Integrated Symbolic Planning in the Tidyup-Robot Project

Christian Dornhege and **Andreas Hertle**
Department of Computer Science
Georges-Köhler-Allee 52
University of Freiburg, Germany
email: {dornhege, hertle}@informatik.uni-freiburg.de

Abstract

We present the integration of our symbolic planner as the high-level executive in the Tidyup-Robot project. Tidyup-Robot deals with mobile manipulation scenarios in a household setting. We introduce our system architecture and report on issues and advantages observed during development and deployment.

Introduction

When designing a complex mobile manipulation robotic system, high-level decision making is a central task. Especially when multiple heterogeneous actions are added, maintaining hand-written scripts or other approaches, like state machine based control, become cumbersome or very complex. A symbolic task planner solves abstract planning problems often arising in such scenarios efficiently. However, as those planners usually operate on an abstracted symbolic domain description, it is an important issue to properly integrate a symbolic planner into a system acting in the real-world.

The *Tidyup-Robot* project aims to solve mobile manipulation tasks with the PR2 robot by Willow Garage. The goal is to tidy objects in the world, i.e., to bring them back to the place, where they belong. This entails to be able to detect objects, grasp and place them, open doors and as an additional task wipe the spots where objects were found (see Fig. 1).

In this paper, we focus on the integration of the symbolic task planner *Temporal Fast Downward/Modules* (TFD/M) to facilitate high-level control. We use two key concepts: First, the planner runs in a observation-monitoring-execution loop to be able to react to the possibly changing state of the world. Second, the planner uses semantic attachments that determine predicate semantics during the planning process by an external procedure.

Earlier approaches for combined task and motion planning extended the planner Metric-FF to support manipulation planning (Cambon, Alami, and Gravot 2009). Our work provides a domain-independent interface to separate the planner from the specific application (Dornhege et al. 2009) by allowing to compute predicate semantics externally. More recently Gregory et al. (2012) developed a system to integrate first-order theories into a generic planner.

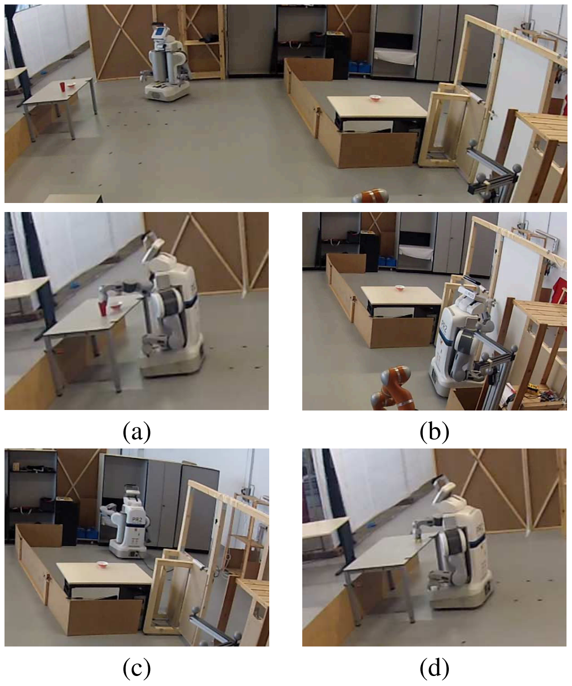

Figure 1: Example scenes from the PR2 operating in the Tidyup-Robot domain. The robot collects any cups and bowls from the two tables (a). Next, it opens the door to the other room (b) to put the objects into the shelf (c). Finally, the robot wipes the tables where the objects were initially located (d). A video is available at: `http://www.youtube.com/watch?v=pTSz2RBZ2wA`

Kaelbling and Lozano-Perez (2011) use a hierarchical decomposition of the planning process to quickly find executable plan prefixes also integrating motion planners in the symbolic planning. Other approaches for combining symbolic and geometric reasoning use plan-based action control to implement sophisticated symbolic actions (Kresse and Beetz 2012; Beetz et al. 2001).

In the following sections, we will describe those concepts in more detail and then discuss our experiences within the

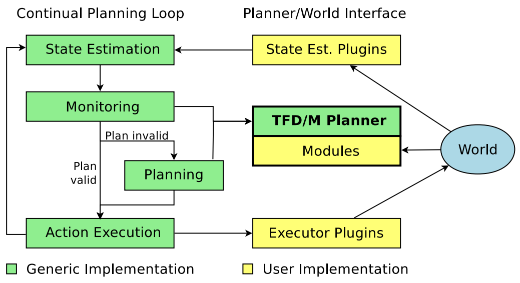

Figure 2: This figure gives an overview of the planner's integration in the continual planning loop.

