plans execution.

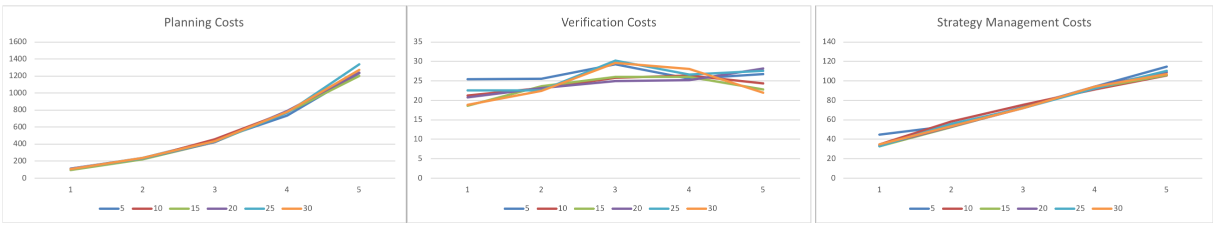

Figure 2: Average costs (in milliseconds) for: planning, verification and strategy synthesis, strategy management.

Figure 2 shows the average temporal costs for, respectively, planning, verification (PLAN2TIGA + UPPAAL-TIGA) and strategy management when TIGA2EXEC is involved for the problem instances considered in *Domain3* varying the number of goals and the temporal uncertainty. First, it is clear that verification and strategy management costs are compatible with planning latencies. Then, an increase of temporal uncertainty does not affect costs as planning and verification systems are effective in managing temporal flexibility. For PLATINUM and TIGA2EXEC, the most important element influencing costs is the number of goals. In fact, increasing the number of goals, generated plans contains more tokens and control strategies are larger (in terms of control rules). This entails larger times for strategy management.

4 Conclusions

This paper presents TIGA2EXEC, a new software prototype, for dynamic controllable execution of timeline-based plans leveraging the integration of P&S and Model Checking techniques. TIGA2EXEC implements a work chain that i) encodes flexible plans as a network of TGA, ii) exploits a verification tool to check dynamic controllability property for flexible plans and, possibly, generate a dynamic execution strategy and iii) implements such strategy as a robust plan execution controller for P&S systems. TIGA2EXEC is deployed in PLATINUM, a timeline-based planning system to control plan execution guaranteeing dynamic controllability. A preliminary experimental evaluation is also presented.

Acnkowledgments

CNR authors are partially supported by the E.C. and ShareWork project (H2020 – Factories of the Future G.A. nr. 820807).

References

[1] J. F. Allen. Maintaining knowledge about temporal intervals. *Commun. ACM*, 26(11):832–843, 1983.

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

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

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

[5] M. Cialdea Mayer and A. Orlandini. An executable semantics of flexible plans in terms of timed game automata. In *The 22nd International Symposium on Temporal Representation and Reasoning (TIME).* IEEE, 2015.

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

[7] M. Cialdea Mayer, A. Orlandini, and A. Umbrico. Planning and execution with flexible timelines: a formal account. *Acta Informatica*, 53(6-8):649–680, 2016.

[8] A. Jonsson, P. Morris, N. Muscettola, K. Rajan, and B. Smith. Planning in Interplanetary Space: Theory and Practice. In *AIPS-00. Proc. of the 5th Int. Conf. on AI Planning and Scheduling*, 2000.

[9] S. Lemai and F. Ingrand. Interleaving Temporal Planning and Execution in Robotics Domains. In *AAAI-04*, pages 617–622, 2004.

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

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

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

[13] A. Orlandini, A. Finzi, A. Cesta, and S. 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.

[14] 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.

[15] A. Umbrico, A. Cesta, M. Cialdea Mayer, and A. Orlandini. Platinum: A new framework for planning and acting. In F. Esposito, R. Basili, S. Ferilli, and F. A. Lisi, editors, *AI*IA 2017 Advances in Artificial Intelligence*, pages 498–512, Cham, 2017. Springer International Publishing.