Safe Behavior Specification and Planning for Autonomous Robotic Systems in Uncertain Environments

Doctoral Consortium

Jan Vermaelen
imec-DistriNet, KU Leuven
3001 Leuven, Belgium
jan.vermaelen@cs.kuleuven.be

ABSTRACT

The safe operation of an autonomous robotic system is a complex endeavor, with decision-making being a pivotal element. Formal analysis of decision-making logic can be done using model checking or other formal verification approaches. However, the non-deterministic nature of realistic environments can make these approaches impractical and troublesome. Constraint-based planning approaches have been shown to be capable of generating policies for a system to reach its goals while abiding safety constraints.

We extend such a constraint-based approach, Tumato, to support non-deterministic outcomes of actions. Actions have one specific intended result, yet can be modeled to have alternative outcomes that may realistically occur. The adapted Tumato solver generates a policy that enables the system to reach its goals in a safe manner even when alternative outcomes of actions occur. Furthermore, we introduce a purely declarative way of defining safety in Tumato, increasing its expressiveness and facilitating the specification of actual safe behavior. Finally, we add cost or duration values to actions, enabling the solver to restore safety when necessary in the most preferred way.

KEYWORDS

Autonomous Systems; Behavior Specification; Safe and Robust Planning; Constraint Programming; Tumato

ACM Reference Format:
Jan Vermaelen. 2023. Safe Behavior Specification and Planning for Autonomous Robotic Systems in Uncertain Environments: Doctoral Consortium. In *Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), London, United Kingdom, May 29 – June 2, 2023,* IFAAMAS, 3 pages.

1 INTRODUCTION

Autonomous robotic systems are becoming increasingly popular in both industry and households. The number and complexity of tasks that they are expected to execute are expanding. Generating behavior that is both productive (goal-oriented) and safe is far from trivial. It requires taking into account the robot's available actions, information about the environment, and desired goals, as well as additional well-defined safety constraints. If a plan for such safe and productive behavior can be generated, it is sound by construction.

We investigate a constraint-based approach to deal with foreseeable non-deterministic transitions while guaranteeing safety. For this purpose, we extend Tumato, a constraint-based planning framework by Hoang Tung Dinh et al. [2]. Specifically, we support the explicit declarative specification of safety conditions as well as account for foreseeable non-deterministic transitions. We will fully elaborate on this extension and its language in a separate paper.

The intended approach combines classical planning using states, actions, and goals with constraint programming to explicitly enforce safety. The set of Constraint Satisfaction Problems (CSPs) [12] yielding from the specification can be solved offline. The resulting solution maps every state to the actions that have to be executed in that state. If such a sound and complete (and hence productive, safe, and robust) policy exists, it will be found by the constraint solver. The obtained policy can safely be used at run-time without requiring online re-planning. If no such policy exists, the troublesome state is provided as feedback to the user.

2 GENERATING SAFE ROBOT BEHAVIOR

Traditionally, the behavior of robots has been defined manually. Finite State Machines (FSMs) are most often used to represent a robot's behavior [6, 10], even in present times. However, FSMs are known not to cope well with the increasing complexity of the behavior. Furthermore, one has to rely on simulation and verification approaches to guarantee that the behavior effectively meets the requirements. This problem can partly be solved by automatically generating the behavior based on a model of the system, along with a representation of the desired requirements.

Different specification languages have been proposed, each with corresponding planners. Linear Temporal Logic (LTL) [5] is often used in robotics. Techniques exist to generate FSMs based on (fragments of) LTL [9, 14]. Since all contingencies can be taken into account, the generated behavior will be sound and complete. The main drawback of LTL approaches is their computational complexity. Two other examples are Temporal Action Logic (TAL) [3, 4] and, more generic, Planning Domain Definition Language (PDDL) [7]. Both rely on replanning at run-time to cope with contingencies. They do not guarantee completeness of the behavior since the replanning could fail due to an unrealizable specification. This lack of completeness would only be detected at run-time.

