Artificial Intelligence Research and Development
À. Nebot et al. (Eds.)
IOS Press, 2016
© 2016 The authors and IOS Press. All rights reserved.
doi:10.3233/978-1-61499-696-5-19

Completing Ambiguous Plans in Cognitive Robotics with Delayed Information

Sai Kishor KOTHAKOTA [a] and Cecilio ANGULO [b,1]
[a] *SASTRA University, Thanjavur, India*
[b] *Universitat Politècnica de Catalunya, Barcelona, Spain*

Abstract. Robots equipped with a set of simple action skills should complete complex tasks, defined as the concatenation of a number of those basic abilities. Traditionally, planners have been used to decide skills to be activated, as well as in which sequence, like state machines. Recently, cognitive architectures like SOAR have been proposed to act as the reasoner by selecting which competence the robot should perform, addressing it towards the goal. However, they have been unidirectionally integrated: once the plan is completely designed by the cognitive architecture, it is sent to the robot, but no feedback is provided to the reasoner. Instead, our proposal allows to establish bi-directional communication between the reasoner and the robot. In this form, the reasoner can develop incomplete plans under the assumption that a part of the information to complete the plan for achieving the goal will arrive delayed from the robot's environment as well as the user. Our work develops this bi-directional communication between the SOAR cognitive architecture and the ROS (Robot Operating System) environment, usual in mobile robotics. The proposed architecture has been tested on a UAV (Unmanned Aerial Vehicle) Parrot AR.Drone 2.0, which acts as mobile robot, in a searching task.

Keywords. Cognitive architecture, Mobile robot, ROS, SOAR, Bi-directional communication

1. Introduction

Mobile robotics is a research area for living a comfort life and for faster development of technology. Even though the mobile robots are used for specific applications, they lag in performing actions from decisions taken in real-time scenarios and also interaction with human at the time of ambiguity [5].

The usual approach for planning in intelligent mobile robots is mainly based on state machine technology, where a given goal is achieved with a proper complete plan available before its implementation. This complete plan is a sequential combination of simple robot skills to achieve the goal. Since no feedback from the environment is considered, robot behaves in the same manner in different situations of the environment. So, using a complete planning from the start the robot proceeds with the commands having little

[1]Corresponding Author: Universitat Politècnica de Catalunya, Pau Gargallo 5, 08028 Barcelona, Spain; E-mail: cecilio.angulo@upc.edu.

knowledge about the environment, but when the selected action was successful [14]. The main drawback of this approach is that too much information is needed preceding the start of decision making, which prevents the robot to react to novel situations.

The most common alternative to state machines are planners. Planners are usually based on probabilistic criteria that determine the best skill combination in real-time for a specific goal. A different approach to planners is the use of reasoners i.e., the use of cognitive architectures which has attracted a renewed attention both from academics and industry [4]. Hence, a cognitive architecture has been applied to a general service robot in [14] by establishing a uni-directional communication from the architecture to the robot.

Bi-directional communication between the environment/robot and the cognitive architecture would help to process the goal with incomplete plans based on delayed information coming from the environment. The cognitive architecture with incomplete planning would also use information available from the user to build plans. Hence, robots using this type of architecture could re-plan according to environmental data and could also establish a user interactive environment, which prevents providing huge data before-hand and completes the plan as it reaches the specified goal.

Several cognitive architectures [13] such as SOAR [6,11,17], ACT-R [3,2,16], CRAM, SS-RICS [7,8] exist in the literature. They were evaluated in [14] in a general purpose robotic scenario based on their generalization, reasoning, learning, sub-goal capacity and scalability. Using the same discussion, SOAR is also selected for this study. Other architectures that can be considered are ICARUS [12], using Horn clauses that define abstract goals, and DISCIPLE [15], based on concept hierarchies to represent knowledge.

SOAR is a cognitive architecture that has been under continuous development since 1980s. It is also able to communicate with the external environment. SOAR provides special features that, when there is no plan to achieve goal, it can try to implement several mechanisms and available knowledge such as chunking, reinforcement learning, and sub-goal capacity. This cognitive architecture can select the skill for current situation to achieve the specified goal without any pre-defined list about situations and plans. This will be a key feature for establishing bi-directional communication with the surrounding environment [10].

