An ontology-based hybrid architecture for planning and robust execution in tabletop scenarios

Alessio Capitanelli and Fulvio Mastrogiovanni

University of Genoa, Italy
`alessio.capitanelli@dibris.unige.it`
`fulvio.mastrogiovanni@unige.it`

Abstract. The aim of this work is to develop a task representation and execution framework for dual-arm manipulators operating in tabletop scenarios. In particular, we want to enforce robot's autonomy, robustness to failures, and (in perspective) a natural human-robot interaction. To this purpose, the framework integrates (i) point cloud perception, (ii) ontology-based knowledge representation, (iii) high-level task planning, as well as (iv) task execution and monitoring.
Our main contribution is an *open source*, closed-loop hybrid architecture based on semantic knowledge and high-level reasoning to ground perception-based task representation, reasoning and execution. An ontology integrates perceptual cues with planning-relevant knowledge to automatically perform context assessment, infer when to act to modify the environment and to generate appropriate definitions of domains and problems to solve. An interface between the ontology and low-level motion planners allows for updating the representation at run-time, thus enforcing robustness *versus* unmodelled traits of the environment. The framework has been validated using a Baxter dual-arm manipulator operating in a tabletop scenario.

Keywords: Reasoning, Knowledge Representation, Software Architecture

1 Introduction

In order to enforce a robot's autonomy and flexibility, a closer coupling is needed between high-level task planning and low-level action execution. Such a coupling is expected to allow *reactive* behaviors to operate not only on the basis of run-time feedback, but also within a well-defined *context*. High-level task planning is scarcely robust with respect to uncertainty and unmodeled traits of the environment. However, it allows for goal-driven and context-based behavior, and for dealing with knowledge at a high level of abstraction. Furthermore, for long-term autonomous operations, a robot's architecture must identify its own goals (in the form of *norms* or *directives* to satisfy) and recognize whether they are met: the robot must check unsatisfied norms, address violated directives, plan for a course of action solving such inconsistences, update its knowledge and, in case of failures, refine its plan.

A number of hybrid reactive/deliberative architectures, with similar goals, have been proposed [1][2][3]. However, our approach is different in that: (i) an ontology determines the need to act and monitors action execution by comparing *normative* and *factual* knowledge; (ii) an interface lays inbetween task and motion planning: on the one hand, the task planner can ignore the problem's geometry to a good extent; on the other hand, the necessity to represent details in the knowledge base is reduced. Our framework combines two approaches in the literature:

– As presented in [3][4], it is possible to infer problem definitions through *norm violation* detection. Previous approaches use ontology inconsistency as a triggering factor, whereas we use *extra logic* rules, i.e., SWRL rules [7]. While inconsistency-based methods are theoretically elegant, they suffer from practical issues: an ontology is not usable while inconsistent, several re-reasoning steps are required and completeness can hardly be ensured. Our approach does not *lock* the ontology, provides a PDDL-friendly representation and does not require re-reasoning for problem generation, exploiting re-planning instead.
– The high- and low-level planners have been integrated using a planner-independent interface layer [5], i.e., *we plan what to do but not how we do it.* To this aim, we recur to a *Skolemization* process: the system *assumes* some of the predicates to hold true and re-plans whenever the low-level motion planner invalidates such assumptions. Previous approaches simply trigger replanning with an updated PDDL problem, whereas we adopt the more general approach to update the ontology and to allow the reasoner to decide whether a new problem must be generated.

2 Design and workflow of the system's architecture

To validate our ideas we designed a software architecture adopting the well-known ROS framework [9], which integrates a point-cloud perception system, a Description Logic backed ontology-based knowledge representation system, a PDDL planner – Fast Downward [6][12], and a low-level motion planner with collision avoidance, namely Moveit! [10]. We use the perception system discussed in [13], which is able to identify pure geometric shapes in a point cloud (i.e., planes, cylinders, cones and spheres) and estimate their geometric parameters using an optimisation procedure based on RANSAC [11]. In this Section, we illustrate how our framework works using a running example.

