Journal of Korea Robotics Society (2017) 12(1):055-064
https://doi.org/10.7746/jkros.2017.12.1.055 ISSN: 1975-6291 / eISSN: 2287-3961

Soar (State Operator and Result)와 ROS 연계를 통해 거절가능 HRI 태스크의 휴머노이드로봇 구현

Implementation of a Refusable Human-Robot Interaction Task with Humanoid Robot by Connecting Soar and ROS

당반치엔[1], 트란트렁틴[1], 팜쑤언쭝[1], 길 기 종[1], 신 용 빈[1], 김 종 욱[†]

Chien Van Dang[1], Tin Trung Tran[1], Trung Xuan Pham[1], Ki-Jong Gil[1], Yong-Bin Shin[1], Jong-Wook Kim[†]

Abstract This paper proposes combination of a cognitive agent architecture named Soar (State, operator, and result) and ROS (Robot Operating System), which can be a basic framework for a robot agent to interact and cope with its environment more intelligently and appropriately. The proposed Soar-ROS human-robot interaction (HRI) agent understands a set of human's commands by voice recognition and chooses to properly react to the command according to the symbol detected by image recognition, implemented on a humanoid robot. The robotic agent is allowed to refuse to follow an inappropriate command like "go" after it has seen the symbol 'X' which represents that an abnormal or immoral situation has occurred. This simple but meaningful HRI task is successfully experimented on the proposed Soar-ROS platform with a small humanoid robot, which implies that extending the present hybrid platform to artificial moral agent is possible.

Keywords Soar, ROS, Human-robot interaction, Artificial moral agent

1. Introduction

Intelligent robots are the machines equipped with a diverse set of visual and non-visual sensors and actuators, which possess decision making and problem solving capabilities within their domain of operation [1,2]. As the intelligent robots are gradually coexisting with human in wider areas such as service robots, social robots, and healthcare robots, a basic level of morality or ethics is necessary for safe and natural HRI tasks.

After the pioneering work of Asimov's three laws of robotics, architectural mechanisms to enable the agent to reason and determine ethically about their actions have been proposed [3,4]. In [4], the authors employed a cognitive robotic architecture, DIARC/ADE, in determining whether a human's directive should be accepted or rejected according to the excuse categories. However, the architecture is mainly focused on HRI and natural language processing, and thus

Received : Oct. 15. 2016; Revised : Dec. 16. 2016; Accepted : Jan. 31. 2017
※This work was supported partly by the Ministry of Trade, Industry & Energy (MOTIE, Korea) under Industrial Technology Innovation Program. No. 10062368, and partly by the MSIP (Ministry of Science, ICT and Future Planning), Korea, under the ITRC (Information Technology Research Centre) support program IITP-2016-H8501-16-1017) supervised by the IITP (Institute for Information & communications Technology Promotion).
†Corresponding author: Department of Electronic Engineering, Dong-A University, Busan, Korea (kjwook@dau.ac.kr)
[1]Department of Electronic Engineering, Dong-A University, Busan (dvchienbkvn@gmail.com)

it has limitation in general cognitive capabilities such as semantic, episodic, and procedural learning and memories, which are essential for ethical reasoning based on sufficient knowledge and experience.

In the artificial intelligence society, human-level cognitive agent architectures have long been developed and disseminated worldwide including ACT-R [5], CRAM [6], SS-RICS [7], and Soar [8]. Soar is reported to have superiority over the well-known cognitive architectures in terms of complete symbolic reasoning, easily understandable structure, and simple programming complexity, and complex implementation in low-level control, which is a consequence of a long development history since 1981 [9].

Soar has been rarely applied to robotics technology despite its rich components of various memories and learning functionalities. Therefore, a successful coordination of the cognitive architecture and ROS [10] will be a higher level integrated development environment (IDE) available to the roboticists whose goal is developing more human-like robots.

In the present paper, the authors propose a hybrid platform which integrates two different types of software platforms, Soar and ROS. The former helps robots restore the environment information and process on them with intelligence, and the latter provides robots with useful libraries to respond well to the situation where the robot is deployed. This study will cover connection of the two platforms and creating Soar agents to help a humanoid robot make reasonable and refusable decisions on an HRI task.

In this paper, Section 2 outlines the structure of Soar and its decision cycle. This section also briefly describes ROS platform as an environment for the soar agent and how messages are shared in ROS. In Section 3, the proposed connection strategy is described. Section 4 provides an experiment result for a simple HRI task implemented on the present hybrid platform with a small humanoid robot, and Section 5 concludes the present work with remarks on future research.