Moreover, SOAR has been recently applied for control of general purpose service humanoid robot [14] in order to solve complex tasks using the basic skills of humanoid service robot like navigation, grasp and object recognition. However, only uni-directional communication between SOAR and ROS is available, and user intervention is not being considered.

Hence, the main goal in this work is to develop a bi-directional communication between SOAR cognitive architecture and ROS environment. This feature will endow mobile robots with a closed control architecture which allow them to generate and execute their own plans to accomplish a goal based on the feedback from the environment, but also from the user interaction. The goal or complex task will be achieved like a combination of several skills implemented in the robot. Bi-directionality allows for delayed information coming from the environment, changing plans on-the-fly. Furthermore, this feature can be used for interactive environments, the robot acting differently in particular situations.

The rest of the paper is organized as follows: the implemented architecture is introduced in Section 2; in Section 3, the robot platform, a general purpose service robot, is described; Section 4 highlights the main results obtained during experimentation; finally, some conclusions and future research lines are offered.

2. Bi-directional SOAR-ROS Communication

SOAR has been selected in our approach for robot planning. In SOAR, a state is a representation of the ongoing problem-solving situation, an operator transforms a state (which makes changes to the representation) and a goal is a desired outcome of the problem-solving activity i.e., the virtual solution which represents the way to solve it.

The proposed overall system is composed of three modules that are mutually connected. The cognitive architecture (SOAR) sends commands according to the received real-time data. Next, the Interface between SOAR and ROS, which is our developed implementation, receives commands from SOAR, processes them into ROS commands and then takes the data from ROS environment or other sources, processes them to be sent into the SOAR cognitive architecture to process future decisions based on the input data.

The interface to communicate with the external environment or robots by sending and receiving commands is programmed using SML (Soar Markup Language), where commands are packed as XML packets and sent. The environment and the debugger to which it supports are referred as clients.

The proposed bi-directional communication between SOAR and ROS will be possible by considering SOAR environment as the master element. Therefore, commands sent from ROS are processed in SOAR and commands from SOAR to ROS are correctly processed in SOAR and formatted for an easy evaluation in ROS. By using the communication in this form, both user and robot's environment inputs can be processed into SOAR for decision making. Hence, this SOAR-oriented interface will allow bi-directionality as well as user intervention.

2.1. Working Memory

Stored short-term information and tested structures from generated rules will be placed in the working memory. It is built of graph structure of elements, where each element is created by rules and from the sensors or data from external environment. SOAR architecture automatically creates some of the work memory structures for every agent, whose elements are built as triplets: identifier, attribute, and value. SOAR automatically creates an `io` attribute, which stands for `input-output` to communicate with the external world. There are two attributes for `io` (see Figure 1): `input-link` and `output-link`:

- The `output-link` attribute in the working memory structure is used to send commands to the external world in order to perform some actions with each decision of the SOAR architecture. The `output-link` is where action commands must be created for the agents in its world.
- The `input-link` attribute in the working memory structure is used to get the information from the external world or from sensors. Normally, for the task involving the information from the external environment, most of the states are built from the perceptual information available from the sensors. This information would be created on the `input-link` structure of the state. it can be also used to take the user input and make it available on the `input-link` structure.

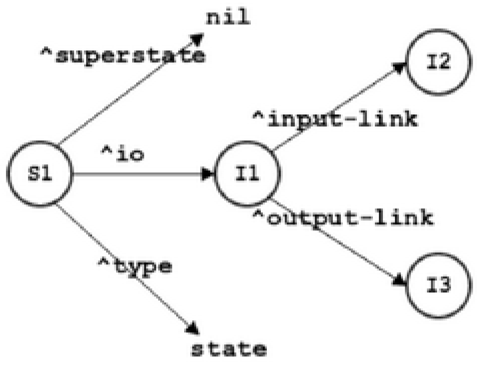

Figure 1. Graphic picture of the working memory structures that SOAR automatically generates.

2.2. Sending Commands From and Receiving Data Into SOAR

SOAR helps the user to communicate information by providing the `input-link` and `output-link` structures of the working memory. Using these structures, it is possible to send commands from SOAR and do operations successfully. In a similar form, we can use the `input-link` structure to get data into SOAR (from ROS in our case) for computational or decision oriented purposes. So, data from on-board sensors in the robot can be obtained and use them for decision making.