Tidyup-Robot project identifying key issues with the integration of symbolic planning in a robotics context and conclude with solution concepts and possible future improvements.

Continual Planning

In a real-world system the first plan cannot always be assumed to fulfill the goal when blindly executed. The system must be able to deal with unexpected events, such as execution failures, and common events like detecting a new object that is not in the current state. Therefore, the planning process runs in an observation, monitoring and execution loop (see Fig. 2), based on continual planning. In each iteration the current state is merged with observations of the world. Then, the current plan is monitored, i.e., it is determined, if executing the plan leads to the goal, given the current state. If that is not the case, a new plan is produced. In our case, we use the TFD/M planner to perform monitoring and planning. For monitoring, we only perform applicability tests of the current plan's actions. As a special case, we gain a generic goal test: If the empty plan fulfills the goal, we are in a goal state. After monitoring, the first action of the plan is executed and removed from the plan and the loop continues until the goal is reached or no plan can be found.

The generic continual planning loop is implemented domain-independently. Integration with a specific system needs two interfaces: One to estimate and update the symbolic state from real-world observations and another to be able to execute symbolic actions. Both interfaces are provided by a plugin-based architecture. Multiple accumulating *StateEstimator* plugins can be defined and each action in the domain description requires an *ActionExecutor* plugin. Besides actually executing the actions, *ActionExecutor* plugins also update the state depending on the action outcome. This allows, for example, to implement and plan for an explicit sensing action, such as object detection. In our case the `detect-objects` action inserts perceived objects into the state. Replanning will be triggered automatically, if the objects are relevant for the goal.

Planning with Semantic Attachments

Symbolic planners work with an abstracted logic-based world state. Such an abstraction can lead to over-approximations, which in turn prevents the plan from being executed. Consider a grasp action that has no information

```
(:durative-action putdown-object
 :parameters (?l - loc ?o - obj
              ?s - surf ?a - arm)
 :duration (= ?duration 5.0)
 :condition
  (and (grasped ?o ?a) (...)
   ([canPutdown ?o ?a ?s ?l]))
 :effect
  (and (not (grasped ?o ?a)) (...)
   ([updatePutdownPose ?o ?a ?s ?l])))
```

Figure 3: This excerpt from the Tidyup-Robot PDDL domain shows the `putdown-object` action. Semantic attachments are denoted in square brackets. Temporal qualifier and additional conditions and effects were omitted.

whether a kinematic solution exists to reach an object. We integrate geometric reasoning into symbolic planning with semantic attachments (Dornhege et al. 2009). They allow to attach external semantics to predicates in the symbolic domain description in a domain-independent way. We extended the standard Planning Domain Definition Language (PDDL) (Fox and Long 2003) to support semantic attachments implemented as modules. In PDDL/M external semantics are given by dynamically loadable libraries. Besides applicability testing, there are also modules for cost computation.

Once defined in a domain, modules are used like normal predicates in action definitions. Within the planner, module calls are made precisely when they are needed. Applicability testing will naturally evaluate predicates' truth values. If there is a semantic attachment to a predicate, instead of reading a value from the state, the appropriate module is called. This is handled similarly for cost computation. TFD supports PDDL 2.1 level 2 and thus the planner state contains logical and numerical fluents. A callback interface is passed that allows to retrieve the current state and compute the correct semantics based on that.

Implementation in the Tidyup-Robot Project

For the mobile manipulation scenario in the *Tidyup-Robot* project we supply planner modules for navigation and manipulation. The navigation module calls a motion planner with full body collision checking to determine if there is a path and compute its cost. The manipulation module determines whether an object (e.g. a cup) held by the robot can be placed on a specified surface. A free 6D pose for the object on the surface is determined and a trajectory planner computes a safe arm trajectory. This pose is also updated when the planner applies the action when generating successors in a state and is considered for collision checking during trajectory planning in subsequent module calls. Figure 3 shows the PDDL/M notation of semantic attachments which invoke the manipulation modules.

When the planner checks the applicability of the `putdown-object` action, the `canPutdown` module receives parameters specifying the location of the robot, the object and the arm holding it and the destination surface.

Since the applicability of this action needs to be determined in a future state, where the robot would have interacted with the world, the module retrieves the coordinates of the robot location and possible objects on the surface via the supplied callback interface from the planner state. A 3D scene of this future state is constructed and passed on to the trajectory planner. If a safe trajectory is found, then the action is applicable in that state.

