# Transfer Learning Method Using Ontology for Heterogeneous Multi-agent Reinforcement Learning

Hitoshi Kono
Graduate School of Advanced Science and Technology
Tokyo Denki University
Tokyo, Japan

Akiya Kamimura and Kohji Tomita
Intelligent Systems Research Institute
National Institute of Advanced Industrial Science and Technology (AIST)
Ibaraki, Japan

Yuta Murata Graduate School of Engineering Tokyo Denki University
Tokyo, Japan

Tsuyoshi Suzuki
Department of Information and communication Engineering
Tokyo Denki University
Tokyo, Japan

***Abstract*—This paper presents a framework, called the knowledge co-creation framework (KCF), for heterogeneous multi-agent robot systems that use a transfer learning method. A multi-agent robot system (MARS) that utilizes reinforcement learning and a transfer learning method has recently been studied in real-world situations. In MARS, autonomous agents obtain behavior autonomously through multi-agent reinforcement learning and the transfer learning method enables the reuse of the knowledge of other robots' behavior, such as for cooperative behavior. Those methods, however, have not been fully and systematically discussed. To address this, KCF leverages the transfer learning method and cloud-computing resources. In prior research, we developed ontology-based inter-task mapping as a core technology for hierarchical transfer learning (HTL) method and investigated its effectiveness in a dynamic multi-agent environment. The HTL method hierarchically abstracts obtained knowledge by ontological methods. Here, we evaluate the effectiveness of HTL with a basic experimental setup that considers two types of ontology: action and state.**

***Keywords*—Transfer learning; Multi-agent reinforcement learning; Multi-agent robot systems**

## I. Introduction

Actual multi-agent robot systems (MARSs) have recently been deployed in real-world situations. Among other applications, a multi-robot inspection systems for disaster-stricken areas, autonomous multi-robot security systems, and autonomous multi-robot conveyance systems for warehouses have been developed [1]–[3]. However, the real world, where such MARSs are expected to operate, is a dynamic environment that complicates the development of the systems because developers must customize the robots to this dynamic environment. The application of multi-agent reinforcement learning (MARL) to MARSs is one of the approaches taken in response to this problem. MARL is a mechanism for implementing a posteriori cooperation among agents, which can behave adaptively in a dynamic environment even when they are not provided with specific control policies. The benefits of MARL have been demonstrated in various studies over the past decade [4]–[6]. The application of MARL to actual robots has been studied by Matarić [7]. A method for accelerating the learning process has also been investigated because reinforcement learning in dynamic environments requires a long time to obtain an optimal (or nearly optimal) solution [6]. However, this method is difficult to apply to MARS with MARL in dynamic environments because the learning speed is impractically low. Moreover, a MARS typically contains at least one pre-programmed robot, and MARL has the following drawbacks.

- The learning process requires a long time.
- The obtained knowledge depends on the situation.
- There is a limit to a robot's capacity to store the knowledge.

In contrast, cloud robotics has recently been proposed [8], [9] as a means to increase the availability of standalone robots by utilizing cloud computing resources. Cloud robotics may increase the utility of MARSs because the robots gain access to broader knowledge, vast computing resources, and external functions. This should be helpful for achieving practical implementation of MARSs with MARL.

In this context, we propose a *knowledge co-creation framework* (KCF) by integrating MARS, MARL, and cloud robotics [10], [11]. To implement this framework, an autonomous mobile robot in a MARS internally executes cyclical processes, and we implement cloud services for gathering and assimilating knowledge (Fig. 1) as follows.

- Knowledge data are generated by using computer simulation and other MARL systems.
- A robot saves knowledge to its own repository via a network connected to cloud computing resources.
- The robot observes the environmental state.
- The robot selects particular knowledge from the repository on the basis of the observed environment.

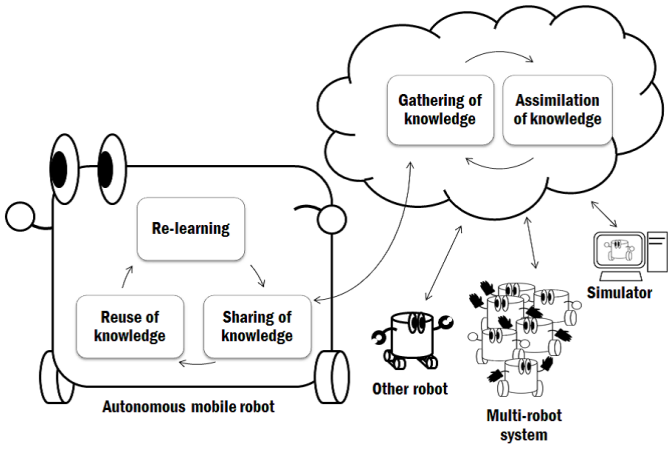



Fig. 1: Simplified representation of a KCF. All systems (including the other robots, MARS and simulator) are connected to cloud-computing resources.

