Reinforcement Learning Framework for Robots in the Real World that Extends Cognitive Architecture: Prototype Simulation Environment "Re:ROS"

Sei Ueno[1*], Masahiko Osawa[2, 3], Michita Imai[2], Tsuneo Kato[4]
Hiroshi Yamakawa[5, 6]

[1]*Kyoto University of Informatics Kyoto, Japan*
[2] *Graduate School of Science and Technology, Keio University.*
[3] *Japan Society for Promotion of Science, Tokyo, Japan.*
[4] *Doshisha University Faculty of Science and Engineering.*
[5] *Dwango Artificial Intelligence Laboratory, Dwango ltd.*
[6] *The Whole Brain Architecture Initiative, a specified non-profit organization.*

Abstract
Reinforcement learning, which is a field of machine learning, is effective for behavior acquisition in robots. The extension of cognitive architecture with synchronous distributed systems is also effective for behavior acquisition. However, early work on the reinforcement learning software framework does not solve the difference between asynchrony and synchrony, and it cannot apply cognitive architecture with asynchronous distributed systems. Therefore, we applied asynchronous distributed systems to reinforcement learning modules to adapt to asynchrony and to extend cognitive architecture with asynchronous distributed systems. We prototyped a reinforcement learning software framework called "Re:ROS."

Keywords: Reinforcement learning, Cognitive architecture, Learning Environment, Robotics, Robot learning

1 Introduction

Reinforcement learning, which is a field of machine learning, is a problem, a class of solutions that work well on the problems, and the field in which learns these problems and their solutions are studied. Unlike supervised learning, reinforcement learning does not show a clear solution to find the correct action. The reinforcement learning algorithm needs to find proper actions based on the reward for reinforcement learning at each step. Reinforcement learning studies an action from a series of states and actions. An agent not only has to exploit what it already knows to obtain a reward, but also must

[*] This work was mainly performed at Doshisha University.

explore for making better action selections in the future. Reinforcement learning has a trial-and-error characteristic, and therefore, it can achieve the goal in the case of a very large state or action spaces, for which supervised learning cannot define supervisory signals. State or action spaces for the control of robots are very large. One early works. (Kimura, Yamashita, Kobayashi, 2002) reported that a four-legged robot successfully learned to walk in practical learning steps and another work (Yamada, Ohkura, Ueda, 2003) reported that arm robots acquire cooperative behavior through reinforcement learning. These reports suggest that reinforcement learning is effective for behavior acquisition in robots and that the reinforcement learning software framework is very important. In terms of the control of robots, it seems that reinforcement learning modules can be applied to cognitive architecture or the agents applying cognitive architecture to achieve the goal in a reinforcement learning problem. In particular, cognitive architecture with asynchronous distributed systems has been reported, such as subsumption architecture (Brooks, 1986), and adaptive control of thought—rational (ACT-R) (Anderson, 1996) and cortical capacity-constrained concurrent activation-based production system (4CAPS) (MA Just, 2007). For example, Genghis and Roomba, developed by iRobot, are robots using subsumption architecture. Thus, in the case of the control of robots in the real world, cognitive architecture with asynchronous distributed systems is also very important.

There exist a toolkit for reinforcement learning called Gym (Greg Brockman, 2016) and a toolkit called Gym-Gazebo (Iker Zamora, 2016) for reinforcement learning using Robot Operation System (ROS) (Morgan Quigley, 2009) and Gazebo (Koenig Nathan, 2006) that extends Gym for robotics. Gym provides environments in video games such as Pong and, Go and a toolkit for developing and comparing reinforcement learning algorithms in these video games.

There are two problems in terms of the environment when applying the control of robots in the real world using Gym. First, the environments provided by Gym are mostly 2D games. Gym provides few environments using 3D simulation and is not scalable to robots in the real world or to another environment. Second, Gym assumes synchrony, and changes of the environment, process of states, and reward signals occur simultaneously at each step. Gym designs a toolkit based on asynchrony. However, in the real world, changes of the environment, states, and reward signal do not occur strictly at the same time, as the real world has asynchrony. Gym-Gazebo is scalable to a 3D simulator and robots in the real world. However, Gym-Gazebo does not solve the difference between synchrony and asynchrony owing to the use of Gym and it cannot apply cognitive architecture with asynchronous distributed systems.