Hoang Tung Dinh et al. [2] obtain sound and complete behavior via constraint programming. The behavior specification enables enforcing safe behavior via *reaction rules*, allowing the specification of reactive behavior. While this planning approach assumes a deterministic environment, preliminary experiments show that it can deal with uncertainty to a certain extent for some systems. We aim to achieve the level of robustness necessary to deal with

Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), A. Ricci, W. Yeoh, N. Agmon, B. An (eds.), May 29 – June 2, 2023, London, United Kingdom. © 2023 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

robotic systems in practice. The constraint programming approach we investigate builds upon the original work on Tumato [2].

To a certain extent, robustness can be obtained by explicitly dealing with non-determinism. Planning with non-deterministic models is (one of the aspects) covered in the book *Automated Planning and Acting* by Malik Ghallab et al. [8]. Following that approach, the planning can try to use the non-deterministic effects of actions to reach the goal. Unlike this work, we opt to define one intended effect for each action, as well as a number of alternative (less likely) outcomes. This approach is more closely related to the behavior of actions in practical planning problems.

Before dedicating ourselves to this constraint-based approach, we have surveyed existing frameworks, combining safe and robust planning [13]. The use of Markov Decision Processes (MDPs) [11] for probabilistic planning and, to a smaller extent, Simple Temporal Networks (STNs) [1] for temporal scheduling was investigated. Further, we are aware of more recent and more practical work on safe planning and control in robotics. For the conciseness of this section, however, we have focused on the above-mentioned (more formal) pillars.

3 CURRENT WORK

As mentioned in Section 1, the specification of a system contains the information about the environment, the actions the robot can execute, the goal of the system, and a set of safety rules. The output of the solver is a policy, mapping every state to the action that has to be executed in that state. Since this is a *complete* policy, it can be used at run-time without the need for online re-planning.

Since the environment, and hence the effects of actions, are often not deterministic in practical robotic applications, the plan must be sufficiently robust to deal with this kind of uncertainty. Ideally, all contingencies are taken into account. A first step is indeed to pursue a *complete* policy. For each state in which the system could be, the policy must provide the actions to execute next. We maintain this strong feature of Tumato's original approach. In a second step, we take the uncertainty into account by allowing the effects of actions to be modeled in a non-deterministic way. In our approach, we assume that each action has one intended outcome, called the *nominal* effect of the action. Additionally, each action can have a number of *alternative* effects, which *could* emerge instead of the nominal one, but they are not intentional. An action with one set of alternative effects is shown in Figure 1. For the productivity aspect of planning, only the nominal effect is relevant. When dealing with safety, also the alternative effects must be taken into account.

```
action deliver_workpiece
duration: 3
controlled resources: MOTORS, CONVEYOR
preconditions:
S_Location is workstation_2, S_Load is loaded
nominal effects:
S_Conveyor is on, S_Load is free
alternative effects:
S_Conveyor is on, S_Load is loaded
```

Figure 1: A code snippet showing the action *deliver_workpiece*

To guarantee safety, we enable specifying declarative safety conditions explicitly as *State Rules*. One example is shown in Figure 2. In the original version of Tumato, to obtain a similar result, one has to list all state-action combinations that could lead to unsafe states separately, which is error-prone and inflexible when the specification of the system evolves.

```
BEGIN STATE RULES
rule: IF S_Location is corridor OR S_Location is
  charger THEN S_Conveyor is off
END STATE RULES
```

Figure 2: A code snippet showing one State Rule

Finally, if the system arrives in an *unforeseen* state (due to an external force), the planner will make sure that the policy contains instructions on how to get back on (safe) track to the goal immediately. If multiple such instructions are possible, the planner is capable of selecting the preferred one based on a metric such as cost, duration, or risk.

In our work, the driving use case is an Autonomous Mobile Robot (AMR) operating in a highly automated demo factory. The AMR, depicted in Figure 3, can navigate across the factory and dock to different workstations or machines to pick up and deliver workpieces. Since human agents and AMR share the factory floor, adequate safety guarantees are required.

4 PROBLEMS AND FUTURE WORK

Despite the promising combination of uncertainty and safety with the constraint-based planning approach, challenges remain open for further investigation.