- If the observed environment is unknown, the robot acquires the learned knowledge of other robots (reuse of knowledge) [12].
- As a result of this action, the robot obtains new knowledge about unknown environments and shares new knowledge with other robots and systems.

Note that an autonomous agent acts on the basis of existing knowledge if the observed environment is known.

We developed the hierarchical transfer learning (HTL) method as the core technology of KCF. The HTL method enables inter-task mapping (ITM) by using ontology among heterogeneous agents. This allows autonomous robots and virtual agents to reuse knowledge from other types of robots and agents. Here, we describe experiments that confirm the HTL enables reuse of knowledge by using action and state ontologies to mediate among heterogeneous MARSs.

The rest of the paper is organized as follows. Section 2 describes the theory and assumptions of reinforcement learning and transfer learning. Section 3 is an overview of the proposed HTL. Section 4 provides details about the preconditions of simulation experiments. Section 5 details evaluation of the effectiveness of HTL through simulation and contains a discussion of the results, which suggest that autonomous learning agents can reuse knowledge from other heterogeneous agents by using HTL. Section 6 contains concluding remarks.

## II. Reinforcement Learning and Transfer of Knowledge

### A. Reinforcement Learning

Reinforcement learning is one type of machine learning method, in which agents can use a trial-and-error method to create a policy for accomplishing tasks. Many kinds of reinforcement learning mechanisms have been proposed over the past few decades. In this study, we adopt Q-learning, defined below, as the reinforcement learning mechanism:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\{r + \gamma V(s') - Q(s,a)\} \quad (1)$$

$$V(s) = \max_{a \in A} Q(s,a) \quad (2)$$

Here, $S$ is a state space, with $s, s' \in S$; $a$ an element of an action space $A$; $\alpha(0 < \alpha \leq 1)$ is the learning rate; $\gamma(0 < \gamma \leq 1)$ is the discount rate; and $r$ is the reward. The learning agents select each defined $a$ with a probability given by the Boltzmann distribution according to

$$p(a|s) = \frac{\exp\left(\frac{Q(s,a)}{T}\right)}{\sum_{b \in A} \exp\left(\frac{Q(s,b)}{T}\right)}. \quad (3)$$

Here, $T$ is a parameter that determines the randomness of selection. The Q-learning model can select actions in descending order according to the action value from learned knowledge. When the values of available actions are the same or are equal to default value, the Boltzmann distribution is used to select the action at random.

### B. Transfer Learning in Reinforcement Learning

Transfer learning, as proposed by Taylor, is a framework for reuse of a policy obtained through reinforcement learning [12]. The policies and solutions obtained through reinforcement learning are here regarded as knowledge. In the transfer learning method, an agent first learns the policy as an action–state pair during the source task. Next, an agent performing the target task can reuse the knowledge obtained during the source task via ITM. ITM defines the relation of the spaces $S$ and $A$ between the target and source tasks. If the target task agent has state space $S_{target}$ and action space $A_{target}$, then ITM for simple tasks will map $S$ and $A$ between the target and source tasks. This is formulated as follows:

$$\begin{aligned} \chi_S(s): &\quad S_{target} \rightarrow S_{source} \\ \chi_A(a): &\quad A_{target} \rightarrow A_{source} \end{aligned} \quad (4)$$

Here, $s$ and $a$ are the elements of the state space and action space, respectively; $\chi_S(s)$ and $\chi_A(a)$ are the corresponding functions of ITM. The agent completing the target task can have different characteristics from the agent that learned the source task. Hence, the agent performing the target task can adapt its behavior for a new environment or target task. This method is fundamental in a single-agent environment.

### C. Transfer Learning in a Multi-agent Domain

In recent years, transfer learning has been investigated not only for single–agent systems but also for MARSs. For example, Boutsioukis et al. proposed a transfer learning method with multi–agent reinforcement learning, which enables the use of ITM among agents [13]. Taylor et al. proposed a parallel transfer learning method, which runs the target and source tasks simultaneously [14]. Their method speeds up learning in multi–agent transfer learning. However, many such methods do not take into account the operation of large numbers of single–agent systems and MARSs, which means

that an inter–task map must be either created or modified with every entry of a new agent system. The quality of ITM is the most important factor in agent performance on target tasks. Therefore, we believe that ITM for a system should be designed by humans (such as researchers and engineers) on the basis of experience and intuition. However, as already mentioned, manually designing an ITM system is problematic when large numbers of single-agent systems and MARSs are involved in the transfer learning system.