Therefore, in this work, reinforcement learning modules apply asynchronous distributed systems to adapt to asynchrony such as in the real world. We propose a reinforcement learning software framework to control robots in the real world through extended cognitive architecture with asynchronous distributed systems, named "Re:ROS."

To apply asynchronous distributed systems, Re:ROS uses ROS. ROS has nodes that are functions for robots or publishing/subscribing orders. Nodes are distributed and asynchronous allowing distributed operation over multiple cores, multiple processors, GPUs and clusters. Re:ROS applies nodes to the reinforcement learning state, reward signal, changes of the environment, and the action of the agent. Each node publishes and subscribes to information on reinforcement learning needs. Because asynchronous distributed systems are applied, Re:ROS is scalable to cognitive architecture with asynchronous distributed systems.

ROS has many libraries of robots and sensors and a physical simulator called Gazebo. Using ROS and Gazebo, Re:ROS can use existing robots to simulate an asynchronous world similar to the real world.

The reminder of the paper is organized as follows. Section 2 describes the elements of reinforcement learning that should be asynchronous. Section 3 describes the elements of Re:ROS. Section 4 describes an example involving an agent called Turtlebot with subsumption architecture and a problem of dribbling a soccer ball and shooting it to a goal. Finally, Section 5 presents the conclusion.

2 Elements of Reinforcement Learning

The most important aspects of reinforcement learning are the agent and environment. A reinforcement learning agent interacts with its environment. Beyond the agent and the environment, there are four main subelements of reinforcement learning: a policy, a reward signal, a value function, and an optical model of the environment (Richard S. Sutton, 1998).

A policy defines the learning agent's behavior at a given time. It is the core of an agent in the sense that it alone is sufficient to determine behavior.

A reword signal defines the goal in a reinforcement learning problem. At each step, the environment sends a reward signal to the reinforcement learning agent a reward signal.

A value function specifies what is desirable in the long run. Values indicate the long-term desirability of states after taking into account the states that are likely to follow.

A model of the environment entails information of behavior of the environment that allows inferences to be made on how the environment will behave. Methods for solving reinforcement learning problems that use models and planning are called *model-based* methods, as opposed to simpler *model-free* methods that are explicitly trial-and-error learners. In this work, Re:ROS can use both methods.

2.1 Asynchronous Distributed Systems

We describe functions that should be in the asynchronous distributed systems. The agent and environment have sub-functions. The functions are as a follows.

Agent

- Actions (policies);
- Value functions.

Environment

- Changes of the environment;
- Reward signals;
- States;
- (Optical models of the environment).

Each function receives information for operating its own function and sends the result of calculation.

In this work, we define the agent as robots. The agent performs reinforcement learning actions, rule-based actions, and random actions for exploration. The actions are not performed simultaneously; rather they are asynchronous operations. For extending cognitive architecture, we separated a function to determine the action from a function to control an agent.

2.2 Cognitive Architecture with Asynchronous Distributed Systems

The functions of Re:ROS depend on cognitive architecture. For example, Re:ROS using subsumption architecture has a function for decision-making. This function needs information on the lowest-layer action, second-layer action information, and so on till the highest-layer action.

In Figure1, Action1, Action2 and Action3 in Agent distributedly send information to subsumption architecture. Subsumption architecture chooses which actions the agent performs. State, reward signal, and others distributedly receive information on the action from subsumption architecture, and each function sends information to functions that need it.