2. Soar and ROS

Soar and ROS are both software frameworks on which libraries and applications are executed. For the connection, the two structures are analyzed and investigated on how data moves.

2.1 Soar

Soar is a cognitive agent software architecture, which considers problem solving as application of an operator to a state to get a desired result [11,12]. Soar integrates knowledge-intensive reasoning, reactive execution, hierarchical reasoning, planning, and learning from experience, with the goal of creating a general computational system that has the similar cognitive abilities as humans. Soar is both a software platform for agent development and a theory of what computational structures are necessary to embody human-level agents.

Fig. 1 illustrates the Soar architecture composed of working memory, decision cycle, and long-term memories including procedural memory, semantic memory, and episodic memory, which correspond to their learning mechanisms. Each of the memory systems in Soar will

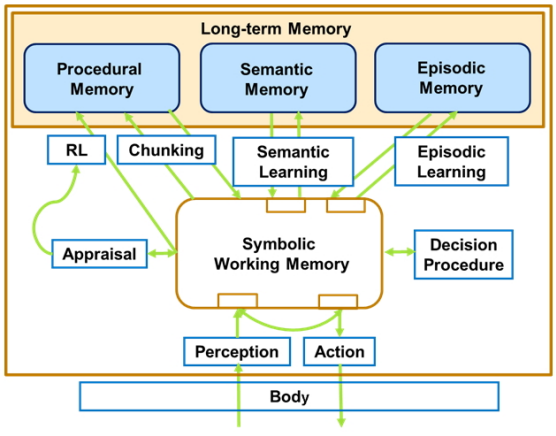


Fig. 1. Soar architecture

```
sp {hello-world
    (state <s> ^type state)
-->
    (write |Hello World|)
    (halt)}
```

Fig. 2. Soar rule for hello-world

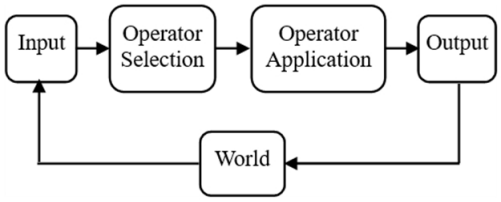


Fig. 3. Soar processing cycle

be briefly introduced and compared below.

Firstly, the working memory is the short-term memory that represents the current problem-solving situation. This memory holds the current state and operator, or the current knowledge of the world and the status in problem solving. Secondly, the procedural memory is the long-term knowledge which encodes procedural knowledge. Thirdly, the episodic memory deals with specific events which are snapshots of working memory images. In other words, it is a record of an agent's experience. Finally, the semantic memory restores general facts independent of specific contexts. These memories are like those of human, helping a software agent interact with external environment.

Soar acquires knowledge by production rules. If the conditions of a production rule match with those of working memory, the production fires, and the actions are performed. These production rules are to be understood as "if-then" statements in the conventional programming languages as shown in Fig. 2. The "if" part of the production is called its condition and the "then" part separated by "-->" is called its action. When the conditions of some production rules are matched with the current situation defined in the working memory, the corresponding production rules will fire in parallel, which means that their actions are executed and make changes to working memory.

A Soar agent is created with rules and applied to send commands out via output link based on the current context. The decision of the Soar agent is governed by a reasoning cycle as shown in Fig. 3, where the agent chooses an appropriate operator based on the information it knows about its current situation, i.e., working memory, and its long-term knowledges and memories. This cycle can be divided into four phases.

In the input phase, Soar agent can receive information from a simulated or real surrounding environment. In the operator selection phase, the current state is elaborated and one or more operators are proposed depending on the contents of the working memory. If more than one operator have been proposed, Soar has a mechanism to help its agents attempt to choose the most appropriate operator for the current situation using preference symbols such as acceptable (+), reject (-), indifferent (=), better/best (>), worse/worst (<). In the operator application phase, the selected operator is fired and make changes to the current working memory. In the output phase, Soar agent sends information to external environment or its actuators such as motors.

2.2 ROS

ROS is a sophisticated open source robot middleware platform. It can provide a communication framework, hardware abstraction, device drivers with a lot of useful libraries and tools to quickly develop robot applications. ROS consists of a number of executables called nodes which are connected in a peer-to-peer network topology. These nodes perform the system's computation and communicate with other nodes through the message exchange mechanism in ROS.