## III. Hierarchical Transfer Learning

### A. Heterogeneity of Robots and Agents

For actual environments, it is assumed that the heterogeneity of robots implies that they may have different sensors (e.g., camera and laser range finder) and actuator arrangements (e.g. crawler platform, omni-directional mobile platform, and humanoid platform). Moreover, different versions of robot types and differences in manufacturing are also aspects of heterogeneity. In contrast, characteristics of virtual agents are similar to other agents in the virtual environment, such as for simple task agents. For the purpose of evaluation in this paper, we assume a simulated environment. Hence, heterogeneity is characterized by the number of elements of $S$ and $A$. We suppose that the heterogeneity of $S$ arises from differences in tasks, and the heterogeneity of $A$ arises from differences in the motion characteristics of agents.

### B. Ontology-based ITMs

Our KCF with HTL enables integration of ITMs among agents [10]. In a previous paper, we proposed HTL, which uses the concept of ontologies as a method for creating ITMs. We call this technique ontology-based ITM (OITM). Ontology is introduced here as an "explicit specification of a conceptualization" for the purpose of learning [15]. Our OITM leverages the function of ontology by which we can describe many different relations in terms of ontology, and specifically we can describe integrative ITMs among agents (Fig. 2). Moreover, if we first define the ITM of a system in terms of ontology, then agents can use ITM to search the knowledge of many other agents. We assume that a concrete action of an agent is called an instance of ontology and an abstract action of ontology is called a *class* or *upper class*. We additionally specify that any ontology presentation in cloud resources can be accessed by all agents.

An example of OITM is shown in Fig. 3. First, the agent developer maps concrete actions of an agent to the ontology. Another agent developer also maps actions to this ontology. When the agent reuses the knowledge of other agents, it searches for a mapping that matches its actions with other agent actions. Second, the agent transfers knowledge from other agents to itself using the knowledge and mapping of ontology for ITM. Note that when the agent transfers the knowledge from other agents, OITM requires two ontologies, such as an action ontology and a state ontology. Hence, the agent individually searches corresponding actions and states of other agents.

Fig.3 shows a case where three heterogeneous agents are present in an environment. The action spaces of these three agents are as follows:

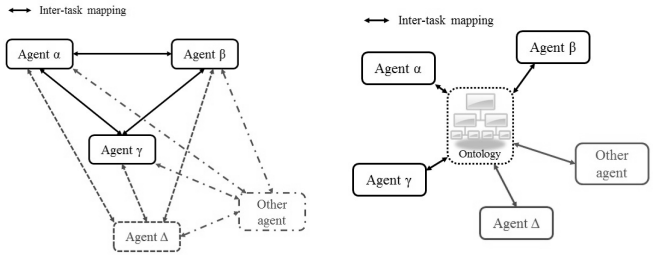



Fig. 2: Difference between ITM and OITM. (a) Simplified image of ITM with four agents and others. (b) Simplified image of OITM, which integrates ITM among agents.

$$\begin{aligned} A_\alpha &= \{a_{\alpha 1}, a_{\alpha 2}, a_{\alpha 3}, a_{\alpha 4}\} \\ A_\beta &= \{a_{\beta 1}, a_{\beta 2}, a_{\beta 3}\} \\ A_\gamma &= \{a_{\gamma 1}, a_{\gamma 2}, a_{\gamma 3}, a_{\gamma 4}, a_{\gamma 5}\} \end{aligned} \tag{5}$$

We connected each instance (concrete action) to the class $C_3^A = \{c^a_{3,1}, c^a_{3,2}, c^a_{3,3}, c^a_{3,4}\}$. The class space $C_3^A$ is also mapped to an upper class $C_2^A = \{c^a_{2,1}, c^a_{2,2}, c^a_{2,3}\}$, and $C_1^A = \{c^a_{1,1}\}$. These mapping describes functions like ITM, defined below, as mappings between instances and classes, and between classes and upper classes.

$$\begin{aligned} \chi^O_S(s): &\quad S \to C^S_h \\ \chi^O_A(a): &\quad A \to C^A_h \end{aligned} \tag{6}$$

$$\begin{aligned} \chi^O_S(c^s): &\quad C^S_h \to C^S_{h-1} \\ \chi^O_A(c^a): &\quad C^A_h \to C^A_{h-1} \end{aligned} \tag{7}$$

Here, we defined two types of OITM, namely, $\chi^O_S(\cdot)$ and $\chi^O_A(\cdot)$. The function $\chi^O_S(\cdot)$ represents an OITM about the state space among instances, classes, and upper classes. $\chi^O_A(\cdot)$ is an OITM about an action space. In the implementation, the agents have mechanisms to search the OITM.

