Formal Methods-Based Maximally Satisfying Action Planning

Jana Tumova, Alejandro Marzinotto, Dimos V. Dimarogonas, Danica Kragic

***Abstract*— We focus on synthesis and execution of action plans for an autonomous robot, where the action refers to a "simple" motion or manipulation task, such as "go from A to B", or "grasp a ball". Due to imprecisions in the robot's sensors and actuators, an action execution attempt might succeed as well as fail, resulting in a possible infeasibility of the desired robot's goal. In this work, we propose a formal methods-based framework that allows to cope with such situations; to express a complex robot mission, to automatically compute a maximally-satisfying reactive action plan, and to interface the action plan with the robot's low-level controllers.**

I. Introduction

Recent research has shown that temporal logics can formalize many complex, high-level, recurrent task, such as surveillance ("periodically visit region A"), sequencing ("visit region A, then B, then C"), safety ("always avoid region D"), and their combinations. Interfaces have been designed between temporal logics and user-friendly specification languages, such as structured English, or graphical representations. Leveraging ideas from formal verification, algorithms have been developed to automatically synthesize a discrete plan that guarantees the satisfaction of a given temporal logic goal. Various discrete robot models and high-level plans have been coupled with the robot's continuous workspace and controllers. All in all, these ingredients advance the formal methods-based approach to translating a user-input high-level mission into a robot's provable mission satisfaction.

While the approach has gained a considerable amount of attention in robot motion planning context [1], [2], the literature focused on *motion and manipulation* planning has been, to our best knowledge, limited to several standalone studies [3], [4]. In this work, we aim to enhance the temporal logic planning of "where to go" with "what to do" there. We devote specific focus to the uncertain outcome of a motion or manipulation task execution; we consider that such an action can succeed or fail.

Related work includes techniques for reactive temporal logic planning e.g., synthesis for nondeterministic systems, or general-reactivity goals [2]. Other closely related works aim at planning under unsatisfiable temporal logic goals, where the authors focus on least-violating planning with respect to a proposed metric [5], [6].

The authors are with the Centre for Autonomous Systems, Royal Institute of Technology (KTH), SE-100 44 Stockholm, Sweden. J. Tumova and D. Dimarogonas are also with the Automatic Control Lab and the ACCESS Linnaeus Center. A. Marzinotto and D. Kragic are also with Computer Vision and Active Perception Lab. This work was supported by the EU STREP RECONFIG. e-mail: {tumova|almc|dimos|dani}@kth.se.

A. Objectives

Our goal is to propose a formal methods-based framework that 1) bridges the high-level plan with the low-level motion, grasping, and other controllers, 2) is modular and extensible to handle different robots and their capabilities, 3) is reactive and able to cope with the unreliability of the robot's sensors and actuators leading to failures of its actions, and 4) guarantees that the temporal logic task is met as closely as possible in case it cannot be met completely.

This extended abstract summarizes the key ideas of the proposed solution and we include an illustrative example of a NAO robot executing walking, ball grasping and ball dropping motion primitives. The details can be found in the full version of this paper [7].

II. Our Approach

Our approach to the above problem is inspired by the existing formal methods-based hierarchical motion planning framework (see, e.g. relevant overview papers [1], [2]). Similarly as in some related literature we assume that the robot's states and action capabilities are at the highest level of abstraction captured through a finite, discrete *state-transition system*. Note that this does not mean that the transition system is input explicitly, it can be as well automatically generated from other robot's models [1], [2]. The set of states of the transition system encode the possible states of the robot, such as its position in the environment, or the object it is carrying. The transitions of the system abstract the action primitives, such as "move from position A to position B", or "grasp a ball" and the corresponding evolution of the robot's states upon a successful execution of an action primitive. Due to imprecisions in robot's sensors and actuators, each attempt to execute an action might result into a success (transiting into another state) or a failure (staying in the same state). As the specification language, we use a surveillance fragment of the *State-Event Linear Temporal Logic* (SE-LTL) allowing to impose requirements and restrictions on a long-term evolution of both the robot's states and its actions.

The proposed solution comprises two standalone steps: (1) maximally satisfying high-level planning in which we synthesize a reactive, discrete plan that satisfies the formula as closely as possible, and (2) low-level plan execution in which we translate the plan into a robot's hybrid controller.

A. Maximally Satisfying High-level Planning

To synthesize a maximally satisfying plan, we first formally define a measure of the *level of noncompliance* of a robot's behavior (i.e. a trace of the transition system) with respect to the given SE-LTL formula. Following the

automata-based approach to model checking and similar ideas as in [5], we translate the SE-LTL formula into an equivalent Büchi automaton. We construct a specialized weighted product automaton that captures not only the traces of the transition systems that satisfy the Büchi automaton (hence also the SE-LTL formula), but also the level of noncompliance of the individual traces through its weights. We prove a correspondence between the length of the lasso-shaped accepting runs in the product automaton and the level of noncompliance of the traces of the transition system. The remainder of the solution builds on a systematic exploration of the product automaton via standard graph algorithms.

The resulting discrete plan is reactive. It is formalized through an *Input/Output (I/O) automaton* that reads as an input a success/failure status of the previous action execution attempt and outputs the next action to be executed.

B. Action Plan Implementation

The I/O automaton obtained in the previous step serves a discrete action plan for an abstract model of the considered robotic system. However its application in a dynamical system itself is not straightforward. We propose an automatic translation of the I/O automaton into a *Behavior Tree (BT)* that maps abstract action primitives onto their respective continuous controllers and hence serves as a bridge between the high-level planning and the low-level robot control layers.