ROS supports two types of messaging mechanisms between nodes; topic and service. A topic associates publisher and subscriber where a publisher broadcasts topic messages to the whole ROS system, while the

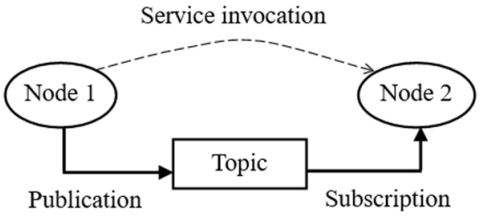


Fig. 4. Communication between nodes in ROS

subscribers listen to the published information on this topic. There may be more than one publisher or subscriber for the same topic. On the other hand, service is a pair of request and response. Unlike the topic, there is only one provider to handle the request for each service. In general, basic communication between nodes in ROS can be described in Fig. 4.

3. Connection of Soar and ROS

With Soar and ROS software platforms mentioned above, a method is required to bridge the two architectures which must ensure the processing cycle of Soar as well as the message exchange mechanism in ROS.

Fortunately, Soar provides users a tool named Soar markup language (SML) which allows to build an interface for collaboration of two or more Soar agents [12]. The interface is also designed to support connecting other software development environments to Soar where input and output data structures are sent back and forth. In addition, ROS provides users with the two types of communication mechanisms, i.e., topic and service, which can be created in a package to send and receive messages from other packages.

In this paper, the authors propose a connection method to bridge Soar and ROS by creating a Soar wrapper inside a ROS package. This Soar wrapper package plays a key role as a common ROS package which subscribes to topics from other packages and publishes topics as well. Moreover, the Soar wrapper package with SML tool puts input data to input link of Soar agent and reads

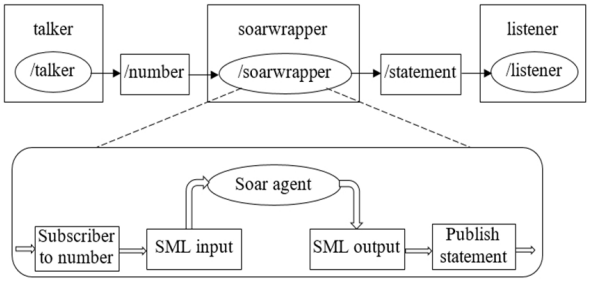


Fig. 5. Soar-ROS connection example graph

output data from its output link. By using this method, Soar wrapper becomes a ROS package obeying ROS communication framework and also following Soar reasoning cycle.

Fig. 5 depicts the Soar-ROS example graph as a first successful result of Soar-ROS association, where according to the input number between 0 and 9 the capital city for the corresponding country is printed as an output using semantic memory of Soar [13]. The figure clearly points out that there are three nodes (talker, soarwrapper, and listener) which exchange messages via the “number” topic from the talker node to the soarwrapper node, and the “statement” topic from the soarwrapper node to the listener node. Inside the soarwrapper is a Soar agent that gets identifier number using SML input and make decisions according to it.

SML output reads decision commands from the Soar agent via output link and publishes them. This is how Soar agent interacts with other environments. The rules of the Soar wrapper agent are shown in Fig. 6 written in plain words.

As the next step of connecting Soar and ROS, the authors attempt to apply this Soar-ROS hybrid platform into a humanoid robot ROBOTIS-OP2 (OP2 hereafter) [14]. The OP2 robot has been well known in the literature [15,16] which proved the robot appropriate for research and development. Fig. 7 shows an overall structure of the Soar-ROS hybrid platform implemented on OP2.

Soar agents in Fig. 7 are inherited from the soarwrapper

Propose*soar-ros-task
If no task is selected,
then propose the initialize-soar-ros-task.
Apply*soar-ros-task
If the initialize-soar-ros-task operator is selected,
then create an agent soar-ros and a semantic memory.
Propose*read-input-link
If a state is named soar-ros and input-link has not been read,
then propose reading input link.
Apply*read-input-link
If a state is named soar-ros and the reading input link operator is selected,
then read input link.
Propose*retrieve-semantic-memory
If a state is named soar-ros and input matches within semantic memory,
then propose retrieving semantic memory.
Apply* retrieve-semantic-memory
If a state is named soar-ros and retrieving semantic memory is proposed,
then retrieve contents in semantic memory and put commands to output link.