### C. Method for Transfer of Knowledge

As mentioned above, the agent can reuse knowledge of other agents through HTL. In this study, we adopted Q-learning as the reinforcement learning model. In the Q-learning mechanisms, transferred knowledge is reused as follows.

$$Q^j(s,a) = (1-\tau)Q^t(s,a) + \tau Q^s(\chi^o_S(s), \chi^o_A(a)) \tag{8}$$

Here, $Q^t(s,a)$ is knowledge about the target task and $Q^s(s,a)$ is knowledge about a source task, known via HTL. The transferred knowledge also uses OITM and the functions $\chi^O_S(\cdot)$ and $\chi^O_A(\cdot)$ means OITM. The term $Q^j(s,a)$ is the combined knowledge of the target and source tasks, and $\tau (0 < \tau < 1)$ is a parameter for adjusting the action's value for the difference between the target and source task. A target task agent selects an action from $Q^j(s,a)$ according to a Boltzmann

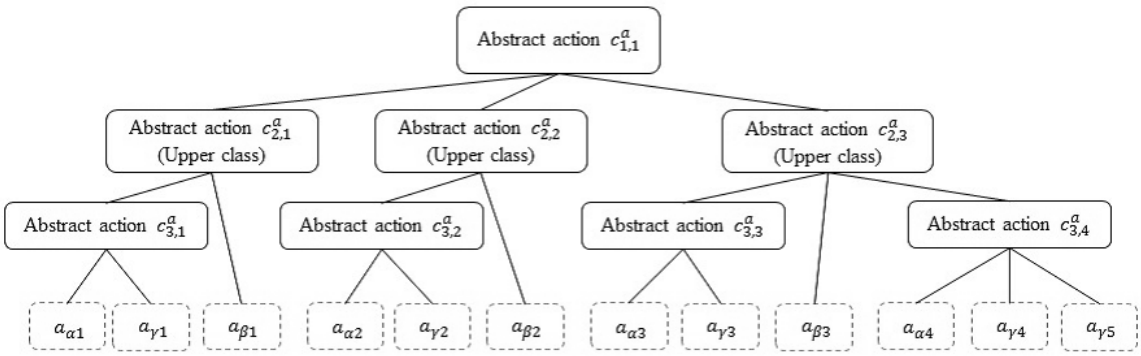



Fig. 3: OITM for agent actions. The agent's developer maps concrete actions of an agent to abstract actions in upper classes, which may be mapped into still higher classes. All actions of all agents are mapped to an ontology in this manner.

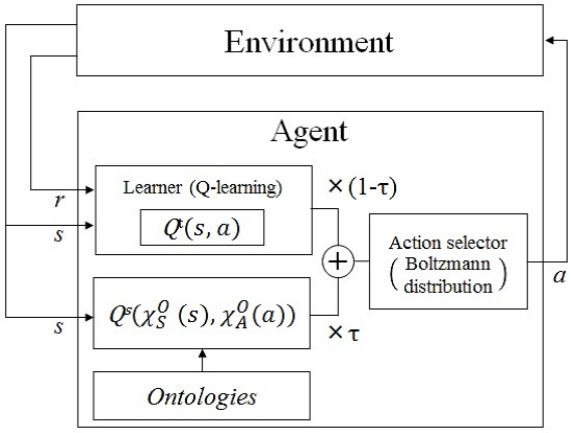



Fig. 4: Simplified schematic of an internal reinforcement learning model in a target task with Eq. (8). A learner can receive state and rewards from the environment. The source of transferred knowledge cannot receive the reward. The action is selected by using combined knowledge according to Eq. (8).

distribution (Equation (3)). However, updating of knowledge occurs only for $Q^t(s,a)$ by Q-learning (Fig.4). In an actual environment, when an actual agent, such as a robot, reuses transferred knowledge, the knowledge of source tasks consists of a data file generated by the source task agent, and the target task agent must receive the transferred knowledge (in the form of these files) about the source task via the network infrastructure. Hence, to reuse knowledge, HTL requires a communication infrastructure, as well as a list of available repositories of knowledge and public ontology servers.

