Enabling a Resource Limited Robot to Formulate Complex Plans

Demetrius Taylor
Lamar University

Abstract

In order to perform complex tasks autonomously a robot must be able to formulate plans. Robots used for educational purposes often have limited memory and processor speed, insufficient to run many of today's planning systems. The INTERRAP architecture for autonomous agents can be used for controlling autonomous robots. INTERRAP is a layered architecture. One of the layers is responsible for formulating plans for a robot to accomplish its goal. An advantage of the layered architecture is that one layer can be changed without affecting the others. This paper describes how to use an existing planning system in conjunction with the INTERRAP architecture to enable an autonomous robot with limited resources to formulate complex plans. A design is formulated that allows a planning system running on a PC with adequate resources to formulate plans for a Khepera III robot running the INTERRAP architecture. The benefit of this approach is that it enables the resource limited robot to expand its planning capabilities by using an existing powerful planner. It also serves as a model of how to interface a sophisticated planner with the INTERRAP architecture's local planning layer. An example application for a forklift robot is included to illustrate the approach.

1. Problem and Motivation

Jorg Muller's INTERRAP architecture (Muller 1996) is a layered control architecture that enables robots to behave autonomously. It has been identified as an example of a design pattern for multi-agent systems. (Lind 2003). One of the layers is responsible for formulating plans for a robot to accomplish its goal. An advantage of the layered architecture is that one layer can be changed without affecting the others. This paper describes a method for interfacing the INTERRAP architecture with an existing powerful planning system to expand the planning capabilities of a resource limited robot.

2. Background and Related Work

2.1 The robot

The Khepera III robot is manufactured by the Swiss company K-Team, and is used extensively for education and research (K-team 2012). The Khepera III has the KoreBot II as the motherboard, pre-installed with a Linux OS. The KoreBot II has a 600MHz Intel XSCALE PXA-270 processor with 128 Megabytes of RAM and 32 Megabytes of flash memory for storage. It has 8 infrared light sensors around the side, 2 infrared sensors on the bottom of the robot that can be used as ground proximity sensors or for following lines, and 5 ultrasonic sensors. The Khepera III is equipped with a gripper from K-team for the KoreBot II motherboard. Figure 1 shows the Khepera III with a top mounted gripper.

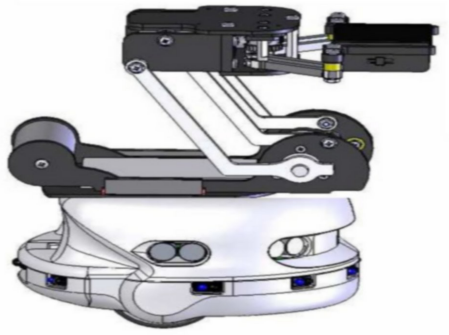

Figure 1. The Khepera III robot with its gripper (K-team 2012)

2.2 The control architecture

The INTERRAP architecture consists of a World Interface and three layers: the behavior-based layer (BBL), the local-planning layer (LPL), and the cooperative-planning layer (CPL) (Muller 1996).

The World Interface lies between the BBL and the robot's environment. It includes sensors that take information from the environment and send it to the BBL; actuators that take information from the BBL and control wheels, grippers, and other robot components; and a communications module that allows robots to send and receive messages to and from each other.

The BBL is the layer that deals with "reactivity and procedural knowledge" on the robot. The BBL holds the robot's innate knowledge in the form of Patterns of Behavior (POBs) that specify what

behavior or action the robot should take given input from the environment.

The LPL is the layer that formulates plans for the agent to achieve its goals. Muller's INTERRAP design recommends the use of a Hierarchical Task Network (HTN) style planner. The HTN style planner takes a high-level description of the task to be performed and breaks it down into a set of lower-level actions that can be executed by the BBL's POBs. The action decomposition process continues at each level until only primitive actions remain.

The CPL is the layer that allows the robot to work with other autonomous agents to achieve common goals.

2.3 The planning system

Following a survey of available existing planners, SHOP2 is selected as the planning system for the INTERRAP architecture's LPL. The Simple Hierarchical Ordered Planner (SHOP2) was developed in 2001 at the University of Maryland (UMD 2012). It is based on the 1999 version of SHOP, which is an ordered task decomposition planner. An ordered task decomposition planner outputs plan tasks in the order in which they will be executed. (Nau et. AL. 2001, 2013) SHOP2 is written in Common Lisp, and can run on any computer that has a Common Lisp compiler. The authors of SHOP2 have run the code on "x86 and x86_64 Linux and Mac OSX." SHOP2 uses its own formalism that uses Lisp expressions to represent the domain. SHOP2 is chosen for several reasons:

1. It is free. Because SHOP2 is licensed under the GNU General Public License (GPL), it is easily accessible, the source can be easily modified, and it can be redistributed easily as long as it is still under the GPL license.

