Capabilities in Heterogeneous Multi Robot Systems

Jennifer Buehler
University of New South Wales, Sydney, Australia
jenniferb@cse.unsw.edu.au

1 Introduction

Groups of robots are often able to accomplish missions that no single robot can achieve by themselves. Robustness and flexibility are increased by the diversity of the robots, each contributing different capabilities.

In highly unpredictable domains such as search and rescue, accurate predictions of the outcomes of a robot's actions are virtually impossible. Approximate models and algorithms are required which help to estimate the outcome with highest possible confidence. Although many aspects of heterogeneous multi-robot systems have been widely studied, few researchers explicitly formalize robot *capabilities*. A model of capabilities can prove very useful for describing and reasoning about robots' diversity, task suitability and execution. This work presents a framework that formalizes a robot's capabilities, abstracting from underlying robot architectures and providing a means to estimate a robot's performance in a task.

2 Related work

One key element in multi-robot systems is to assign tasks to robots such that a meaningful division of work is achieved. For estimating "expected quality of task execution", *utility* is a widely used concept in multi-robot coordination. The goal of *task allocation* is to find robot-task assignments such that the overall utility is maximized [Gerkey and Matarić, 2004]. Many approaches to compute such a utility measure have been proposed, but only a few explicitly consider different notions of *robot capability*. Most of this work relates capabilities to some kind of *resource*, e.g. sensors/actuators, processing capacities [He and Ioerger, 2003], [Chen and Sun, 2010], and/or software modules [Parker and Tang, 2006]. A capability can also be a simple *subtask*, for which each robot learns their suitability [Fua and Ge, 2005]. Such concepts are ultimately used in different ways to determine a robot's utility for a task. Other research also formalizes capabilities relating to robot components to infer what a robot can do [Kunze *et al.*, 2011] or how to decompose a task into simple 'skills' [Huckaby and Christensen, 2012].

Previous research has taken into account a robot's **intrinsic** capabilities[1] for estimating such utility values [Fua and Ge, 2005]. Simple **extrinsic** factors have also been considered, such as metric distance to the task or resource requirements [Chen and Sun, 2010]. However, such considerations were mainly tailored to the specific experiments. To the best of our knowledge, no systematic approach to incorporate task details in a general way for all possible tasks has been proposed to date. We propose a framework which can generate task solutions using robot capabilities and helps *estimating* task execution qualities considering task-specific details.

[1] *Intrinsic* capabilities express what a robot can generally do (e.g. lift a rock); *extrinsic* factors specify task details (i.e., weight of rock).

3 Our approach

Diversity does not only emerge from differences in hardware, but also from distinct robot software architectures and algorithms. Therefore, a robot's physical description as for example specified in URDF[2] is not enough to infer a robot's capabilities. We define capability in a way abstracting from hard- *and* software, such that a robot can learn constraints and quality relating to its capabilities.

A *capability* is a simple functional element which can be part of many different tasks. Our definition is supported by Zuech and Miller [Zuech and Miller, 1989] p. 163:

> "There are a limited number of task types and task decompositions [...]. There are only a few different types of **reach, grasp, lift, transport, position, insert, twist, push, pull, release**, etc. A list of *parameters* with each macro can specify *where* to reach, *when* to grasp, *how far* to twist, *how hard* to push, etc."

With this definition, a capability *abstracts* from underlying architectures at a medium level of granularity. For example, it is not important *how* a robot grasps an object (e.g., which finger movements), but only *what* it can *probably* grasp. We further add **computational capabilities** such as Localisation, Path Planning and Object Recognition.

We define OWL[3] ontologies grouping different **devices** and **algorithms** according to their function and purpose. We link **capabilities** to these functional elements in another OWL ontology. Additionally, capabilities have assigned *pre- and postconditions* and defined *in- and output datatypes* (e.g. images, maps, object id's, poses...). For example, a postcondition of REACH is that the manipulator is `in-pose`, which is

[2] see `http://www.ros.org/wiki/urdf`
[3] see `http://www.w3.org/TR/owl2-overview/`

also the precondition for GRASP. Postcondition of GRASP is that the object is `held`, which also is the precondition for LIFT, while its postcondition is that the object is `not grounded`. Expressing this definition of capabilities in the common planning language PDDL[4] we can **generate task solutions** using a general (classical) planning approach. In the above example, *picking up an object* is specified by the goal that the object is `not grounded`. One solution is to connect REACH, GRASP and LIFT. Connecting capabilities in such a way also describes **dependencies** among them. For example, for acquiring the `pose` of an object, the robot needs to see it using VISION (other dependencies are implied but won't be detailed here). In the above example, VISION and other capabilities will be required to generate the correct dataflow, i.e. provide the object's `pose` to the REACH capability. One advantage of such capability dependencies is that if any of the required capabilities is not functioning or reduced (e.g., if the image quality is reduced under current conditions), it can be inferred that the quality of depending capabilities (i.e. grasp) will be reduced as well.