## IV. Task Description

We carried out simulation experiments to confirm the effectiveness of HTL in four dynamic environments. We designed environments for MARL and heterogeneous experiments. We provide the following experimental conditions of computer simulation.

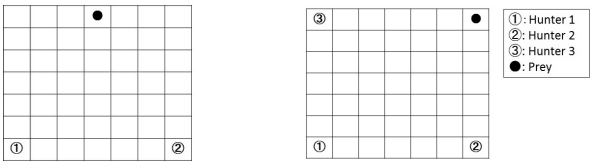
### A. Pursuit Game

Previous studies have adopted tasks such as zero-sum games, foraging tasks, and cooperative carrying tasks for evaluating MARL. Here, we adopt a pursuit game to evaluate MARL performance. The pursuit game is a benchmark test of agent performance, measured as time until capture. We set an $N \times N$ grid as the simulation world. An arbitrary number of hunter agents and prey agents are deployed in this world, and we evaluate the number of steps (i.e., time) until the hunters capture all of the prey. In our pursuit game, we set locations for prey in the grid world. The final state of this game occurs when all prey has been captured by hunters, which occurs when all hunters are adjacent to the prey at the end of turn. The locations of all agents are reset to their initial positions after capture. A single episode is defined as number of steps to reach a state of capture. Agents act in a predefined order, such as hunter 1 $\rightarrow$ hunter 2 $\rightarrow$ prey, and one set of actions is regarded as a single step. A cell cannot be simultaneously occupied by multiple agents, and agents cannot cross the world boundaries. Moreover, hunters can learn cooperative capture actions, but prey cannot learn.

### B. Difference in Tasks

The heterogeneity of the state space depends on the *Task* and *Sensor* characteristics in actual learning. In this experiment, we defined heterogeneity of the state space as the difference in *Tasks*.

We define the grid world of a pursuit game according to a study by Tan [4] and Arai et al. [5]. In this particular implementation, hunters and a prey agent can move in a 7 $\times$ 7 grid world. The initial position of each agent is shown in Fig.5. The difference between tasks is the number of hunters. We call the task in Fig. 5 (a) "2 vs. 1" and that in 5 (b) "3 vs. 1". Note that in the 2 vs. 1 task, the observable environmental state of a hunter is the set containing the coordinates of the other hunter and of the prey. In the 3 vs. 1 task, the observable environmental state is the set containing the coordinates of the other two hunters and of the prey. Therefore, the concrete difference between tasks is the observable number $s$ of the set of $S$. In each task, the observable environmental state as a set $S$ is defined as follows.




(a) (b)

Fig. 5: Difference in tasks. (a) Two hunters vs. one prey in $7 \times 7$ grid world, with initial positions of each agent. (b) Three hunters vs. one prey in $7 \times 7$ grid world with initial positions of agents in the four corners.

$$\begin{aligned} S_{2vs.1} = \{ & \ x\text{-}coordinate\ of\ self, \\ & \ y\text{-}coordinate\ of\ self, \\ & \ x\text{-}coordinate\ of\ the\ second\ hunter, \\ & \ y\text{-}coordinate\ of\ the\ second\ hunter \\ & \ x\text{-}coordinate\ of\ prey, \\ & \ y\text{-}coordinate\ of\ prey\} \end{aligned} \quad (9)$$

$$\begin{aligned} S_{3vs.1} = \{ & \ x\text{-}coordinate\ of\ self, \\ & \ y\text{-}coordinate\ of\ self, \\ & \ x\text{-}coordinate\ of\ a\ second\ hunter, \\ & \ y\text{-}coordinate\ of\ a\ second\ hunter, \\ & \ x\text{-}coordinate\ of\ a\ third\ hunter, \\ & \ y\text{-}coordinate\ of\ a\ third\ hunter, \\ & \ x\text{-}coordinate\ of\ prey, \\ & \ y\text{-}coordinate\ of\ prey\} \end{aligned} \quad (10)$$

### C. Heterogeneity of Agents

As mentioned above, the game involves two types of agents: multiple hunter agents and one prey agent. Only hunters are provided with learning mechanisms; the actions of the prey are provided by a fixed strategy, as discussed in detail below.

Agents can select only one action per step. Prey can choose an action from five actions in an action space $A_{prey}$, which is defined as follows.