2. It is relatively new and is still being actively maintained.

3. It is a HTN style planner, which is compatible with the LPL design presented by Muller (Muller 1996).

4. It won one of the top 4 awards at 2002 International Planning Competition for solved problems in every planning domain at the competition (UMD 2012).

3. Approach and Uniqueness

The approach taken in this work is to interface the INTERRAP controller with the powerful SHOP2 planner. The robot controller and the planner communicate with each other via a language for task oriented robots. The Local Planning Layer of the robot sends a request for a plan to SHOP2 when needed, and the SHOP2 planner formulates and sends the plan back to the robot's Local Planning Layer for execution. A search of the literature finds that this has not been done before.

3.1 Design

Figure 2 shows the high-level design of the robot system. The INTERRAP controller runs on the Khepera robot. Because the Khepera does not have enough resources to run both INTERRAP and SHOP2, SHOP2 runs on a Linux workstation. The robot communicates with the workstation via TCP/IP on a wireless ad-hoc network. When the robot needs a plan it sends a message with its domain information to the workstation and asks the workstation to find a plan. When the workstation receives the request it converts the message into the SHOP2 language and passes the information to SHOP2. When a plan is found the workstation then parses the plan into steps and sends it back to the Khepera robot for execution.

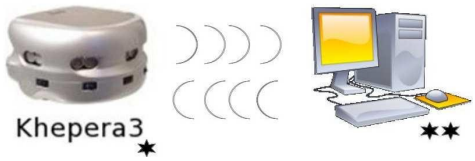

Figure 2 illustration of the wireless communication between the Khepera robot and Linux workstation
**** image from the k-team website;***
***** image from wikipedia***

Communication Language for Autonomous Mobile Robots (CLAMR) (Doerschuk et AL 2006) is used in this project as the language in the communications modules on the robot and the workstation. This language for task oriented robots includes several types of simple messages, called speech acts, including requests, informs, acknowledgements, promises, etc. The description of the speech act types (SATs) is shown in Table 1. Each message refers to a type of action, such as go forward, pick up, turn, plan, etc. Some actions may include parameters, such as direction, duration, etc. For example, a message can be a request for a robot to turn right. CLAMR also provides a means of representing complex messages, including sequences of actions and conditional actions

Table 1. Speech Act Types in CLAMR

Speech Act Type (SAT)	Description
INFORM	Message contains information about some-thing
REQUEST	Message contains a request for some action to be performed
QUERY	Message is asking for information
ACK-LEDGE	Message is acknowledging another SAT
OFFER	Message is offering to perform an action
REQUEST-QUEST-WHEN	Request to perform a task when a signal is given

Figures 3 and 4 illustrate the dynamic behavior of the INTERRAP/SHOP2 system. The project starts execution with the initialization of the INTERRAP architecture and the workstation server code. Then both the INTERRAP architecture and the workstation try to establish a connection to one another. Once a connection is established, the INTERRAP architecture's BBL and LPL go into an infinite loop waiting for messages. Eventually the BBL receives a request to perform a high-level task for which it has no POB. It sends the request up to the LPL for a plan. The LPL checks its plan library to see if it has a plan. If it has a plan it translates the plan into low-level POB's and sends them to the BBL for execution. This sequence is shown in Figure 3.