Scenario. Let us consider a dual-arm manipulator operating in a tabletop scenario, which mimics a *smart factory* industrial setting. The robot must satisy two operational requirements: (i) keeping the workspace tidy, e.g., assuring that a certain class of objects is always located on the left hand side of the table, and (ii) satisfying user requests, e.g., handling objects to human staff. In our

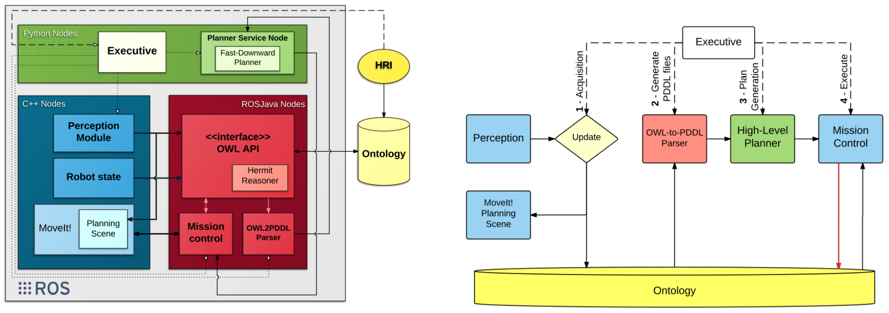

Fig. 1: Left: an overview of the proposed framework. Right: input data processing.

simplified scenario, we discriminate objects only by their shape and size. We define a *directive* as a maintenance goal, one that must be satisfied at all times. Furthermore, let us assume the following situation:

– The robot is in front of a table with spheres and cylinders only.
– The directive is: all cylinders must be on the left hand side of the table.
– A user asks the robot to pick up a sphere and to put it down inside a tray.
– The sphere cannot be picked up because it is cluttered by other objects.
– The goal position for the sphere is already occupied by another object.

To fulfill the requests, all the four phases in Figure 1 are involved.

Scene perception. The perceived scene is encoded in the ontology in terms of objects and their relationships to define predicates, e.g., `On(blue-ball,table)`. We distinguish two kinds of predicates: *descriptive* predicates represent factual knowledge about the environment, and *normative* predicates encode the desired state. Both objects and predicates are encoded as individuals belonging to the appropriate class in the ontology. We consider only unary and binary relationships and express their arguments as individual's object properties.

The *perception* module is always executed in the background to track object locations and keep the Moveit! planning scene consistent with the environment. Periodically, the last perceived scene is encoded in the ontology and updates the previous one. All predicates originating from the last encoded scene are considered *descriptive*.

Goal generation. Goals and directives are provided by the human interacting with the robot and are encoded as normative predicates. The system keeps updating its knowledge till the sets of *descriptive* and *normative* predicates do not mutually conflict. In our case, one directive is specified and it is satisfied till the human asks the robot to pick up a sphere and put it down in a tray. The conflicting predicates are detected by SWRL rules and the Pellet reasoner [8], which marks conflicting predicates for the generation of an appropriate PDDL problem file. Similarly, since action predicate definitions are represented in the ontology, corresponding relevant individuals are marked as well to generate a

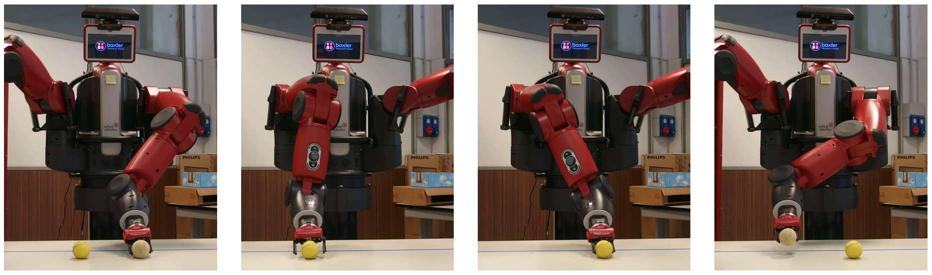

Fig. 2: The Baxter robot performing a simple swap task.

domain specification. Finally, a stored parser generates the problem and domain descriptions, and the system moves on to the planning phase.