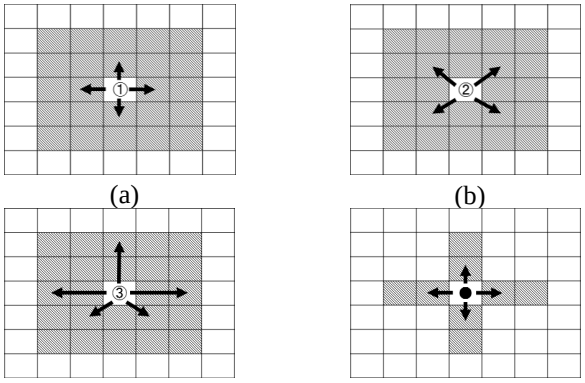
$$A_{prey} = \{front, back, right, left, stop\} \quad (11)$$

Heterogeneity of hunters means that differences are permitted between the strategies and action spaces of different hunters. In addition, each agent is provided with a sensor, such as sight. We define the allowed actions of each hunter in the following way.

$$A_{hunter1} = \{front, back, right, left, stop\} \quad (12)$$

$$A_{hunter2} = \{upper\ right, lower\ right, lower\ left, upper\ left, stop\} \quad (13)$$

$$A_{hunter3} = \{long\ front, long\ right, lower\ right, lower\ left, long\ left, stop\} \quad (14)$$



(a) (b) (c) (d)

Fig. 6: Actions and sight range of each agent. Arrows denote movable direction and distance in grid world. Gray areas show the sight range of each agent, and if other agents are in sight range, agent can observe the coordinates of other agents.

Here, characteristics of $A_{hunter1}$, $A_{hunter2}$, $A_{hunter3}$ and $A_{prey}$ are shown in Fig.6 subfigures (a), (b), (c), and (d), respectively. Each agent has its own sight range (shown as shaded cells), and the shape of this range differs among agents. The sight range of the prey is the same as that shown in Fig. 6 (c). Initially, hunters and prey choose their actions randomly. Hunters adjust the probabilities with which actions are selected as the learning progresses. Although the prey does not learn, it selects an escape action when it recognizes a hunter. The prey moves away from the hunter when it detects only one hunter, or in any of the possible escape directions (uniformly chosen) when it detects multiple hunters in its vicinity.

### D. Experimental Conditions

To confirm the effectiveness of HTL, we set the experimental conditions as listed in Table I. In this experiment, we adopted the 2 vs. 1 task and the 3 vs. 1 task of the pursuit game. In the source task and self-transfer experiment, hunters 1 and 2 and the prey are deployed in a $7 \times 7$ grid world. In the 3 vs.1 task, two hunter 1s and one hunter 2, or one each of hunters 1, 2, and 3 are deployed with the prey in the grid world. Moreover, on top of the above experimental conditions, we test the self-transfer condition. Self-transfer is used as confirmation of transferred knowledge properly generated by the agent of the source task, and we transfer the generated knowledge from the source task to the source task agent.

The Q-learning parameters are set to $\alpha = 0.1$, $\gamma = 0.99$, and $r = 1$. The Boltzmann parameter $T$ is 0.01. These parameters are common to the self-transfer condition. The default Q-value is 0 in all experiments, and $\tau$ is 0.5. In each experiment, 10000 episodes are conducted for the source and target tasks.

In this experiment, we designed two ontologies: action ontology in Fig.7 and state ontology in Fig.8. For example, when the hunter 3 reuses the knowledge of hunter 1 by using action ontology and state ontology, the information of observed states is put in the state ontology. The hunter 3 can translate its own observed states to an observable state of hunter 1, and translated states are input to the knowledge that was

TABLE I: Experimental conditions of transfer in four experiments. The target task agent uses transferred knowledge from a source task agent of the same type.

| Experiment | Conditions | Source task | | Target task |
|---|---|---|---|---|
| Self-transfer | Task | 2 vs. 1 | | 2 vs. 1 |
| | Hunters | Agent 1 and Agent 2 | | Agent 1 and Agent 2 |
| | Direction of transfer | Agent 1 | → | Agent 1 |
| | | Agent 2 | → | Agent 2 |
| Different action space | Task | 2 vs. 1 | | 2 vs. 1 |
| | Hunters | Agent 1 and Agent 2 | | Agent 2 and Agent 3 |
| | Direction of transfer | Agent 1 | → | Agent 2 |
| | | Agent 2 | → | Agent 3 |
| Different state space | Task | 2 vs. 1 | | 3 vs. 1 |
| | Hunters | Agent 1 and Agent 2 | | Two agent 1 and one Agent 2 |
| | Direction of transfer | Agent 1 | → | Agent 1 |
| | | Agent 2 | → | Agent 2 |
| Heterogeneous | Task | 2 vs. 1 | | 3 vs. 1 |
| | Hunters | Agent 1 and Agent 2 | | Agent 1, Agent 2, and Agent 3 |
| | Direction of transfer | Agent 2 | → | Agent 1 |
| | | Agent 1 | → | Agent 2 |
| | | Agent 1 | → | Agent 3 |