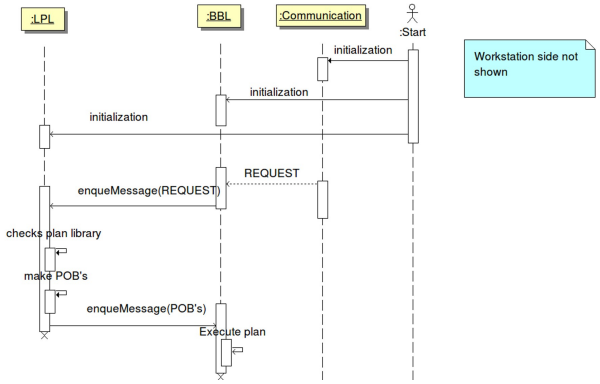

Figure 3. Sequence diagram for a plan in the LPL library

If the LPL does not have the plan in its library, it formulates a message requesting a plan using the information from the BBL and sends the new message to the BBL to send to the workstation via the communications modules on the robot and the workstation. After receiving the message, the workstation translates that message into a SHOP2 problem definition. The workstation then writes that definition to a file and forks a new instance of SHOP2 with a pointer to the file. SHOP2 then formulates a plan and sends it back to the server via standard out and exits. The server reads the SHOP2 standard out and parses it into plan steps. Those plan steps are then made into a message informing the robot of the plan, and this message is sent back to the robot via the communications modules. The BBL, checking for messages from the server, receives the message, and sends it back to the LPL. The LPL updates its plan library, then translates those plan steps into low-level POB's and sends them back to the BBL to execute. This sequence is shown in Figure 4.

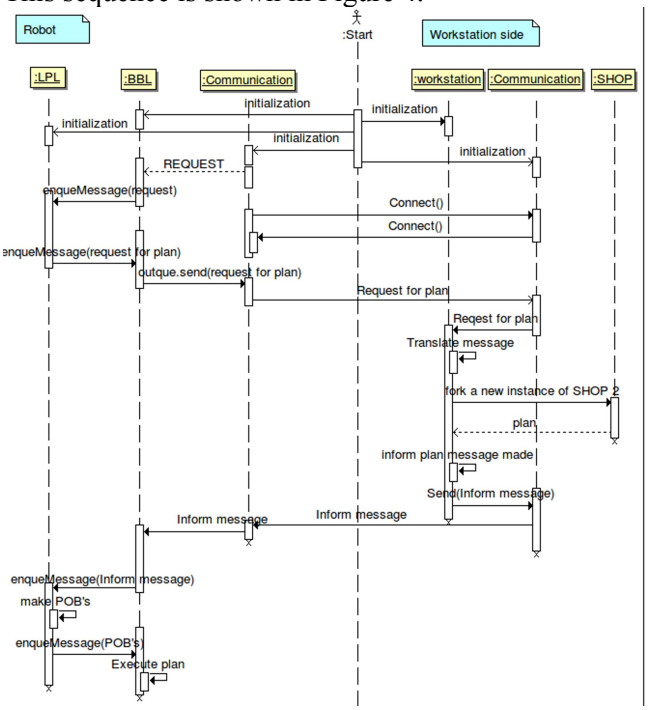

Figure 4. Sequence diagram for a plan not in the LPL Library.

3.2 Example

To illustrate the working of the system, an application of a planning problem in the forklift domain is presented here. The domain is based on a problem presented in (Muller 1996). Muller describes a rectangular area in which there are shelves and a loading ramp. When a truck arrives at the loading ramp with boxes it is the forklift robot's job to unload the truck and place its cargo on the shelves. Sometimes the robot will have to load the truck, in which case it will be the robot's job to retrieve a box from a shelf and place it on the truck for shipment. The warehouse is built in an indoor environment with hard floors.

The warehouse contains 4 shelves, each capable of holding two pieces of cargo, and a loading ramp capable of holding 4 pieces of cargo. Figure 5 shows a map of the warehouse environment. Each place that

the robot can pick up and place a piece of cargo is marked by a node. Each place that the robot changes direction is also marked by a node. Tracks connect each node to its nearest neighbor that does not go through a shelf or the ramp. The track is marked by a black line on the ground. The robot moves along the track from node to node. In Figure 5, the nodes are the numbered circles, the shelves are the blue rectangles, and the ramp is the red rectangle.

Five POBs are used for the robot to complete its job: FOLLOW_LINE, PICKUP, PUTDOWN, TURN_LEFT, and TURN_RIGHT. The first, FOLLOW_LINE, makes the robot follow the line from its starting node to an adjacent destination node. PICKUP and PUTDOWN make the robot pick up a box or put down the box using the gripper. The last two POBs make the robot either turn left or right by 90 degrees; they are executed before the other POBs to make the robot face the correct direction before moving to the destination node, picking up or putting down a piece of cargo.

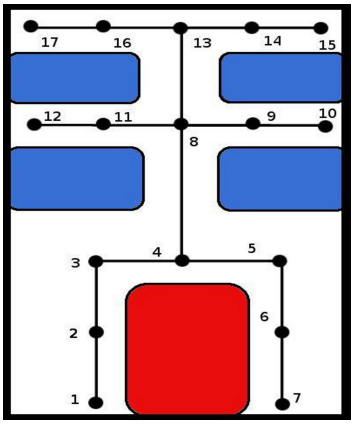

Figure 5. A map of the warehouse.