Fig. 6. Pseudocode for the soar wrapper agent

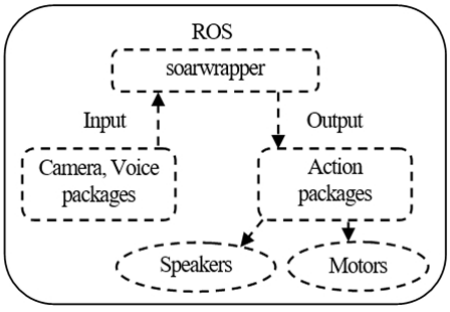


Fig. 7. Structure of Soar-ROS connection

node mentioned above. The Soar agents are integrated in a ROS package, and thus it can exchange information with other packages in ROS. Packages for machine vision, voice and image recognition, voice synthesis, and various OP2 actions are all ROS packages which can be obtained from the worldwide ROS communities. They individually interact with Soar agents through the publishing and subscribing mechanism of ROS topics.

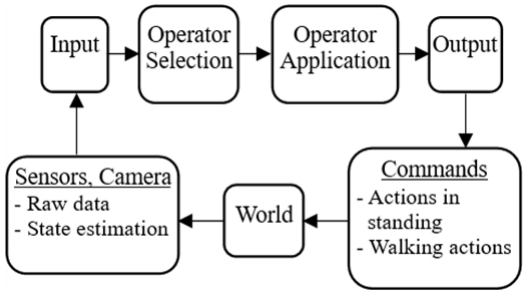


Fig. 8. Soar cycle model interacting with outside world

Fig. 8 shows up a developed reasoning processes where in fact the approaching method points out how Soar agents understand the environment.

4. Experiment Results

Below we present a demonstration of cooperation between Soar and ROS for an HRI task where OP2 follows the human's instructions and it may reject a command if the command may lead to damage or harm for human and/or robot, which is a new trend for more safe and natural HRI.

In the present HRI scenario, we begin with the first command when the robot is ready with a sit-down position. At this state, robot is considered to be in a safe situation. A more advanced robot will obey human's order unless the result might give a harm to robot itself and others. To this end, there are two marks in the present scenario; an O mark means a safe situation, whereas an X mark denotes a dangerous situation. That is to say, when the robot sees the X mark, it will only follow the commands predicted to be safe such as sitting down or shaking its head for expressing yes or no. Since walking forward in this case can cause the robot to be broken by clashing with other robots or falling from a table for example, the robot will deny such a bad commands and say "No, I cannot do it." by the present scenario. It will continue refusing to the inappropriate command until it sees the O mark, which indicates that all the unsafe

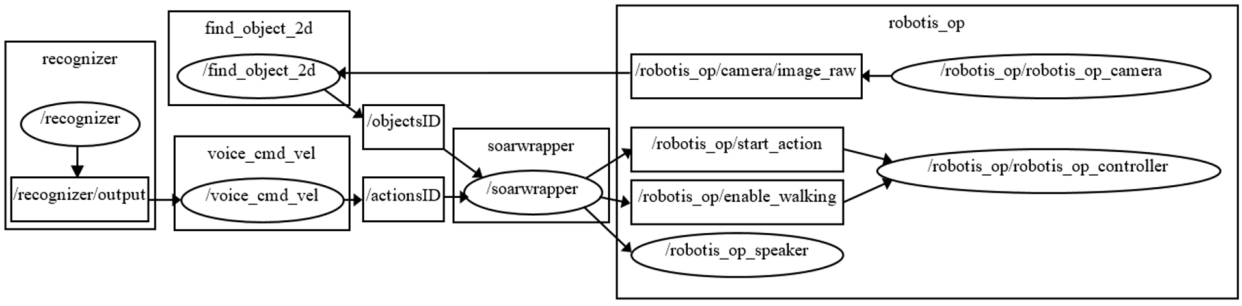


Fig. 9. Node structure of the developed HRI package using Soar-ROS hybrid platform