2.3. The Interface Client

In order to build communication with the ROS environment, an interface client must be designed so that it can access the `input-link` and `output-link` structures for processing the communication of sending/receiving data into/from SOAR cognitive architecture. This connection from the interface to the SOAR cognitive architecture can be established using the SML (Soar Markup Language) clients. These clients help the interface to establish a successful connection for the transfer of data between them.

SOAR provides different SML client files for different programming languages (C++, Python, Java) and these files can be used to establish connection between the programmed interface and SOAR cognitive architecture.

Actions selected by the SOAR architecture will be translated into skill activation's in the ROS environment by means of the interface. These ROS commands will then be reflected in the actions of the robot, which are the steps to achieve the specific goal.

The SML clients used from the interface file helps the interface to establish a kernel connection to SOAR cognitive architecture. The client can help in either, creating a local SOAR kernel or a remote connection to an existing SOAR kernel. This kernel object can be used to create a SOAR agent. The created SOAR agents are in charge for managing the work memory elements in the working memory. This agent helps in getting information from SOAR architecture and also sending information into it (see Figure 2).

These agents are in charge of sending data through the `input-link` structure by creating new input-link working memory elements (WME), loading production files, and receiving commands from SOAR as well as the parameters of these commands by accessing the `output-link` structure in the work memory. The SOAR agents are required in creating `input-link` WME, updating `input-link` attribute values, and destroying

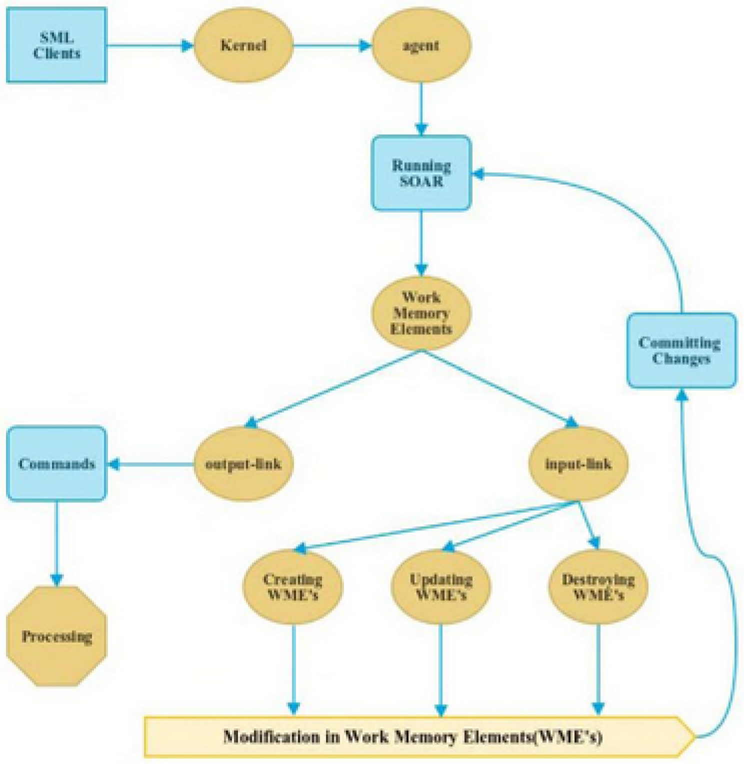

Figure 2. Interface development from the SML client.

WME. Hence, design of working memory structures using agents is an important phase in the interface client design.

2.4. Dealing with Working Memory Elements

SOAR agents are in charge to create, update and destroy Working Memory Elements (WMEs), in particular `input-link` WME, where external data is obtained. The interface client can build- up arbitrarily complex graph of WMEs attached to the `input-link` structure.

We can also update the values that were present in the `input-link` structure with the agent created using the SML clients. In this form, it is possible to update/overwrite data in the attributes of the input-link structure in the work memory.

Finally, SML clients also provide a feature to destroy a Working Memory Element (WME) in the SOAR interface client. A WME value can be removed by destroying it, which also makes working memory object invalid. By this way, we can destroy the required WMEs using the SOAR agent.

It should be noticed that the interface client explicitly requests that changes made in WMEs be sent over to SOAR. This explicit command allows the communication layer to be more efficient, by committing all the changes together and sending them as a single command.