transferred from hunter1. Then, knowledge outputs the action values of hunter 1, and hunter 3 translates it to its own actions by utilizing action ontology. Finally, hunter 3 calculates the combined knowledge (Eq. (8)), and it selects a valuable action by using the Boltzmann distribution (Eq. (3)). When the hunter 3 reuses the knowledge of hunter 1 by using state ontology, if the hunter 3 detects another hunter (hunter 1) in the grid world, then the hunter 3 can behave in cooperative actions between hunter 1 and hunter 2 in the source task. Here, we assume that the two ontologies and the necessary search function are preprogrammed in all hunters.

## V. Experimental Results and Discussion

In this section, we describe the experimental results and discuss *Jumpstart* (JS), which is the difference between the value resulting from an agent with transfer and one without transfer. This is formulated as follows:

$$\mathrm{JS} = \frac{1}{100}\left(\sum_{i=1}^{100} s_i^{wt} - \sum_{i=1}^{100} s_i^{t}\right) \tag{15}$$

Here, $s_i^{wt}$ is the number of steps of the learning curve without transfer; $s_i^t$ is the number of steps of the learning curve with transfer. Moreover, to aid intuitive understanding, we define the *ratio of JS* (RJS), as follows.

$$\mathrm{RJS} = \sum_{i=1}^{100} s_i^{t} \Big/ \sum_{i=1}^{100} s_i^{wt} \tag{16}$$

If we obtained the result for JS that the number of steps until convergence for the learning curve with transfer exceeds the analogous value without transfer, then transfer is not effective since the final performance of learning is worse than without transfer. Hence, *Difference in convergence steps* (DCS) is defined as follows, and we also define the *ratio of DCS* (RDCS).

$$\mathrm{DCS} = \frac{1}{100}\left(\sum_{i=9901}^{10000} s_i^{wt} - \sum_{i=9901}^{10000} s_i^{t}\right) \tag{17}$$

$$\mathrm{RDCS} = \sum_{i=9901}^{10000} s_i^{t} \Big/ \sum_{i=9901}^{10000} s_i^{wt} \tag{18}$$

DCS is the average steps in the final 100 episodes in the learning curve with transfer and without transfer. DCS and RDCS express the difference in convergence performance between agents with knowledge transfer and without transfer.

### A. Results for Self-transfer

In this experiment, the result of learning without transfer shows improved performance (Fig. 9(a)). This learning curve does not converge to a single solution, in contrast to the performance of general reinforcement learning in a static environment; this difference occurs because the agents in all of our experiments learn in a dynamic environment.

The values of JS are shown in Table II along with the values of other parameters such as RJS, DCS, and RDCS. The JS value of self-transfer experiments is 297.16 steps, and the improvement rate with a JS is 80%. The learning curve of self-transfer exhibits an obvious JS relative to the "without transfer" condition. Moreover, the number of steps of the final 100 episodes in the learning curve with transfer is lower than the number of steps of the final 100 episodes in the learning

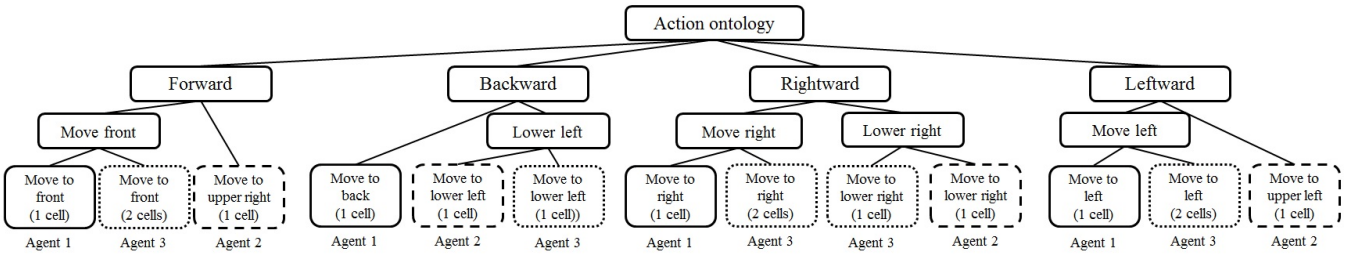



Fig. 7: Action ontology. We map the instance of actions to a similar upper class. In this action ontology, for example, “Move to right (1 cell)” of hunter 1 and “Move to right (2 cells)” of hunter 3 are similar actions in the action ontology. If the ontology designer has not specified similar actions, actions of agents are connected to upper classes.