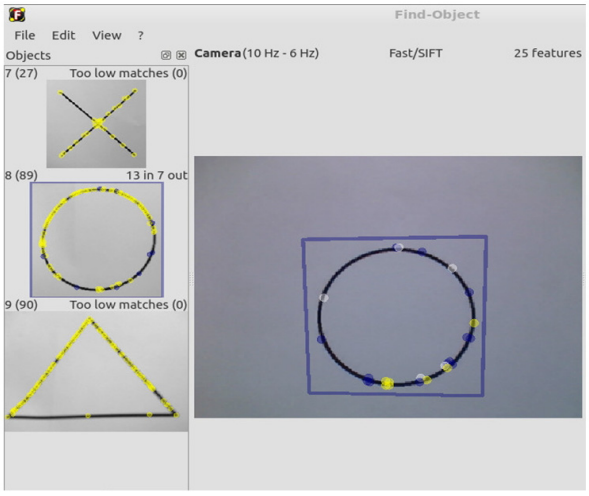


Fig. 10. Recognition of an 'O' mark using the find-object package

situation have been removed in the current environment. Then, the robot will behave normally following all the human's commands.

Fig. 9 shows the node structure of the developed package based on Soar-ROS hybrid platform implemented on OP2. Two well-known open-source packages are employed to carry out the HRI task. First, pocketsphinx is a speech recognizer shown as a voice package in Fig. 9, recognizing incoming vocal sounds [17]. Second, find-object package is a ROS integration for objects recognition. Using a webcam on the OP2 robot, objects can be detected and published on a ROS topic [18]. For instance, as shown in Fig. 10, a circle object, i.e., an O mark, is recognized because thirteen points are matched with it.

All the information from camera and voice packages

Table1. Available voice and image instructions

Instruction	Meaning
Stand	Stand up motion
Down	Sit down motion
Yes	Nod motion
No	Shaking head motion
Go	Walking forward motion
Stop	Stand still posture
X mark	Dangerous situation
O mark	Safe situation

are gathered into the soarwrapper node as shown in Fig. 9. The find_object_2d node receives raw images from OP2's camera, detects each shape, and publishes objects' identification numbers. In parallel, the voice_cmd_vel node recognizes human's utterances thanks to its recognizer. It then publishes command's identification numbers. These two kinds of identification numbers inform the situation where the robot is deployed.

Based on these identification numbers, the soarwrapper node decides which behavior the robot should perform as a suitable response to the command and situation. For example, if the robot gets a "Go" instruction but it has seen an X mark, it will not go forward for the safety of itself and human based on the scenario, denying the instruction by saying "No, I cannot do it." via the robotis_op_speaker node. There are six voice commands and two mark images available in the present HRI package as shown in Table 1.

