

(AI) Planning to Reconfigure your Robot?

Mark Judge

Department of Automatic Control and Systems Engineering, University of Sheffield, Sheffield
m.judge@sheffield.ac.uk

Abstract: Current and future robotics and autonomous system applications will be used in highly complex environments. In such situations, automatic reconfiguration, due either to changing requirements or equipment failure, is highly desirable. By using a model of autonomy based around the Robot Operating System (ROS), and building on previous work, it is possible to use Artificial Intelligence (AI) Planning to facilitate the automatic reconfiguration. In this way, standard AI Planning machinery may be used with a suitable mathematical model of a given system. This paper reviews the background to this concept and details the initial steps taken towards the combination of AI Planning technology and a physical system, with the aim being to have the planner provide possible validated system reconfigurations which can then be implemented on the hardware.

1 Introduction

Early robotics systems [7] were built around three fundamental primitives: Sense, Plan, Act (SPA). However, since it became clear that planning was too complex to be carried out on-board in real time, AI planning has become a research area in its own right.

As a consequence of the divergence of planning and robotics, AI planning has grown, matured, and now employs a wide range of techniques, many of which are applied to a broad spectrum of complex problem domains [2].

Hence, originally, planning was intended as a mechanism for controlling the operation of a robot, but may now often be considered a general solving method, applicable to problems with a known current (*initial*) state and a desired *goal* state.

It is in this general problem solving sense in which AI planning technology is used here in order to provide a solution “plan” for the reconfiguration “problem” of an autonomous (robot) system. To facilitate the description of the autonomous system(s) as a planning domain, the Robot Operating System (ROS) [8] was used, with a mathematical model developed to describe the ROS system, with this in turn allowing autonomous reconfiguration.

A more detailed discussion of autonomous systems control and the formal model used for reconfiguration is provided in our foundational paper [1]. Here, in this current paper, we summarise that previous work, focusing more on the planning aspect and emphasising the use of an AI Planning logic system for the reconfiguration process.

2 Background

In this section, we highlight the need for the reconfiguration of autonomous and robotic systems, introduce ROS, and briefly describe one form of AI planning.

Autonomous control and robotics systems operate in highly complex environments, much more complex than the environments in which traditional control systems operated. The reconfiguration of such systems is required in order to accommodate either changes in the environment or

changes in a given system’s hardware [5], perhaps due to a fault or a better subsystem component becoming available. Such reconfiguration demands considerable resources from system engineers. Hence, automatic reconfiguration is desirable not only to minimise resource use, but also to reduce or remove errors and speed up redevelopment and redeployment.

ROS is an open source robot operating system, originally developed for use on specific large-scale service robot and mobile manipulator platforms. The designers’ stated aims in developing ROS were that their system would be peer-to-peer, free, open source, thin, tools based, and multi-lingual (C++, Python, LISP, Octave).

A typical ROS system comprises a number of processes, optionally distributed over multiple hardware systems, and consists of *nodes*, *messages*, *topics*, and *services*. Nodes (software modules) carry out the computation and communicate via messages, doing this by publishing to a particular topic. In addition to this “broadcast” type model, services are used for synchronous transactions, which are defined by a named string and pairs of messages of a strict type.

Classical planning, also known as STRIPS planning [3], requires that the *state space* be finite and fully observable. It is assumed that only specified actions can change a state and that they do so instantaneously, with the resulting state being predictable. A STRIPS planning problem, $P = (O, I, G)$, where O is a set of operators, I is a conjunction of fact literals describing the initial state, and G another conjunction of facts describing a partially specified goal state.

AI planning problems can be described using the Planning Domain Definition Language (PDDL) [6]. In order to find solutions to such planning problems, a dedicated planning system must be constructed. This is generally a time consuming process. Hence, for this work, one of the most well known, and best performing, planning systems, called Fast Forward (FF) [4], was used to solve the problem instances once these were formulated.

3 AI Planning for System Reconfiguration

One way of mathematically describing [1] a ROS system is by making use of a tri-partite graph (Figure 1).

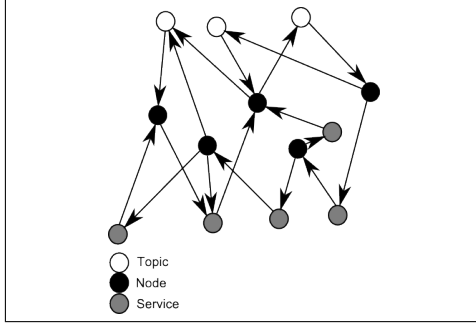


Figure 1: Example ROS graph.

In Figure 1, the three vertex types are considered different, with all inter-node communication occurring via a service or topic. Hence, the edges represent data flow, with a given topic or service requiring a minimum of one inbound edge from a ROS node.

To use AI planning machinery to reconfigure a ROS system of the type shown in Figure 1, we model the system using a PDDL domain description. This contains the action schema, which will be instantiated with the data contained in an associated problem instance description file.

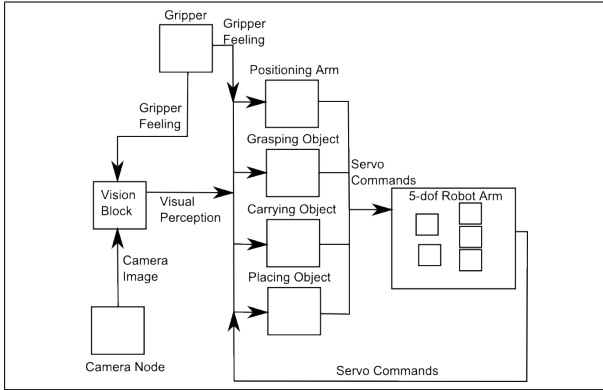


Figure 2: Example robot arm system diagram.