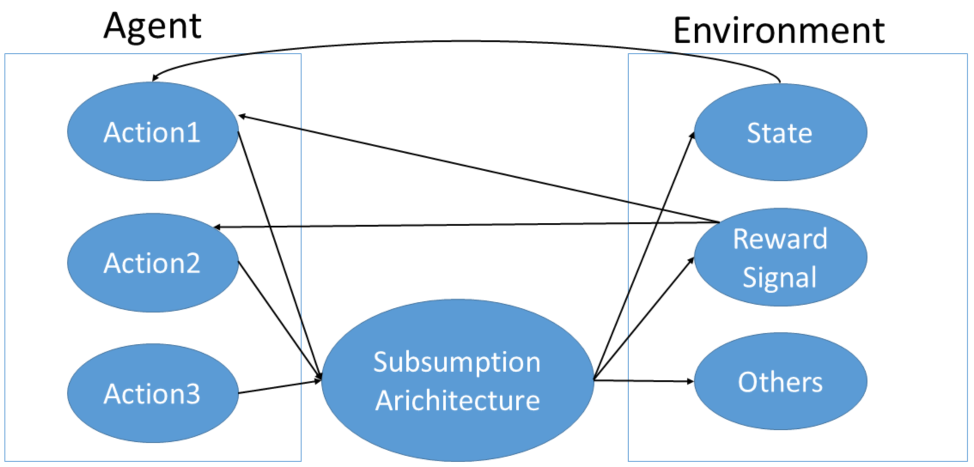

Figure 1: Example of Re:ROS elements

3 Elements of Re:ROS

Re:ROS has ROS and Gazebo to make asynchronous distributed systems. Gazebo is a physical simulator made by the Open Source Robotics Foundation (OSRF). We used it as an environment of reinforcement learning. Indeed, training in the real world is ideal. However, it is difficult to adapt early reinforcement learning such a trial-and-error to the real world because of problems that may be caused by the robot’s random actions, danger to humans, and cost of replacing. Therefore we use a simulator that is similar to the real world for the training.

3.1 Re:ROS Packages

Re:ROS performs reinforcement learning with four packages made by ROS.

- re_ros;
- re_rule;
- re_agent;
- re_environment.

“re_ros” determines the training environment’s settings: which environments to use, which agents to learn, and which cognitive architecture to select. After determining the settings, “re_ros” starts them.

“re_rule” defines cognitive architecture. Its main role is to select the action to send to the environment.

“re_agent” defines an agent for reinforcement learning. It includes robot management, action of reinforcement learning, and policy.

“re_environment” defines the environment and the reinforcement learning problem. This package includes reward signals, state and models of the environment.

“re_ros” starts another three packages. Each package continuously publishes information for learning. Moreover, it subscribes to information from other packages to operate its own programs.

4 Experiment

We performed an experiment with the proposed system. In this experiment (Osawa, Ashihara, Shimada, Kurihara, Imai, 2016), we used Turlebot as the agent to dribble a soccer ball and shoot it to a goal. It applies subsumption architecture as cognitive architecture. The agent has five actions: Forward/Backward (v=1 m/s), Left/Right (w = +-2 rad/s), and stop. The state is detected using Kinect (60 × 60 depth image); Figure 3. The policies are Deep Q-Network (DQN) (Volodymyr Mnih, 2013), random walk and suppressor. Reward signals are as follows.

- Ball near the goal: 0 ~ 1.0;
- Goal: 1.0;
- No goal in ten seconds: -1.0.