We define all possible conditions and datatypes in respective ontologies and generate solutions using a classical STRIPS-like planner. Such a generated task solution however only describes a possible solution and does not yet consider task-specific (extrinsic) details. This is done in an additional step which evaluates a robot's performance for a specific task solution. As mentioned before, we consider task-specific details in an *approximate* way in order to compare different robot's likely performance. For this, we introduce capability **parameters**: the area that a robot can REACH can be represented by a spherical shape around the manipulator; terrain on which a robot can MOVE may be described by indices of "terrain roughness" with assigned average speeds; sizes it can GRASP may be approximated with any bounding volume; LIFTING will be assigned weight ranges a robot can lift, and so on. While such parameters by no means allow for accurate predictions, they still provide a much better estimate than simply assuming the robot has or does not have the capability. Such approximate information can also be useful to improve heuristics (e.g. in planners or task allocation algorithms) and to share knowledge among team-mates. Another advantage of this representation is that it also accounts for cases in which only approximate information is available (e.g. only approximate object shape known anyway). Verifying that a capability meets the requirements of the task involves matching a parameter **specification** (e.g. the object size a robot can grasp) to a task-specific **instantiation** (i.e., the actual size of the object. The matching is done by checking if one shape fits into the other). Closer examination revealed that many capabilities can share the same type of parameter spaces (e.g. shape-within-shape, point-in-shape).

Finally, after a robot was found to meet the task-specific details, an overall utility for the task can be obtained by summing up the *qualities* the robot has learned for the single capabilities involved in the task. We use *execution time estimates* and *success probabilities* as a measure of quality.

Our work also includes a **learning framework** for a robot's capability parameters, e.g. for learning possible areas to reach with the manipulator. The learing is done using a robot simulator, and finally verified on on the real robots.

Furthermore, we are looking into inferring possible capabilities from a robot's URDF description. If required hardware dependencies are not met, some capabilities can be ruled out right away. Other directly or indirectly depending capabilities will also be affected by this. This leaves a set of candidate capabilities which can be ruled out or further configured by the user. This eases the integration of new architectures.

We keep a database of learned capability parameter and quality values for all robots. Hence the learning for a new robot can be initialized by values which other robots with similar or equal hard- and software have learned before.

[4]Planning Domain Definition Language

4 Evaluation and Contribution

We demonstrate task solution generation with a classical planning approach. In a next step we will compare predicted and actual task execution in simulation and on real robots. We also aim to show how potential capabilities can be inferred from a URDF description and how other robot's experience can be used to initialize learning for others.

Our proposed framework makes a contribution with the task solution generation using capabilities independent of underlying architectures and domain. Task execution estimates considering task-specific details are included. The framework also considers dependencies to hard- and software configurations and includes a learning algorithm for robot capabilities.

References

[Chen and Sun, 2010] Jian Chen and Dong Sun. An online coalition based approach to solving resource constrained multirobot task allocation problem. In *2010 IEEE International Conference on Robotics and Biomimetics*, 2010.

[Fua and Ge, 2005] Cheng-Heng Fua and S.S. Ge. COBOS: cooperative backoff adaptive scheme for multirobot task allocation. *IEEE Transactions on Robotics*, 21(6), 2005.

[Gerkey and Matarić, 2004] Brian P. Gerkey and Maja J Matarić. A formal analysis and taxonomy of task allocation in multi-robot systems. *The Intl. J. of Robotics Research*, 23(9), 2004.

[He and Ioerger, 2003] Linli He and Thomas R Ioerger. A quantitative model of capabilities in Multi-Agent systems. *Proceedings of IC-AI'2003*, 2003.

[Huckaby and Christensen, 2012] J. Huckaby and H. Christensen. A taxonomic framework for task modeling and knowledge transfer in manufacturing robotics. In *Workshops at 26th AAAI Conference on Artificial Intelligence*, July 2012.

[Kunze *et al.*, 2011] L. Kunze, T. Roehm, and M. Beetz. Towards semantic robot description languages. In *2011 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2011.

[Parker and Tang, 2006] L.E. Parker and Fang Tang. Building multirobot coalitions through automated task solution synthesis. *Proceedings of the IEEE*, 94, 2006.

[Zuech and Miller, 1989] Nello Zuech and Richard K. Miller. *Machine Vision*. Springer, 1989.