Taking as an example a modular robotic arm system combined with visual perception equipment, it is possible to set as a goal certain (lifting and moving) tasks, with the planning system providing the actual, physical (re)configuration detail to guarantee that a given task can be completed.

Considering Figure 2, the prototype planning system was set up to model two robot arms, each with 5 degrees of freedom (DOF), one with DC motor control, the other with servo control capability. The system is required to perform two different tasks. The first is Disk Loading, the second, Object Repositioning. The first task can only be performed with the servo control system in place, whilst the second can be carried out under either configuration.

In the planning action schema, each modelled ROS node

supplies (or requires) certain services required (or supplied) by other nodes. This aligns well with the notion of preconditions and effects [3] in AI planning.

```
(:action dc_place_object
:parameters
  (?dcservo1 - dcservo
   ?grippermodel - grippernode
   ?visualperceptnode1 - visualperceptnode
   ?positionarmnode1 - positionarmnode
   ?dcplacingobjectnode1 - dcplacingobjectnode)
:precondition
  (and (dcservo_available ?dcservo1 ?grippermodel)
        (gripper_sensing_available ?grippermodel)
        (visual_percept_available ?visualperceptnode1)
        ?visualperceptnode1 ?positionarmnode1)
        (no_dc_placing_object_available ?dcplacingobjectnode1)
        (visual_percept_not_assigned ?visualperceptnode1))
:effect
  (and (dc_placing_object_available ?dcplacingobjectnode1)
        (not (visual_percept_not_assigned ?visualperceptnode1))
        (not (no_dc_placing_object_available ?dcplacingobjectnode1))))
```

Figure 3: Example prototype action.

Part of the action description for the prototype is shown above in Figure 3. This shows the idea of one node (action) relying on the services provided by another, in the sense that the preconditions of the current action are satisfied by the effects of other, previous actions. Thus, a *plan* is a (re)configuration of the system.

Manual analysis of the ROS system used in the example gives two possible configurations for the Disk Loading task, and four for the Object Repositioning task. Running the planner on the PDDL encoded model gives the correct sets of solutions for both. A sample is shown in Figure 4.

```
0: DIGITAL_SERVO_BASE_TO_SHOULDER DIGITALBASENODE1 DIGITALSHOULDERNODE1 DISCLOAD1
1: DIGITAL_SERVO_SHOULDER_TO_ELBOW DIGITALBASENODE1 DIGITALSHOULDERNODE1 DIGITALELBOWNODE1
2: DIGITAL_SERVO_ELBOW_TO_WRIST DIGITALSHOULDERNODE1 DIGITALELBOWNODE1 DIGITALWRISTNODE1
3: DIGITAL_SERVO_WRIST_TO_GRIPPER DIGITALELBOWNODE1 DIGITALWRISTNODE1 DIGITALGRIPPERNODE1
4: SETUP_CAMERA_IMAGE CAMERANODE2
5: DIGITAL_SERVO_AVAILABLE DIGITALWRISTNODE1 DIGITALGRIPPERNODE1 DIGITALSERVO1
6: SETUP_GRIPPER_SENSING GRIPPERNODE2
7: VISUAL_PERCEPTION_READY VISUALPERCEPTNODE2 GRIPPERNODE2 CAMERANODE2 POSITIONARMNODE2
8: DIGITAL_DISC_LOADING DIGITALSERVO1 GRIPPERNODE2 VISUALPERCEPTNODE2 POSITIONARMNODE2 DIGITALDISCLOADINGNODE1
```

Figure 4: Example Disk Loading configuration.

4 Conclusion

This paper summarises¹ a foundation for the development of autonomously reconfigurable robot systems. Results show, by defining a system in ROS and modelling this in PDDL, it is possible to attain automatic reconfiguration.

References

- [1] J.M. Aitken, S.M. Veres, and M. Judge. Adaptation of System Configuration under ROS. In *Procs of 19th IFAC World Congress.*, 2014.
- [2] A. Coles et al. A survey of the 7th ipc. *AI Magazine*, 33(1):1–8, 2012.
- [3] R. E. Fikes and N. J. Nilsson. Strips. In *Readings in Planning*, pages 88–97. Kaufmann, San Mateo, CA, 1990.
- [4] J. Hoffmann and B. Nebel. The FF planning system. *Journal of Artificial Intelligence Research*, 14:253–302, 2001.
- [5] S. Karapinar, D Altan, and S. Talay. A robust planning framework for cognitive robots. In *Proceedings of AAAI Workshops (online)*, 2012.
- [6] Craig Knoblock. Pddl. *AIPS98*, 78(4):1–27, 1998.
- [7] N. J. Nilsson. *Principles of artificial intelligence*. Morgan Kaufmann, San Francisco, CA, USA, 1980.
- [8] M. Quigley et al. Ros: an open-source robot operating system. In *ICRA Workshop on Open Source Software*, 2009.

¹Thanks to J. Aitken and S. Veres, for access to their parts of [1] from which this paper draws material.