2.5. Running SOAR with the Interface Client

Firing a rule in SOAR is determined by comparing them with the data structures present in the working memory. This working memory notifies the current scenario where the agent constructs the perception of the world around it. Next, the WME changes after successful processing of each rule, which updates the working memory structures with new values, helping SOAR to take further decisions based on that, making it a dynamic data structure.

The agent created using the SML clients can be used to run SOAR until it receives a certain number of commands to interface. It can also be used to run agent forever and stop whenever it is required. Finally, it can also be used to get the flow of commands until it achieves either output or its goal. In most of the cases, the flow of commands until the output is used.

In order to check the status of commands from SOAR, methods `GetCommandName` and `GetParameterValue` extract the appropriate attributes and values from the `output link` structure of the WME and return them to the environment of the interface in which the instance of the kernel is created.

Moreover, the method `AddStatusComplete` adds to the command structure of the working memory, indicating to the agent in the interface that this command has been successfully executed and can now be removed by the agent.

2.6. The Complete View

In order to design an interface client using SML that connects SOAR with the external environment, the following design steps are needed: creating a kernel instance, creating a SOAR agent, loading productions, creating WMEs, updating WMEs, destroying WMEs, committing changes, running SOAR, retrieving commands, and using the command line. Besides, the interface client also needs the ROS part contribution to accomplish the complete task. ROS program for the robot can be done separately, those files are imported in the interface and they are used for the successful completion of the operations according to the SOAR decision.

SOAR commands are received by the interface client and then, according to the command received to the interface, it is processed by the interface and sent out as ROS commands. Depending on the ROS commands, the robot will complete the operation and then a ‘status completed’ confirmation is sent to the SOAR environment, to make sure that every operation has been completed successfully. After the successful operation, Work Memory Elements are changed by the SOAR for a better mapping for the future decisions that need to be made in order to successfully complete the task or goal. The information from ROS can be sent into SOAR by creating Work Memory Elements (WMEs) and adding them to the input-link structure of the WME, hence external environment can be made available to SOAR for decision making.

3. Experimentation Setup

For the experimentation of the bi-directional communication between SOAR cognitive architecture and ROS environment the UAV (Unmanned Aerial Vehicle) mobile robot

Table 1. Available robot's skills and their associated actions.

Skill	Action
Forward	Pitch forward
Reverse	Pitch backward
Up	Increase altitude
Down objects	Decrease altitude
Side	Rolling right
Sidel	Rolling left
Right	Yaw towards right
Left	Yaw towards left

Parrot AR.Drone 2.0 has been employed. This UAV is equipped with many on-board sensors and a controller to hold a stabilized flight, hence it can hover at a spot waiting, before receiving the next command.

AR.Drone is an interesting research platform for computer vision and robotic exploration [9]. In the experimentation process, SOAR will employ simple skills of the robot such as increasing/decreasing altitude, moving forward/reverse, rolling right/left, yaw towards right/left in order to accomplish a specific goal (see Table 1). The Parrot AR.Drone will be controlled using ROS environment, so executing ROS commands from the interface which are processed from the SOAR decision commands makes the drone to perform the skills or actions accordingly.

Parrot AR.Drone 2.0 responds to the ROS commands with the use of ardrone_driver and ardrone_autonomy packages in ROS environment[2]. These drivers in the interface helps the commands received from SOAR cognitive architecture to be processed into ROS environment. ROS environment provides the navigation data from the on-board sensors before, during and after the flight. The available sensor readings are altitude, magnetometer, gyroscope, and accelerometer signals, barometer readings, temperature value, wind angle and many more. These can be made available to SOAR cognitive architecture and help it in making decisions.

The goal in the experimentation is to find an Augmented Reality (AR) marker during the UAV flight and eventually landing based on user input acceptance. AR Markers detection will be performed with the ar_pose package, which is an augmented reality marker pose estimation using ARToolkit [1]. The UAV will search for an AR marker during its flight and send the information about its presence to SOAR architecture through the interface. This data is used by SOAR cognitive architecture for decision making.

With this experiment we are able to show the bi-directional communication between SOAR and ROS, sending commands from SOAR and receiving data from ROS in the developed interface, as well as how the user can also send commands in order to complete the plan by accepting landing or eventually rejecting it. The experimentation has been made available at `http://tiny.cc/ardrone`.