Fig. 11 shows a part of the Soar code for the HRI

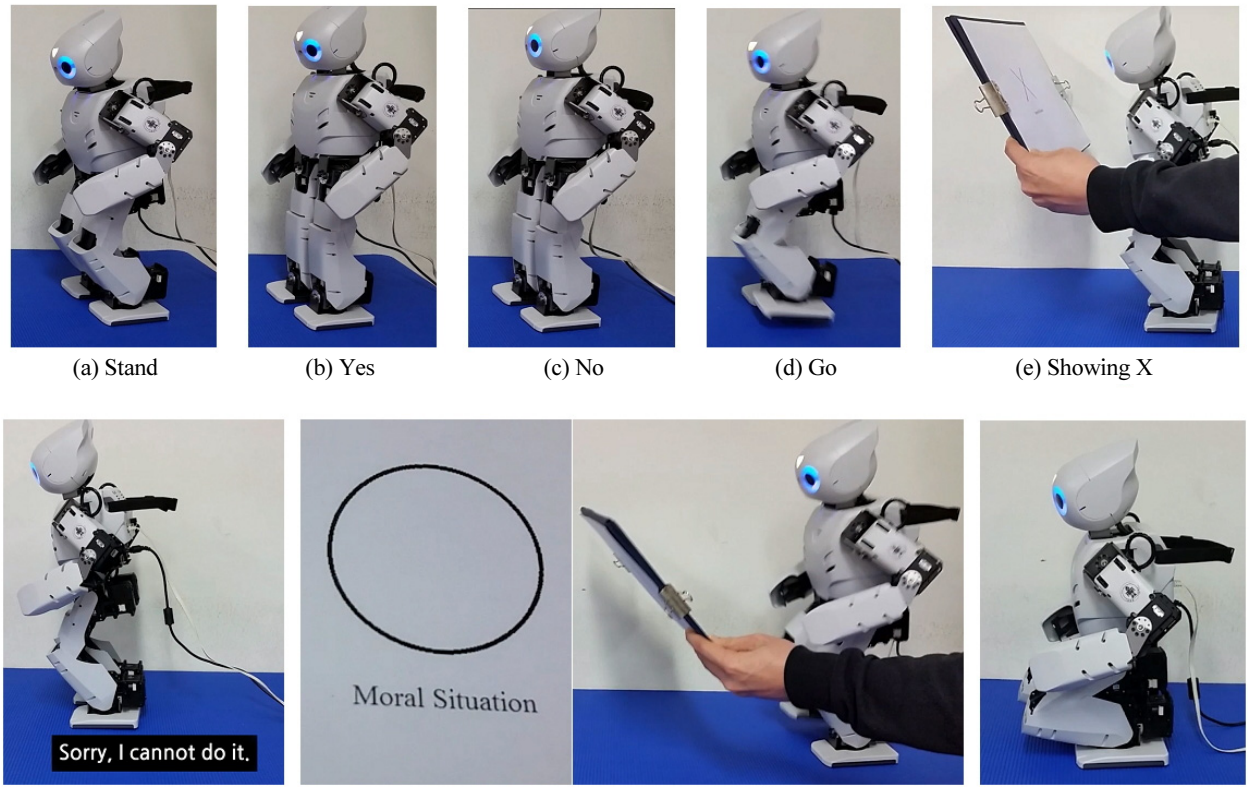
```
sp {soar-camera*propose*case_1
    (state <s> ^name soar-camera
               ^io-link-status retrieved
               ^io.output-link <i3>)
    (<i3> ^obj_shape << X Triangle >> )
-->
    (<s> ^operator <op> + =)
    (<op> ^name case_1
          ^walk False ^last_situ X ^sound 0)}

sp {soar-camera*apply*case_1
    (state <s> ^name soar-camera
               ^io-link-status retrieved
               ^operator <o>
               ^io.output-link <i3>)
    (<o> ^name case_1
         ^walk <v1> ^last_situ <v2> ^sound <v3>)
    (<i3> ^walk_cmd <v11>
          ^prev_situ <v22> ^sound <v33>)
-->
    (<i3> ^walk_cmd <v11> - <v1>
          ^prev_situ <v22> - <v2>
          ^sound <v33> - <v3>)
    (<s> ^io-link-status retrieved - changed)}

sp {soar-camera*propose*case_2
    (state <s> ^name soar-camera
               ^io-link-status retrieved
               ^io.output-link <i3>)
    (<i3> ^obj_shape no ^last_situ <v>)
-->
    (<s> ^operator <op> + =)
    (<op> ^name case_2 ^last_situ <v>)}

sp {soar-camera*apply*case_2
    (state <s> ^name soar-camera
               ^io-link-status retrieved
               ^operator <o>
               ^io.output-link <i3>)
    (<o> ^name case_2 ^last_situ <v1>)
    (<i3> ^prev_situ <v11>)
-->
    (<i3> ^prev_situ <v11> - <v1>)
    (<s> ^io-link-status retrieved - changed)}

sp {soar-camera*propose*case_3
    (state <s> ^name soar-camera
               ^io-link-status retrieved
               ^io.output-link <i3>)
    (<i3> ^obj_shape Circle)
-->
    (<s> ^operator <op> + =)
    (<op> ^name case_3
          ^walk True ^last_situ Circle ^sound 0)}
sp {soar-camera*apply*case_3
    (state <s> ^name soar-camera
               ^io-link-status retrieved
               ^operator <o> ^io.output-link <i3>)
    (<o> ^name case_3
         ^walk <v1> ^last_situ <v2> ^sound <v3>)
    (<i3> ^walk_cmd <v11>
          ^prev_situ <v22> ^sound <v33>)
-->
    (<i3> ^walk_cmd <v11> - <v1>
          ^prev_situ <v22> - <v2>
          ^sound <v33> - <v3>)
    (<s> ^io-link-status retrieved - changed)}
sp {soar-camera*propose*case_4
    (state <s> ^name soar-camera
               ^io-link-status retrieved
               ^io.output-link <i3>)
    (<i3> ^walk_voice True ^last_situ X)
-->
    (<s> ^operator <op> + =)
    (<op> ^name case_4
          ^walk False ^last_situ X ^sound 1)}
sp {soar-camera*apply*case_4
    (state <s> ^name soar-camera
               ^io-link-status retrieved
               ^operator <o> ^io.output-link <i3>)
    (<o> ^name case_4
         ^walk <v1> ^last_situ <v2> ^sound <v3>)
    (<i3> ^walk_cmd <v11>
          ^prev_situ <v22> ^sound <v33>)
-->
    (<i3> ^walk_cmd <v11> - <v1>
          ^prev_situ <v22> - <v2>
          ^sound <v33> - <v3>)
    (<s> ^io-link-status retrieved - changed)}
```

Fig. 11. Soar rules for the camera agent

task with four pairs of the proposed and the applied rules (four cases) in the production memory which is responsible for reasoning camera images. Each rule has two parts, “if” and “then” parts, separated by “-->”. Depending on different object shapes, robot will make corresponding interactive actions accordingly.

