Commentary on "On-board Timeline Validation and Repair: A Feasibility Study" by Fox, et al.

Mark S. Boddy
Adventium Labs
Minneapolis, MN
USA
mark.boddy@adventiumlabs.org

1 Introduction

In this paper, the authors report on work done for an ESA project entitled "Mars Mission Onboard Planning and Scheduling" (MMOPS), exploring a limited form of onboard reasoning for space-related robotic systems. Striking a middle ground between ground-based and on-board planning, the authors describe experiments with an on-board "planning assistant" which repairs plans using "Timeline Validation, Control, and Repair." The application domain for which results are reported is a simulation of Beagle 2. This work touches on several of the most important issues involved in designing, building, and fielding high-level autonomous or semi-autonomous control systems for complex robots. Three of these issues are discussed briefly, below.

2 Trusted Autonomy

The authors describe the process of gaining the acceptance of more "intelligent" software in controlling spacecraft, rovers, and other complex automation as primarily cultural, involving gradually exposing mission personnel to more and more capable systems. Judging by past history, they're right. For example, the very complex software of commercial aircraft Flight Management Systems is certified according to a standard called "DO-178B." What this standard specifies is primarily a software development and documentation process, not a formal analysis. At JPL, the planning and execution system Claraty, including the Casper planner, is increasingly being relied on for on-board intelligent planning and control. This is less due to any formal proof of correctness or reliability, and more to increasing familiarity and an increasing track record of reliability.

Seen in this light, the authors' decision to implement a more limited form of plan modification fits perfectly in providing an incremental step into greater autonomy, from which other steps can be taken. However, it will be interesting to see what happens when this system or some successor to it is considered for use on a mission. The difficulty of establishing trust is one of the main reasons that autonomous systems have largely been confined to low-risk, sandbox experiments such as Remote Agent on DS-1 or the Autonomous Sciencecraft experiment on EO-1. To pick another example from many available, the MER rovers (and, for that matter, their predecessor Sojourner) are capable of considerably more autonomous operation than has been permitted (Pedersen *et al.* 2003).

3 What is Planning, Anyway?

The planning model used in CONTOOL has roots in work done by two of the authors (Fox & Long) on extending PDDL to permit representation and reasoning about complex forms of continuous constraints. However, in the process of addressing the requirements of this application, the authors have added features that will look very familiar to those who come at this kind of problem from other angles. In particular, work on either constraint-based planning and scheduling or HTN planning have long incorporated features such as plan fragments, ordering constraints, explicit resources, and execution windows.

Furthermore, the process of timeline validation, control, and repair implemented in CONTOOL bears a strong similarity to the functioning of the Casper planner, for example in the Autonomous Sciencecraft Experiment on EO-1. Casper plans by iteratively adding and removing things that look very much like the plan fragments described in this paper, tracking resource subscriptions, also as described here. CONTOOL appears to differ from Casper in that CONTOOL's validation will handle a more complex continuous model, and the set of timeline operations is is much more limited. As discussed above, the constraints on timeline manipulation are deliberate, an attempt to provide a small enough step from fully manual control to be acceptable to mission controllers. But this difference more in degree than in nature does beg the question of which of these systems is or is not doing "planning."

4 Exploiting the Structure of Application Domains

The implementation of this system within the context of a specific mission and a reasonably faithful mission simulator is one of the strengths of this work. Evaluation against problems with a real or at least a realistic structure means that the authors can make claims for utility and performance that do not have to be qualified: for this domain and these instances, the system *works*. However, working against this strength, the authors claim a broader applicability for CONTOOL, without any real discussion of how qualitative differences in application domains will affect the tool's configuration or utility. For example, the appropriate interpretations of plan robustness and repair will differ quite a bit among a rover, an orbiter, and a deep-space probe. Robustness is relevant to the authors' notion of "plan failure isolation" in that one can characterize a plan as more or less robust depending on both how likely it is that a repair will be needed, and how extensive a repair is likely to be required.

Domain characteristics will also affect the behavior of the system in the face of over- and under-subscription. Consider the differences between a cautiously-managed spacecraft in cruise mode on the one hand, and space shuttle refurbishment on the other. For the spacecraft, there is relatively little in the way of persistent state information (physical configuration), and that little is not usually too expensive to change (rotate an instrument platform). "Opportunities" can easily be added to the timeline to fill in gaps, supporting an approach in which a very conservative timeline is generated, then filled in with opportunities as time becomes available. For shuttle refurbishment, the problem is very different. State has very broad effects and can be very hard to change. Execution uncertainties are very large (what will you find when you take off that panel?). Idle time is extremely expensive. So, they generate an enormously oversubscribed schedule, whose primary characteristic is that there is always something to do next. Schedules are regenerated frequently, as new information becomes available. This is almost the exact opposite of the approach described in this paper, but it works for that domain.

5 Summary and Conclusion

This paper presents an interesting application of planning and scheduling technology in limited forms of plan repair, suitable for on-board execution. The techniques employed are not particularly novel. However, the implementation and evaluation have been driven by the requirements of a real domain (this is clear, for example, in the discussion of plan fragments). In addition, CONTOOL combines several threads of work that have not historically been well-integrated, drawing from classical planning, constraint-based planning and scheduling, timeline scheduling, and the ongoing extension of PDDL to ever-more-expressive dynamical models. We are still a ways from seeing all of these disparate threads integrated in a coherent framework, but, as is evident in this paper, the friction of practical applications appear to be pushing us in that direction.

References

L. Pedersen, D. Kortenkamp, D. Wettergreen, and I. Nourbakhsh. A survey of space robotics. In *Proceedings of the International Symposium on Artificial Intelligence, Robotics, and Automation for Space*, 2003.