[2]http://autonomylab.org/

3.1. Implementation

Implementation has been designed in a closed but uncontrolled environment, a corridor in a public building. In order to meet the goal, the robot can work on two types of interactions:

- **Category 1.** Using bi-directional communication between SOAR cognitive architecture and ROS environment like data of presence of AR marker with the allowed sub-commands for the robot, which are the ones defined in Table 1.
- **Category 2.** User Intervention whose data can be used for decision making.

In Category 1, according to the presence of an AR marker, the SOAR module takes decisions in a form of feedback from the system: the SOAR module modifies the plan in real time in the way to achieve the goal by sensing the environment. During the skill operation, the presence of AR markers is audited and sent to the SOAR module before the next decision is taken and this information is used by SOAR to make decisions such a way that it knows whether it found an AR marker in the previous operation. It tries to find AR Marker by this method and lands once it finds an AR Marker.

In Category 2, the user is asked regarding landing the UAV after the AR marker is found: "*Enter 1 to land, 0 to hover for some time and land*". Depending on the user input, the robot will either immediately land or do it after hovering for a certain period. Hence, the user intervention is introduced into the SOAR-ROS communication.

4. Results

The bi-directional SOAR-ROS communication has been tested in a closed environment in the process of a Parrot AR.Drone 2.0 searching for an AR marker. In this experimentation, the robot performs operations based on the ROS commands that are processed in the interface from the decision commands received from the SOAR architecture and then feedback about detection of an AR marker is sent to SOAR for decision making. Depending on the user decision, the plan changes eventually and the skills are determined by SOAR module for achieving the goal in a proper manner. During the implementation, both categories, as described in Section 3.1, have been tested.

Testing involves to provide the robot system with a required user input in command line, and checking that the robot is able to perform the required actions to complete the goal. Situations where the robot is tested are:

Category 1. :"*Search for an AR marker and land*". The sequence of actions performed by the robot is: *forward, check for AR Marker* x 5, *side, check for AR Marker*, *reverse, check for AR Marker* x 5, *side, check for AR Marker*, *forward, check for AR Marker* x 5 and so on (up to a certain limit of side operations) which shows that initial complete planning is not needed in this case.

Category 2.:"*User Intervention*" When an AR marker is found, then SOAR is notified about its presence through a feedback system in interface. Then the user input among the available options will be asked as the AR marker is recognized. When the user takes an option, SOAR will perform an action according to the user interaction / intervention in the system.

The system introduced in this paper guarantees that the actions proposed will lead to the goal, so the robot will find a solution based on the feedback from ROS environment and from user intervention. This situation can not be declared to be optimal. For instance, in some situations, the robot might move to a location that was not right path for the command executed, before moving on a second action it need to step to the correct one. However, the completion of the task is guaranteed since the architecture will continuously provide steps until the specific goal is accomplished.

5. Conclusions and Further Research

The bi-directional communication between the SOAR cognitive architecture and the ROS environment has been introduced and tested on a mobile UAV (Unmanned Ariel Vehicle) robot Parrot AR.Drone 2.0 for solving a complex task by the combination of simple skills. Bi-directional communication is a key issue to increase information for taking better decision by the SOAR reasoner. In this form, incomplete plans can be considered, information from the environment can be introduced in the reasoner, user can interact in the planning process by modifying / completing it.

The introduced bi-directional communication has improved on two main drawbacks from the previous SOAR-ROS communication approach. Firstly, completing the incomplete plans of a specified goal based on the real time data available from the environment. Secondly, User Intervention which helps the architectures to communicate directly with humans and take suggestions or input and use them for decision making. This feature makes them to get adapted to real world and in more intelligent manner.

As a drawback, since partial plans are allowed when starting the plan execution, the SOAR cognitive architecture cannot disclose whether the goal requested to the robot will be completely reachable or not. However, as the user action and environment feedback are now allowed, so re-plan is made easier as well as user intervention based on the environmental information helps better for reaching goal.

Future research lines are now opened. Using this bi-directional communication, a single architecture can be designed with well defined simple robot skills and feedback from the on-board sensors can be taken and computed. Hence, the robot can behave in different manners with different environments, modifying plans and helping to achieve the goal.