For case one, if camera detects the ‘X’ or ‘△’ object shape, operator case_1 is created. Then, this operator is applied and gives commands to output links which disable walking. Therefore, the robot will stop when it sees the ‘X’ or ‘△’ object.

For case two, if the camera detects no object, an operator case_2 is created, applying output command which saves the robot’s status. As a result, without objects



(a) Stand (b) Yes (c) No (d) Go (e) Showing X

(f) Go after showing X (g) Showing O (h) Down

Fig. 12. Experiment of the proposed HRI task (https://www.youtube.com/watch?v=8T65dsjr8dg&feature=youtu.be)

recognition the robot will keep walking or standing according to its previous command.

In the third case, if the camera detects a circular shape, the third rule is fired, creating the case_3 operator and applying walking decision in output links. This case causes the robot to walk forward whenever it sees an O shape which stands for a safe situation.

For the last case, if the previous object shape was 'X' or a triangle which stands for a dangerous situation and voice command enables robot to walk, the fourth rule is fired. This is a situation in which robot is commanded to walk forward while facing with dangerous warning. Then, the command that disables walking is applied to output link and sound command is turned on for robot to show its denial. These four pairs of rules are made in Soar production memory, giving robot knowledges to reason and make decisions while operating in human being's environment as described in the present HRI scenario.

Fig. 12 shows sequential snapshots captured from the experiment of conducting the proposed HRI task with OP2 using the Soar-ROS hybrid platform. In this experiment, OP2 follows well all the six vocal commands listed in Table 1 by default as shown in Fig. 12(a) to Fig. 12(d). However, after recognizing the symbol 'X' as shown in Fig. 12(e), it only refuses to respond the directive, "Go". This decision is reasonable in case the robot is standing at the edge of a table, and the 'X' symbol means this unsafe situation. For the robot not to be misunderstood as out of order, it says "Sorry, I cannot do it" through its speaker as shown in Fig. 12(f).

When one shows to the robot 'O' mark which means that the danger to the robot and/or human has been removed, OP2 resumes walking forward as shown in Fig.

12(g). Then OP2 obeys well all the commands including the finishing motion of sitting down as shown in Fig. 12(h). Even though orders of telling the six commands and showing the two marks were mixed arbitrarily, the robot responded correctly according to the given setup of moral situation based on Soar.

For the purpose of education and collaborative research in cognitive robotics engineering, source code of the present package is uploaded in the dedicated website (http://robofriend.kr/).

5. Conclusion

This study proposes integration of Soar and ROS, aiming to make robots perceive surrounding objects, reason based on predefined rules, and decide its action appropriately with the situation. The implementation for a simple HRI task is demonstrated successfully with a humanoid robot OP2.

As future work, the Soar-ROS hybrid platform will be applied to development of artificial moral agent, which will interact human both intelligently and gently with basic morality.

References

[1] K.P. Valavanis and G.N. Saridis, Intelligent robotic systems: theory, design and applications, Springer Science + Business Media, LLC, November, 2012.

[2] H.S. Park, K.C. Koh, H.-S. Kim, and H.-G. Lee, "State of R&D projects for intelligent robots," Journal of Korea Robotics Society, vol. 2, no. 2, pp. 191-195, June, 2007.

[3] R. Arkin, Governing lethal behavior: Embedding ethics in a hybrid deliberative/reactive robot architecture, Technical Report GIT-GUV-07-11, Georgia Institute of Technology.

[4] G. Briggs and M. Scheutz, "Sorry, I can't do that: Developing mechanisms to appropriately reject directives in human-robot interactions," AAAI Fall Symposium Series, 2015.

[5] J.G. Trafton, L.M. Hiatt, A.M. Harrison, F.P. Tamborello, II, S.S. Khemlani, and A.C. Schultz, "ACT-R/E: an embodied cognitive architecture for human-robot interaction", Journal of Human-Robot Interaction, vol. 2, no. 1, pp. 30-55, March, 2013.