Calling geometric planners via semantic attachments is an expensive operation. The task planner might ask similar queries from different internal states. However, only part of the full state might be relevant for the geometric computation. For example `canPutdown` is only interested in objects' locations on the target surface. Only the module implementation can know what part of the planner state is relevant. Therefore, an intelligent internal caching strategy is crucial for producing plans in an adequate time frame.

Conclusions and Future Work

We successfully integrated our planner TFD/M to provide the high-level control decisions for the PR2 in the *Tidyup-Robot* project. During development and deployment we identified several issues and advantages of this approach. One disadvantage in comparison to scripted systems is the runtime. Observed runtimes were in the tens of seconds up to some minutes. This is a result of the reasoning using modules calls, which at the same time is an advantage as it allows us to determine the best plan more precisely than by using symbolic approximations. There are however techniques to mitigate this issue. One way is to provide fast relaxed versions of modules for the search guidance heuristic and thus gain an informed heuristic quickly as done in TFD/M. Another way is to not produce a complete plan, but only an executable prefix (Kaelbling and Lozano-Perez 2011).

The major advantage of using a planner shows when multiple different actions are used by the system. A scripted system needs to take action dependencies into account explicitly, which can become arbitrarily complex. Consider the simple situation, when observing two objects that need to be brought into another room. If the door is closed, only one object can be taken as the robot requires a free hand. If it is open, both objects should be taken immediately. Those situations will easily be handled correctly by a plan-based approach. It is especially valuable as a developer does not need to imagine every possible situation to be solved, but only determine conditions and effects of each single action.

Our planner reasons about actions by searching the state space. A logical domain description defines if actions can be applied in a state during search and how they change this state. Semantic attachments might reason about geometric queries in searched states. Therefore geometric reasoners must support queries on arbitrary states independent from the current world state. An example query would be: Can the robot put down a cup, if it were grasped in the left hand and the robot were located in another room? This functionality is not necessarily available on all robotic systems, but forms an important requirement for integrating task and motion planning.

We believe semantic attachments to be crucial to generate good plans for robotic systems. However, in its current form the TFD module interface is difficult to use due to its generality. For instance, retrieving a predicate from the planners state is based on its name in the PDDL domain. This is a source of error, since the names cannot be verified at compile time. Thus, we introduced the Object-oriented Planning Language (OPL) that provides methods to automatically generate a domain-specific module interface (Hertle et al. 2012). Class instances are derived from the OPL description allowing type-safe access to the current planning state.

Beyond OPL, we investigate the automated generation of action executor plugins, similar to the module interface. The generated action executors will be able to update the planner state based on the OPL action preventing unintended or accidental state changes.

Our overall goal is to make a generic planning based execution architecture available, in particular to researchers of other subject areas.

Acknowledgments

This work was partially supported by DFG as part of the SFB/TR8 project R7 and in the PACMAN project within the HYBRIS research group (NE 623/13-1). We would like to thank all members of the *Tidyup-Robot* project.

References

Beetz, M.; Arbuckle, T.; Bennewitz, M.; Burgard, W.; Cremers, A.; Fox, D.; Grosskreutz, H.; Haehnel, D.; and Schulz, D. 2001. Integrated plan-based control of autonomous service robots in human environments. *IEEE Intelligent Systems* 16.

Cambon, S.; Alami, R.; and Gravot, F. 2009. A hybrid approach to intricate motion, manipulation and task planning. *International Journal of Robotics Research* 28(1):104–126.

Dornhege, C.; Eyerich, P.; Keller, T.; Trüg, S.; Brenner, M.; and Nebel, B. 2009. Semantic attachments for domain-independent planning systems. In *International Conference on Automated Planning and Scheduling (ICAPS)*, 114–121. AAAI Press.

Fox, M., and Long, D. 2003. An Extension to PDDL for Expressing Temporal Planning Domains. *Journal of Artificial Intelligence Research* 20:61–124.

Gregory, P.; Long, D.; Fox, M.; and Beck, J. C. 2012. Planning modulo theories: Extending the planning paradigm. In *International Conference on Automated Planning and Scheduling (ICAPS)*.

Hertle, A.; Dornhege, C.; Keller, T.; and Nebel, B. 2012. Planning with semantic attachments: An object-oriented view. In *European Conf. on Artificial Intelligence (ECAI)*.

Kaelbling, L., and Lozano-Perez, T. 2011. Hierarchical task and motion planning in the now. In *IEEE Conference on Robotics and Automation (ICRA)*.

Kresse, I., and Beetz, M. 2012. Movement-aware action control – integrating symbolic and control-theoretic action execution. In *IEEE Conference on Robotics and Automation (ICRA)*.