Moreover, the current implementation can be improved in terms of robustness by solving a known issue. Mainly, if one of the actions is not completely achieved (for example, the robot is not able to reach a position in the space because it is occupied, or the robot cannot find an object that is in front of it), the skill activation will fail. In the current implementation, as well as in the previous ones, the robot has no means to discover the reason of the failure. Hence the robot will detect that the state of the world has not changed, and it will select the same action (retry) towards the goal accomplishment. This behaviour could lead to an infinite loop of retries. For the safety of the mobile UAV robot, when this situation occurs the drone lands at the point of time when it recognizes the occurrence of same commands.

Acknowledgments

This work was possible with the support from Universitat Politècnica de Catalunya (UPC) and SASTRA University, with their exchange program. This research was supported in part by the PATRICIA Research Project (TIN2012-38416-C03-01), funded by the Spanish Ministry of Economy and Competitiveness.

References

[1] D. Amin and S. Govilkar, Comparative study of augmented reality SDK's, *International Journal on Computational Sciences & Applications* **5**(Feb) (2015).

[2] J. R. Anderson, *How can the human mind occur in the physical universe?*, Oxford University Press, New York, 2007.

[3] J. R. Anderson, D. Bothell, M. D. Byrne, S. Douglass, C. Lebiere, and Y. Qin, An integrated theory of the mind, *Psychological review*, **111**(4) (2004), 1036–60.

[4] T. R. Besold, A. D'Avila-Garcez, K. U. Kühnberger, and T. C. Stewart(eds.), Neural-symbolic networks for cognitive capacities (Special issue), *Biologically Inspired Cognitive Architectures* **9** (2014), 1–122.

[5] G. Canal, C. Angulo, and Sergio Escalera, Human multi-robot interaction based on gesture recognition, *International Joint Conference on Neural Networks* (2015). (In Press)

[6] B. Cho, P. S. Rosenbloom, and C. P. Dolan, neuro-Soar: A neural-network architecture for goal-oriented behavior, *Proceedings of the Thirteenth Annual Conference of the Cognitive Science Society* (1991), 673–677.

[7] T. D. Kelley, Developing a psychologically inspired cognitive architecture for robotic control : The symbolic and subsymbolic robotic intelligence control system, *Internation Journal of Advanced Robotic Systems* **3**(3) (2006), 219–222.

[8] T. D. Kelley and E. Avery, A cognitive robotics system: the symbolic and sub-symbolic robotic intelligence control system (SS-RICS), *Multisensor, Multisource Information Fusion: Architectures, Algorithms, and Applications 2010* **25** (2010), 460–470.

[9] T, Krajník, V. Vonásek, D. Fišer, and J. Faigl, AR-Drone as a platform for robotic research and education, *Communications in Computer and Information Science* (2011), 172–186.

[10] J. E Laird, Extending the Soar cognitive architecture, *Artificial General Intelligence Conference* (2008).

[11] J. E. Laird, K. R. Kinkade, S. Mohan, and J. Z. Xu, Cognitive robotics using the Soar cognitive architecture, *International Conference on Cognitive Modelling* (2004), 226–230.

[12] P. Langley and D. Choi, A unified cognitive architecture for physical agents, *National Conference on Artificial Intelligence* (2006), 2006.

[13] P. Langley, J. E. Laird, and S. Rogersa, Cognitive architectures: Research issues and challenges, *Cognitive Systems Research* (2009).

[14] J.-Y. Puigbo, A. Pumarola, C. Angulo, and R. Tellez, Using a cognitive architecture for general-purpose service robot control, *Connection Science* (2015).

[15] G. Tecuci, M. Boicu, D. Marcu, B. Stanescu, C. Boicu, J. Comello, A. Lopez, J. Donlon, and W. Cleckner, Development and deployment of a DISCIPLE agent for center of gravity analysis, *Conference on Innovative Applications of Artificial Intelligence* (2002), 853–860.

[16] J.G. Trafton, N.L. Cassimatis, M.D. Bugajska, D.P. Brock, F.E. Mintz, and A.C. Schultz, Enabling effective human-robot interaction using perspective-taking in robots, *IEEE Transactions on Systems, Man and Cybernetics*, **25** (2005), 460–470.

[17] R. M. Young and R. L. Lewis, The Soar cognitive architecture and human working memory, *Models of Working Memory: Mechanisms of Active Maintenance and Executive Control* (1997).