Task planning and action execution. Execution is staged by a *mission control* module, which is organised as three submodules, namely *plan parsing*, *action execution* and *planner-independent ontology interface*. In the considered scenario, the robot generates a very simple plan, i.e., (i) freeing up the goal position by (ii) moving the occupying object to another free position and (iii) moving the sphere there. The plan instance is parsed and dispatched to Moveit! for execution. In the motion planning phase, Moveit! recognizes that one of the task would fail if executed, since the sphere is not reachable because of clutter. Hence, *mission control* stops the execution and the ontology is updated with a predicate specifying the cluttered situation. The system then moves back to the reasoning and goal generation phase, and a new plan is determined and executed, taking the clutter-related predicate into account. No further reasoning activity is required unless a new violation is detected. This may happen in case the plan would somehow fail. Once satisfied, temporary goals can be cleared from memory, while long-term directives can be kept to ensure robust operation.

3 Conclusions

In this paper, we propose a hybrid reactive/deliberative architecture for perception, knowledge representation and planning with a dual-arm manipulator. The framework has been evaluated in a scenario in which a Baxter robot (provided with an external RGB-D sensor) manipulates objects on a (possibly cluttered) table. Robot actions involve typical tabletop operations, such as pick and place, as well as the swap between the positions of any two objects. Preliminary results suggest that such an architecture can be a viable approach for robust and flexible autonomous operations in smart factory scenarios, especially in those contexts in which human-robot cooperation plays an important role. Current work is focused on expanding the architecture to allow a human user to interact with the robot using a natural language interface. The framework is partially available open source[1].

[1] Please check: https://github.com/EmaroLab.

References

1. Tenorth, M. and Beetz, M.: KnowRob –knowledge processing for autonomous personal robots. IEEE/RSJ International Conference on Intelligent Robots and Systems, 4261-4266 (2009)
2. Lemaignan, S., Ros, R., Msenlechner, L., Alami, R. and Beetz, M.: ORO, a knowledge management platform for cognitive architectures in robotics. In Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference, 3548-3553 (2010)
3. Galindo, C. and Saffiotti, A.: Inferring robot goals from violations of semantic knowledge. Robotics and Autonomous Systems, 61(10), 1131 - 1143 (2013)
4. Galindo, C. and Saffiotti, A.: Semantic Norms for Mobile Robots: When the end does not justify the means. IFAC Proceedings Volumes, 45(22), 84-89 (2012)
5. Srivastava, S., Fang, E., Riano, L., Chitnis, R., Russell, S.: Combined Task and Motion Planning Through an Extensible Planner-Independent Interface Layer. IEEE International Conference on Robotics and Automation (ICRA), 639 - 646 (2014)
6. Fox, M., Long, D.: PDDL2.1: an extension to PDDL for expressing temporal planning domains. JAIR, vol. 20, no. 1, 61124, (2003)
7. Horrocks, I., Patel-Schneider, P.F., Boley, H., Tabet, S., Grosof, B. and Dean, M.: SWRL: A semantic web rule language combining OWL and RuleML. W3C Member submission, 21, 79 (2004)
8. Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A. and Katz, Y.: Pellet: A practical owl-dl reasoner. Web Semantics: science, services and agents on the World Wide Web, 5(2), 51-53 (2007)
9. Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R. and Ng, A.Y.: ROS: an open-source Robot Operating System. In ICRA workshop on open source software, Vol. 3, No. 3.2, 5 (2009)
10. Sucan, I.A., Chitta, S.: MoveIt. http.://moveit.ros.org (2016)
11. Fischler, M.A., Bolles, R.C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM, 24(6), 381-395, (1981)
12. Helmert, M.: The Fast Downward Planning System. J. Artif. Intell. Res.(JAIR), 26, 191-246 (2006)
13. Buoncompagni, L., Mastrogiovanni, F.: A software architecture for object perception and semantic representation AIRO (2015)