Generalized Plan Design For Autonomous Mobile Manipulation in Open Environments

(Extended Abstract)

Jan Winkler and Michael Beetz
Institute for Artificial Intelligence, Universität Bremen
Am Fallturm 1, 28359 Bremen, Germany
{winkler, beetz}@cs.uni-bremen.de

ABSTRACT

Autonomous robotic agents acting in open environments have to master situations and action effects they did not anticipate. To deal with these issues we propose an information processing concept for plan interpretation that is based on three concepts:

- statically defined, and dynamically inferred knowledge,
- context-definition on a task level to implicitly reparameterize all sub-actions during a task,
- and control structures for plans that can monitor plan execution for unexpected events and respond appropriately.

Implications of plans realized according to these design patterns are explained on the example of a reasoning intense autonomous manipulation task.

Categories and Subject Descriptors

I.2.9 [**Robotics**]: Autonomous vehicles; I.2.8 [**Problem Solving, Control Methods, and Search**]: Plan execution, formation, and generation; D.3.3 [**Language Constructs and Features**]: Concurrent programming structures

General Terms

Languages, Reliability, Algorithms

Keywords

Robot planning and plan execution; Robotic agent languages and middleware for robot systems; Reasoning in agent-based systems; Mobile agents

1. INTRODUCTION

Competent performance in complex everyday activities is still a challenging task for robotic agents, as their task details quickly change based on the current situation's context. To make a robot able to realize and use information about its current context, we propose plan control structures that transparently take into account the current situation to implicitly parameterize its subtasks and apply failure handling strategies. With knowledge about anticipated task outcomes, a robot gains the ability to transparently undo an action if it had undesired effects. Such abnormal behaviour is defined as unpredicted context change in dynamic environments, which leads to a change in the robot's current action strategy.

Appears in: *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2015), Bordini, Elkind, Weiss, Yolum (eds.), May 4–8, 2015, Istanbul, Turkey.*
Copyright © 2015, International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

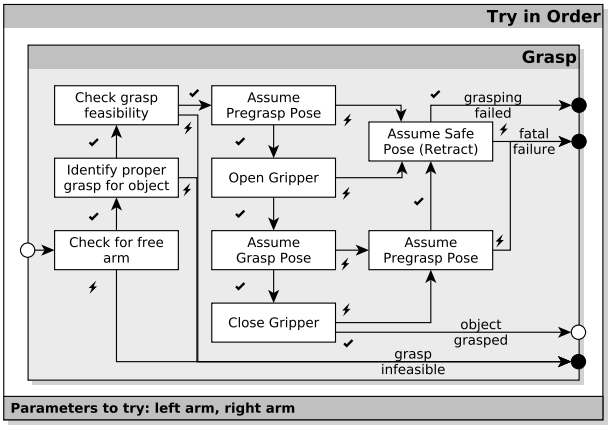

Figure 1: Reasoning, failure handling and recovery steps within a *Grasp* task for picking objects. The task either succeeds, signal infeasibility, fails cleanly, or fails fatally beyond recovery. Parameters are tried: left, right arm.

We briefly describe the importance of static and dynamic knowledge in autonomous systems. The former gives a robotic agent vaguely specified initial plan parameters, while the latter grounds these parameters in the concrete situation. We show their implications in the field of mobile manipulation by autonomous robot agents.

2. CONTEXTUAL KNOWLEDGE IN AUTONOMOUS ROBOT AGENTS

Commonly, robot plans such as in CRAM [1] are structured as hierarchical trees, allowing encapsulation and modular reuse. One aspect not yet part of robot plan languages is implicit context awareness. Tasks ought to perform differently based on their *explicit parameterization*, but also on *implicit circumstances*. These depend on dynamic factors, but also on static knowledge available through external knowledge bases. We therefore propose a language element to enhance a plan's context awareness without encoding all possible situations into the plans themselves: The `with-context` environment.

(`with-context` *(c₁ ... cₙ) code*)

Nested plans can define a contextual setting for other plan building blocks that they are using. Any layer can therefore either add new contextual parameters c_i, or alter old ones. The behaviour of

single plan blocks in `code` can thus be influenced by a semantically higher hierarchy. A context that describes the transport of liquid filled mugs results in different maximum tilting angles during motion planning than when transporting empty ones, given the assumption that nothing should be spilled.

2.1 Static and Dynamic Knowledge

A mobile robotic agent can find its way around an environment by using *design-time generated static knowledge*. Information sources are floor maps, the collision environments, or the tasks to perform. Using static knowledge as task context, hints can be asserted about the situation (*"The cups are in the lower drawer today"*, *"The oven is unusable, use the microwave"*), or about how a task should to be performed (*"Don't tilt the full coffee mug when bringing it"*).