The idea of using BTs as a hybrid control interface has been recently proposed in [8], resulting in an open source BT library for the Robot Operating System (ROS) [9]. The main feature indicating that BTs are suitable for a middle-layer controller representation is their modularity; every sub-tree of a BT can be treated as a standalone entity that represents a certain action or a task. BTs are built using sub-tree blocks in a hierarchical fashion, subsuming thus multiple levels of abstraction and allowing to easily replace an abstract action/task node with a concrete action/task implementation [8]. Additionally, BTs are equipped with control-flow nodes to capture various conditional, or sequential sub-tree executions, hence making BTs an appropriate formalism for reactive controllers.

With the translation from I/O automata to BTs, we conclude the proposed solution to maximally satisfying planning and its execution by a robot.

III. Experiments

We implemented the proposed solution in Robot Operating System (ROS) and tested it in a NAO robot testbed illustrated in Fig. 1. To demonstrate the results of our approach, we consider two different SE-LTL tasks involving motion between regions, grasping and dropping a ball. The grasping action has shown as the most critical, often failing one. Thus, for simplicity of presentation, in Fig. 1 we depict resulting trajectories while considering only grasping action failures.

IV. Future Work

As a future work, we will explore an option to update the BT upon a run of a system instead of its offline generation.

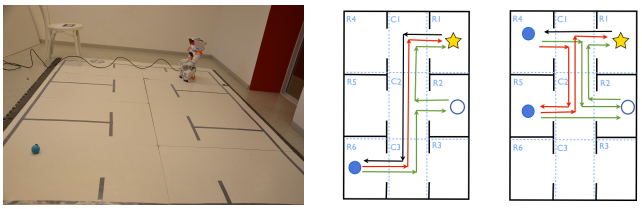

Fig. 1: Left: Example of a NAO robot's workspace partitioned into 9 regions; 6 rooms and 3 corridor regions. We assume the following set of action and motion primitives: r, b, l, t (move to the neighboring region on the right, bottom, left, or top, respectively), $grab$ (grab a ball), $drop$ (drop the ball), and sit. The robot's state is compound of the region the robot is present at and whether or not it holds a ball. The robot is not allowed to cross a black line, and therefore the motion primitives r, b, l, t are not enabled in all the states. **Middle:** Illustration of the (cyclic) solution for a task "periodically grab a ball in R_6 and drop it in R_2": $\mathsf{GF}(R_4 \wedge grab \wedge \mathsf{F}(R_2 \wedge drop)) \wedge \mathsf{GF}\, sit$. The yellow star represents the sit action. The filled and empty circles depict $grasp$ and $drop$, respectively. The black arrows illustrate the behavior before $grasp$ is attempted. The continuation of one cycle after a successful and a failed $grasp$ is in green and red, respectively. Notice, that after a grasp failure, room R_2 is not visited as $R_2 \wedge drop$ cannot be satisfied. **Right:** Illustration of the task "periodically grab a ball in R_4 or in R_5 and drop it in R_2": $\mathsf{GF}((R_4 \wedge grab \vee R_5 \wedge grab) \wedge \mathsf{F}(R_2 \wedge drop)) \wedge \mathsf{GF}\, sit$.

Furthermore, we will focus on finer classification of actions than success or failure; we plan to introduce a probabilistic measure of an action success and algorithms to extend the level of satisfaction to the expected one.

Acknowledgement: We would like to thank Meng Guo and Michele Colledanchise for their help on the NAO testbed.

References

[1] A. Bhatia, M. R. Maly, L. E. Kavraki, and M. Y. Vardi, "Motion planning with complex goals," *Robotics Automation Magazine, IEEE*, vol. 18, no. 3, pp. 55 –64, 2011.

[2] H. Kress-Gazit, T. Wongpiromsarn, and U. Topcu, "Motion planning with complex goals," *Robotics Automation Magazine, IEEE*, vol. 18, no. 3, pp. 65 –74, 2011.

[3] M. Guo, K. H. Johansson, and D. V. Dimarogonas, "Motion and action planning under LTL specifications using navigation functions and action description language," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2013, pp. 240–245.

[4] S. Chinchali, S. C. Livingston, U. Topcu, J. W. Burdick, and R. M. Murray, "Towards formal synthesis of reactive controllers for dexterous robotic manipulation," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2012, pp. 5183–5189.

[5] J. Tumova, G. C. Hall, S. Karaman, E. Frazzoli, and D. Rus, "Least-violating control strategy synthesis with safety rules," in *Proceedings of the 16th International Conference on Hybrid Systems: Computation and Control (HSCC)*. ACM, 2013, pp. 1–10.

[6] M. R. Maly, M. Lahijanian, L. E. Kavraki, H. Kress-Gazit, and M. Y. Vardi, "Iterative temporal motion planning for hybrid systems in partially unknown environments," in *Proceedings of the 16th International Conference on Hybrid Systems: Computation and Control (HSCC)*. ACM, 2013, pp. 353–362.

[7] J. Tumova, A. Marzinotto, D. V. Dimarogonas, and D. Kragic, "Maximally satisfying LTL action planning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2014. To appear.

[8] A. Marzinotto, M. Colledanchise, C. Smith, and P. Ögren, "Towards a unified behavior trees framework for robot control," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 5420–5427.

[9] A. Marzinotto, M. Colledanchise, and I. Tjernberg. (2013) Behavior trees library for the robot operating system (ROS). [Online]. Available: https://github.com/almc/behavior_trees