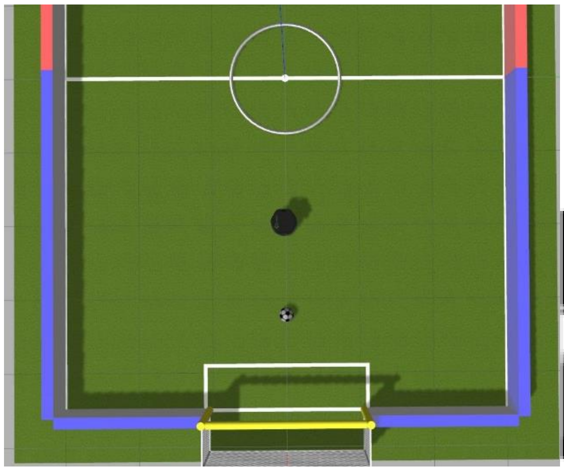
Figure 2: Environment: a soccer field

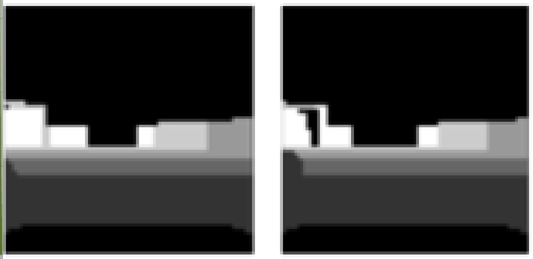
Figure 3: State: 60 × 60 depth image

Figure 4 shows nodes in Re:ROS. The “/accumulator” node subscribes to the messages from “/agent1/random_walk,” “/agent1/suppressor,” and “/agent1/dqn_walk.” The “/accumulator” publishes “/gazebo” as the environment. “/agent1/dqn_walk” needs state and a reward signal, and subscribes to information from “/reward_soccer” and “/gazebo.”

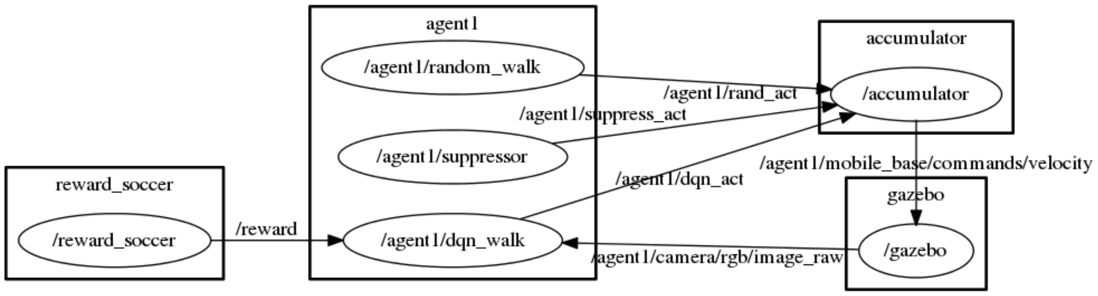

Figure 4: Nodes in Re:ROS using rqt_graph, which is a ROS package.

5 Conclusion

This work proposed Re:ROS, an reinforcement learning software framework extending cognitive architecture with asynchronous distributed systems.

In the future we plan to extend this work to increase the number of robots and environments for use as templates and to compare the real and simulated world.

References

Anderson,J. R. (1996). ACT: A simple theory of complex cognition. American Psychologist, 51(4), 355.

Brooks, R. (1986). A robust layered control system for a mobile robot. IEEE Journal on Robotics and Automation, 2(1), 14-23.

Brockman, G., Cheung,V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). OpenAI Gym. arXiv preprint arXiv:1606.01540.

Zamora, I., Lopez, N. G., Vilches, V. M., & Cordero, A. H. (2016). Extending the OpenAI Gym for robotics: a toolkit for reinforcement learning using ROS and Gazebo. arXiv preprint arXiv:1608.05742.

Kimura, H., Yamashita, T., & Kobayashi, S. (2002). Reinforcement learning of walking behavior for a four-legged robot. IEEJ Transactions on Electronics, Information and Systems,122(3), 330-337.

Koenig, N., & Howard, A. (2006). Gazebo-3d multiple robot simulator with dynamics.

Just, M. A., & Varma, S. (2007). The organization of thinking: What functional brain imaging reveals about the neuroarchitecture of complex cognition. Cognitive, Affective, & Behavioral Neuroscience, 7(3), 153-191.

Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Berger, E., Wheeler, R., & Ng, A. Y. (2009). ROS: an open-source Robot Operating System. In ICRA workshop on open source software (Vol. 3, No. 3.2, p. 5).

Osawa, M., Ashihara, Y., Shimada, D., Kurihara, S., & Imai, M. (2016). Arbitration of multiple learner and application of cognitive architecture using accumulator utilizing prefrontal area. SIG-AGI,4th.

Richard, S., Sutton, G., & Barto, A. (1998). Reinforcement learning: An introduction (Vol. 1, No. 1). Cambridge: MIT press.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602.

Yamada, K., Ohkura, K., & Ueda, K. (2003). Cooperative behavior acquisition of autonomous arm robots through reinforcement learning. Transactions of the Society of Instrument and Control Engineers,39.3,266-275.