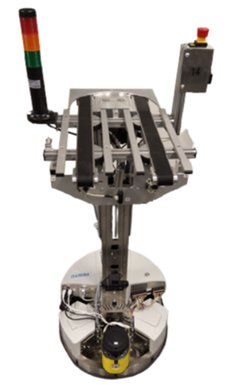
Figure 3: The AMR

The new safety rules currently enforce that the next state (or rather, any foreseeable outcome of the executed actions) is safe. When multiple sets of actions are available to *maintain* safety, the planner considers them equally valid. When multiple sets of actions exist to *restore* safety, the planner is capable of choosing the best one based on a value, such as duration or cost, assigned to each action. When no actions are available to restore safety immediately, the planner will notify the user, indicating for which state safety can not be obtained. Although this is a desirable approach, the case study revealed another interesting challenge. Future research could explore how allowing multiple successive (sets of) actions in unsafe states could be required to restore safety. The same values that get assigned to actions should be applied to obtain the preferred solution. The concrete semantics of this extension are also to be defined by this future research. Further, in practice, different safety constraints relate to different severities. In the same way that actions can be assigned a specific value, also the safety rules could be weighted to enable the constraint solver to find the *best* overall solution. Finally, an empirical study will be conducted concerning the practical use of the adapted planner.

ACKNOWLEDGMENTS

This research is partially funded by the Research Fund KU Leuven.

REFERENCES

[1] Rina Dechter, Itay Meiri, and Judea Pearl. 1991. Temporal constraint networks. *Artificial intelligence* 49, 1-3 (1991), 61–95.

[2] Hoang Tung Dinh, Mario Henrique Cruz Torres, and Tom Holvoet. 2017. Sound and complete reactive UAV behavior using constraint programming. https://lirias.kuleuven.be/retrieve/470086

[3] Patrick Doherty, Joakim Gustafsson, Lars Karlsson, and Jonas Kvarnström. 1998. Temporal action logics (TAL) language specification and tutorial. *Electronic Transactions on Artificial Intelligence (http://www.etaij.org)* 3 (1998).

[4] Patrick Doherty and Jonas Kvarnstram. 2001. TALplanner: A temporal logic-based planner. *AI Magazine* 22, 3 (2001), 95–95.

[5] E Allen Emerson. 1990. Temporal and modal logic. In *Formal Models and Semantics*. Elsevier, 995–1072.

[6] Maksym Figat, Cezary Zielinski, and Rene Hexel. 2017. FSM based specification of robot control system activities. 193–198. https://doi.org/10.1109/RoMoCo.2017.8003912

[7] Alfonso E Gerevini, Patrik Haslum, Derek Long, Alessandro Saetti, and Yannis Dimopoulos. 2009. Deterministic planning in the fifth international planning competition: PDDL3 and experimental evaluation of the planners. *Artificial Intelligence* 173, 5-6 (2009), 619–668.

[8] Malik Ghallab, Dana Nau, and Paolo Traverso. 2016. *Automated Planning and Acting* (1st ed.). Cambridge University Press, USA.

[9] Spyros Maniatopoulos, Philipp Schillinger, Vitchyr Pong, David C Conner, and Hadas Kress-Gazit. 2016. Reactive high-level behavior synthesis for an atlas humanoid robot. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 4192–4199.

[10] Hai Dai Nguyen, Matei T. Ciocarlie, Kaijen Hsiao, and Charles C. Kemp. 2013. ROS Commander : Flexible Behavior Creation for Home Robots.

[11] Martin L Puterman. 2014. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons.

[12] Edward Tsang. 2014. *Foundations of constraint satisfaction: the classic text*. BoD–Books on Demand.

[13] Jan Vermaelen, Hoang Tung Dinh, and Tom Holvoet. 2020. A survey on probabilistic planning and temporal scheduling with safety guarantees. In *ICAPS Workshop on Planning and Robotics*. ICAPS Workshop on Planning and Robotics.

[14] Tichakorn Wongpiromsarn, Ufuk Topcu, and Richard M Murray. 2013. Synthesis of control protocols for autonomous systems. *Unmanned Systems* 1, 01 (2013), 21–39.