[6] M. Beetz, M. Lorenz, and T. Moritz, "CRAM－a cognitive robot abstract machine for everyday manipulation in human environments", IEEE/RSJ International Conference on Intelligent Robots and Systems, Taipei, Taiwan, 2010, pp. 1012-1017.

[7] T.D. Kelley, "Developing a psychologically inspired cognitive architecture for robotic control: the symbolic and subsymbolic robotic intelligence control system (SS-RICS)", International Journal of Advanced Robotic Systems, vol. 3, no. 3, pp. 219-222, Sept., 2006.

[8] J.E. Laird, The Soar Cognitive Architecture, MIT Press, 2012.

[9] J. Puigbo, A. Pumarola, C. Angulo, and R. Téllez, "Using a cognitive architecture for general purpose service robots control," Connection Science, vol. 27, no. 2, pp. 105-117, April, 2015.

[10] M. Quigley, B. Gerkey, and W. D. Smart, Programming Robots with ROS, O'Reilly Media, 2015.

[11] J.E. Laird and C.B. Congdon, The Soar User's Manual Version 9.4.0, Computer Science and Engineering Department, University of Michigan, 2014.

[12] University of Michigan Soar Group, 'Soar Home,' Available: http://soar.eecs.umich.edu/. [Accessed: February 22, 2017]

[13] C.V. Dang, T.T. Tran, T.X. Pham, K.-J. Gil, Y.-B. Shin, and J.-W. Kim, "Connection of Soar and ROS for intelligent and robotic systems", International Conference on Engineering Mechanics and Automation, Hanoi, Vietnam, 2016, pp. 1-4.

[14] Github Inc., 'ROBOTIS-OP,' Available: https://github.com/ROBOTIS-OP. [Accessed: February 22, 2017]

[15] N.-Y. Choi, Y.-L. Choi, and J.-W. Kim, "Optimal joint trajectory generation for biped walking of humanoid robot based on reference ZMP trajectory", Journal of Korea Robotics Society, vol. 8, no. 2, pp. 92-103, June, 2013.

[16] J.-W. Kim, Enjoy together humanoid robot: ROBOTIS OP, HongRung Publishing Company, 2015.

[17] Github Inc., 'pocketsphinx,' Available: https://github.com/mikeferguson/pocketsphinx. [Accessed: February 22, 2017]

[18] Github Inc., 'Find-Object project,' Available: https://github.com/introlab/find-object. [Accessed: February 22, 2017]

Chien Van Dang

2014 M.S. in Electronic Engineering, Dong-A University, Busan, Korea

2014 ~ present Ph.D. in Electronic Engineering, Dong-A University, Busan, Korea

Research Interests: Humanoid robot, AI, Cognitive architecture, Embedded systems

Tin Trung Tran

2013 M.S. in Electronic Engineering, Dong-A University, Busan, Korea

2013 ~ present Ph.D. in Electronic Engineering, Dong-A University, Busan, Korea

Research Interests: Embedded systems, Power Electronics, Humanoid robot

Trung Xuan Pham

2003 B.S. in The University of Danang-University of Science and Technology, Danang, Vietnam

2015 ~ present M.S. in Electronic Engineering, Dong-A University, Busan, Korea

Research Interests: Embedded systems, Humanoid robot, Cognitive architecture

Ki-Jong Gil

2016 B.S. in Electronic Engineering, Dong-A University, Busan, Korea

2016 ~ present M.S. in Electronic Engineering, Dong-A University, Busan, Korea

Research Interests: Embedded systems, Humanoid robot

Yong-Bin Shin

2016 B.S. in Electronic Engineering, Dong-A University, Busan, Korea

2016 ~ present M.S. in Electronic Engineering, Dong-A University, Busan, Korea

Research Interests: Embedded systems, Humanoid robot

Jong-Wook Kim

1998 B.S in Department of Electronics and Electrical Engineering, POSTECH, Pohang, Korea

2000 M.S. in Electronics and Electrical Engineering, POSTECH, Pohang, Korea

2004 PhD. in Electronics and Electrical Engineering, POSTECH, Pohang, Korea

2006 ~ present Associate Professor in Department of Electronic Engineering, Dong-A University, Busan, Korea

Research Interests: Embedded system, Optimization algorithm, Humanoid robot, Cognitive architecture, Artificial Intelligent