The LPL has a map of the warehouse and plans for moving from one node to another. However the LPL does not have a plan for loading or unloading a piece of cargo. SHOP2 must formulate a plan for load/unload operations and produce a set of tasks that the robot can do. These tasks are in the form of pickup, put down, and move.

The information required for SHOP2 to find a loading or unloading plan are the robot's location, what the robot needs to do, the cargo involved, and the cargo's location. This information will be used by the workstation to generate a SHOP2 problem definition.

A SHOP2 domain definition consists of a set of methods, operators, and axioms (Nau et. AL. 2003). Methods are used to decompose a compound task into a set of sub-tasks. They consist of a head, which is the name of the task and its parameters, preconditions, which is a list of conditions that the current state must hold in order to use the Method, and a list of operators that needs to be used in order to decompose the task. Operators describe what primitive tasks can be performed. Each operator consists of a head, which is the name of the operator, and its parameters. Then come the operator's preconditions, which must be true before the operator can be used; and finally delete lists and add lists. Delete lists and add lists are what happens to the world after the operator has been performed. Delete lists are a list of things that are no longer true about the world any more, and add lists are the things that are true now that the operator task has been performed.

Figure 6 shows the SHOP2 forklift domain definition. It consists of three operators: move, pickup, and putDown; and three methods: unload, load, and goto. The operators are the robot's POBs, while the methods are the robot's tasks.

```
(defdomain forklift (
   (:operator (!move ?robot ?from ?to)
      ((at ?robot ?from))
      ((at ?robot ?from))
      ((at ?robot ?to)) )
   (:operator (!pickup ?robot ?block)
      (and (holding ?robot nothing)(blockloc ?block ?shelf)
                 (at ?robot ?shelf))
      ((holding ?robot nothing)(blockloc ?block ?shelf))
      ((holding ?robot ?block)(free ?shelf)) )
   (:operator (!putDown ?robot ?block ?shelf)
      (and (holding ?robot ?block)(free ?shelf)
                 (at ?robot ?shelf))
      ((holding ?robot ?block)(free ?shelf))
      ((holding ?robot nothing)(blockloc ?block ?shelf)))
   (:method(unload ?robot ?block ?shelf)
      ((blockloc ?block ?n)(at ?robot ?n))
      ((!pickup ?robot ?block)
       (goto ?robot ?shelf)
       (!drop ?robot ?block ?shelf) )
      (and (blockloc ?block ?n)(not (at ?robot ?n)))
      ((goto ?robot ?n)
       (!pickup ?robot ?block)
       (goto ?robot ?shelf)
       (!drop ?robot ?block ?shelf)
      ) )
   (:method(load ?robot ?block ?ramp)
      (and (blockloc ?block ?n)(at ?robot ?n))
      ((!pickup ?robot ?block)
       (!move ?robot ?n ?ramp)
       (!drop ?robot ?block ?ramp)
      )
      (and (blockloc ?block ?n)(not (at ?robot ?n)))
      ((goto ?robot ?n)
       (!pickup ?robot ?block)
       (!move ?robot ?n ?ramp)
       (!drop ?robot ?block ?ramp)
      ) )
   (:method(goto ?robot ?destination)
      ((at ?robot ?loc))
      ((!move ?robot ?loc ?destination))
) ))
```

Figure 6. SHOP2 Forklift domain definition.

In this example, the robot needs a plan to unload block 1 from node 6 and place that block on shelf 11. A CLAMR message is sent to the workstation with SAT REQUEST, action PLAN, and parameters indicating the forklift domain and the goal for robot 1 to unload bock 1 from its current position and place it at node 11. The workstation translates the message into the problem definition displayed in Figure 7.

The problem definition consists of the name of the problem, the name of domain, the list of relevant facts that are true about the state of the environment, and the plan that needs to be formulated. The plan definition in Figure 7 says that node 11 is free, block1 is located at node6, robot1 is holding nothing, and

robot1 is at node1. The desired plan is for robot1 to unload block1 and put it at node11.

```
(defproblem Example forklift (
      (free node11)
      (blockloc block1 node6)
      (holding robot1 nothing)
      (at robot1 node1))
((unload robot1 block1 node11)) )
```

Figure 7. An example problem definition for the forklift domain

The plan that SHOP2 generates for the problem defined in Figure 7 is shown in Figure 8. The plan generated by SHOP2 is in Lisp format. There is only one plan, and that plan has 4 plan tasks. The tasks are:

1. Move robot1 from node1 to node6
2. Make robot1 pickup block1
3. Move robot1 from node6 to node11
4. Make robot1 put down block1 at node11

```
(((!MOVE ROBOT1 NODE1 NODE6)(!PICKUP ROBOT1
BLOCK1)(!MOVE ROBOT1 NODE6
NODE11)(!PUTDOWN ROBOT1 BLOCK1 NODE11)))
```

Figure 8. The plan generated by SHOP 2 for the problem defined in Figure 7.