Task constraints are either statically specified at design-time, or inferred dynamically. The former is for example which object to fetch, while the latter is the exact gripper pose to assume for grasping it w.r.t. the current robot pose and the exact object location. The task gets dynamic information from the context. Dynamic contexts together with static knowledge can constrain tasks in various ways. For example, *"The coffee mug is full"* and *"Do not tilt full mugs when grasping them"* leads to the more complex constraint of not tilting a mug in a concrete situation.

2.2 Implicit Modular Task Recovery

Ingham et al. [2] have developed the programming approach RMPL, that allows constraining of concurrently running processes while performing an otherwise sequential task. Their task execution monitors a set of variables and fires upon meeting predefined requirements. We extend this by introducing mechanisms for interrupting plan performance when unusual conditions arise, following a take-down routine. As each of our (sub)plans is accompanied by a take-down strategy to rewind its own effects, each layer in the hierarchy only has to undo its own actions. These failure handling strategies allow to encapsulate implicit recovery mechanisms to properly unwind tasks created by complex plan hierarchies.

Such "building block" plans have well-defined behaviours and capabilities, as well as required input parameters and possible outcomes. Nesting them entails advantages in code reusability, modularity and function encapsulation, but also grants semantic meaning to the contained structures.

3. MOBILE MANIPULATION

When performing manipulation tasks like grasping or transporting objects, such objects might start slipping from a robot's gripper or are taken away by other, possibly malicious agents. To allow an agent to for reactive handle such situations, it needs to be notified about changes in the course of action as early as possible. Monitoring the force exerted on the gripper's sensors can inform an agent about grasp irregularities, while perceptive aids can then give it a clue about the reason, such as an object slipping. To not constantly allocate a perception system for watching an object's position in the gripper, changes in the gripper force state can trigger a rudimentary check using the vision system to verify that the carried object is still in place [3]. This mechanism relies on dynamic knowledge inferred from situational information.

Figure 1 shows an example featuring the described concepts: (○) a modular task architecture, (○) static knowledge, such as available arms and the object's characteristics, and (○) dynamically inferred knowledge, such as free arms and situationally proper grasps.

4. RELATED WORK

Langley et al. [4] presented the ICARUS architecture for controlling cognitive agents in complex physical environments while performing pick and place tasks. Their MÆANDER component is executing plans generated by the DÆDALUS planner. While MÆANDER performs according to (concurrently) delivered plans by DÆDALUS, the latter is responsible for recovering from unexpected situations. While their system is designed upon similar principles with respect to reactivity, concurrency and versatility, our approach puts more emphasize on cascaded control loops for failure recovery before falling back to a higher abstraction layer in order to not replan globally, but first try to recover from the current problems locally.

Simmons et al. [5] have designed a robot architecture using the PRODIGY planning system to control an autonomous agent that performs office delivery tasks. They control the Xavier robot which mainly does navigation tasks in a structured, unprecisely modeled environment of office rooms. They explicitly account for failure handling: Low level components signal whether they reached their goal, while the high level planner verifies every action's outcome. Recovery is then planned by the high level system by adding subgoals to the current plan. We extend this principle idea by allowing lower level components to perform relatively simple recovery actions within their own capabilities, potentially saving planning systems from unnecessarily replanning more complex tasks.

5. CONCLUSION

In this paper, we described two sources of information to support autonomous robotic agents when performing tasks: Static knowledge defined during design-time, and dynamic knowledge derived from the situational context. We proposed a yet missing abstract language construct for robot plan languages to supply modular action plans with situational context. When tasks fail unexpectedly, we describe a task unwinding concept and detailed its use on the example of mobile manipulation.

Acknowledgements

This work was supported in part by the DFG Project BayCogRob within the DFG Priority Programme 1527 for Autonomous Learning and the EU FP7 Projects *RoboHow* (Grant Agreement Number 288533) and *SAPHARI* (Grant Agreement Number 287513).

REFERENCES

[1] M. Beetz, D. Jain, L. Mösenlechner, M. Tenorth, L. Kunze, N. Blodow, and D. Pangercic, "Cognition-enabled autonomous robot control for the realization of home chore task intelligence," *Proceedings of the IEEE*, vol. 100, no. 8, pp. 2454–2471, 2012.

[2] M. Ingham, R. Ragno, and B. C. Williams, "A reactive model-based programming language for robotic space explorers," in *International Symposium on Artificial Intelligence, Robotics, and Automation in Space (i-SAIRAS)*, Montreal, Canada, 2001.

[3] D. Kragic and M. Vincze, "Vision for robotics," *Found. Trends Robot*, vol. 1, no. 1, pp. 1–78, 2009.

[4] P. Langley, K. B. McKusick, J. A. Allen, W. F. Iba, and K. Thompson, "A design for the icarus architecture," *ACM SIGART Bulletin*, vol. 2, no. 4, pp. 104–109, 1991.

[5] R. Simmons, R. Goodwin, K. Haigh, S. Koenig, J. O'Sullivan, and M. Veloso, "Xavier: Experience with a layered robot architecture," *ACM magazine* Intelligence, 1997.