The workstation sends a CLAMR message with SAT INFORM and action PLAN, along with parameters containing the number of steps in the plan and the plan steps formulated by SHOP2 to the robot. The robot's communications module transmits the message to the BBL, which in turns sends it to the LPL. The LPL translates the PLAN message into POBs and sends them to the BBL for execution.

Translating plan tasks into POBs does not always involve a one to one translation. Depending on which direction the robot is facing, the robot might have to turn, meaning that step will get translated into two POBs. The plan task move will get translated into multiple POBs depending on how far the robot has to travel to get to its destination. For example, the first plan step has the robot move from node 1 to node 6; to get to node 6 the robot has to go through nodes 2, 3 4, and 5, each time the robot has to use the POB follow line. In addition the robot has to turn right to get from node 3 to 4 and right again to get from node 5 to 6, so the first plan task is translated into 7 POBs. After the first task the robot will be facing south, for the robot to complete the second task the robot has to turn right before picking up the block, so the second task is translated into two POBs. Likewise, the third task is translated into 8 POBs and the last task is translated into 2. Altogether the plan is translated into 19 POBs, shown in Figure 9.

```
FOLLOW LINE
FOLLOW LINE
TURN RIGHT
FOLLOW LINE
FOLLOW LINE
TURN RIGHT
FOLLOW LINE
---
TURN RIGHT
PICKUP
---
TURN RIGHT
FOLLOW LINE
TURN LEFT
FOLLOW LINE
TURN RIGHT
FOLLOW LINE
TURN LEFT
FOLLOW LINE
---
TURN LEFT
PUT DOWN
```

Figure 9. POBs generated from the plan in Figure 8.

4. Results and Contributions

This paper presents a system that enables a resource limited robot to formulate plans for high-level tasks by using a powerful but resource hungry planner. The resource limited Khepera III robot runs an INTERRAP controller. The powerful SHOP2 planning system runs on a workstation that has adequate memory and processor speed. The robot's INTERRAP controller communicates with the SHOP2 planner when it needs a high-level plan.

A drawback of this system is the overhead required for the resource limited robot to formulate the problem definition and send it to the workstation and to receive, decode and translate the workstation's plan into POBs The communication between the robot and the workstation also creates a delay.

The benefit of this approach is that it enables the resource limited robot to expand its planning capabilities by using existing powerful planners. It also serves as a model of how to interface a sophisticated planner with the INTERRAP architecture's local planning layer. A similar approach could be taken to expand the capabilities of the cooperative planning layer.

References

"K-Team Corporation | Mobile Robotics." K-Team Corporation | Mobile Robotics. HTTP://www.k-team.com/ (accessed August 15, 2012).

Lind, J. "*Patterns in agent-oriented software engineering." In: Proceedings of the 3rd international conference on Agent-oriented software engineering III,* AOSE'02, pp. 47–58. Springer-Verlag, Berlin, Heidelberg, Germany (2003)

Müller, Jörg P.. *The design of intelligent agents: a layered approach*. Berlin, Springer, 1996.

Nau, Dana , Héctor Muñoz-Avila, Yue Cao, Amnon Lotem, and Steven Mitchell. "*Total-Order Planning with Partially Ordered Subtasks.*" IJCAI 1 (2001): 425-430.

Nau Dana, Tsz-Chiu Au, Okhtay Ilghami, Ugur Kuter, J.

William Murdock, Dan Wu, and Fusun Yaman. "*SHOP2: An HTN Planning System.*" Journal of Artificial Intelligence Research 20, no. 1 (2003): 379-404.
P. Doerschuk, L. Wang, A. Keswani, N. M. Lingala, T. S. Mokha, D. O. Doerschuk. Communication Language for Autonomous Multi-Robot System. In Proceeding of the 12th IASTED International Conference ROBOTICS AND APPLICATIONS. Aug 06, Honolulu, Hawaii, USA.
"*Simple Hierarchical Ordered Planner.*" Department of Computer Science. HTTP://www.cs.umd.edu/projects/shop/ (accessed March 21, 2012).