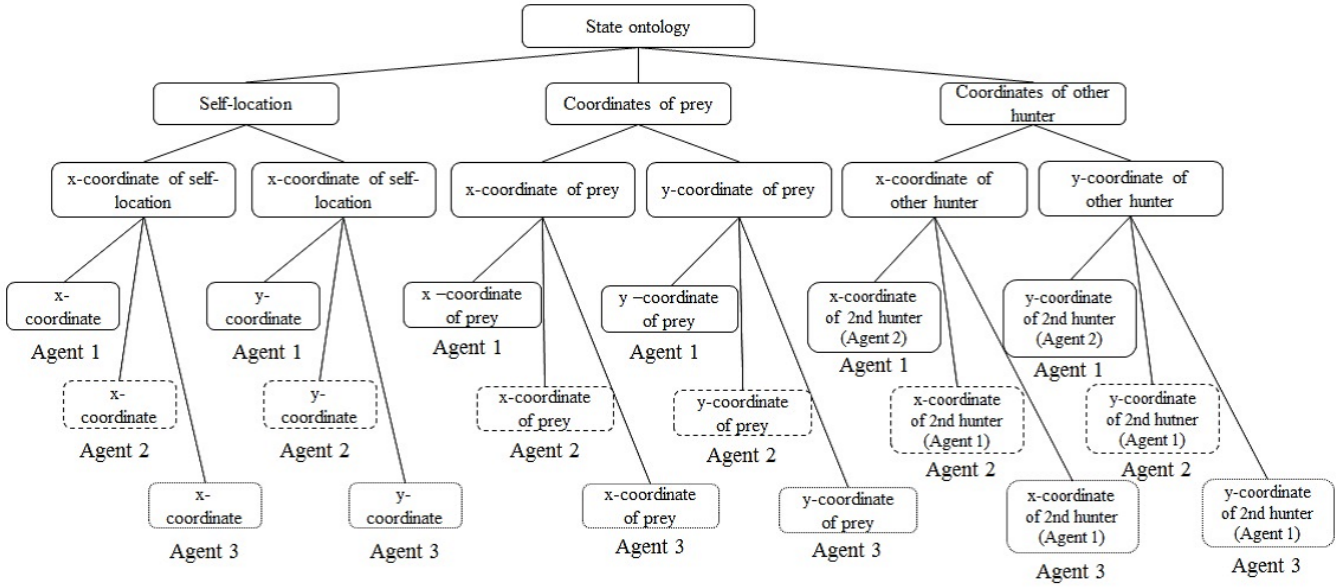



Fig. 8: State ontology, with coordinates of cooperative agents. Instances of “Self-location” and “coordinates of prey” of each hunter are connected to the same class. When the hunter 3 reuses the knowledge of hunter 1, information about “coordinates of hunter 1” are put in the state ontology as “coordinates of hunter 2” according to the knowledge of hunter 1.

TABLE II: Comparison of JS, RJS, DCS and RDCS in each experiment.

| Experiment | JS | RJS | DCS | RDCS |
|---|---|---|---|---|
| Self-transfer | 297.16 | 0.20 | 42.84 | 0.64 |
| Different action space | 108.35 | 0.42 | -3.64 | 1.06 |
| Different state space | 4433.01 | 0.06 | 1095.25 | 0.19 |
| Heterogeneous | 3059.12 | 0.28 | 602.67 | 0.51 |

curve without transfer. in the learning curve without transfer. This result indicates the effectiveness of reusing knowledge, and this emergence effect is considered the basic effect of transfer learning. In this experiment, the agent also use the HTL, and so this result indicates reappearance of the effect of transfer learning.

### *B. Results with Different Action Spaces*

The results for the learning curves are shown in Fig.9(b). In this experiment, the results exhibit an obvious JS. The value of JS is 108.35 steps, which means that the performance of the target task agent improved 58% from agents without transfer. This result indicates the effectiveness of reusing knowledge utilizing HTL. For the learning curve in the “with transfer” condition in Fig. 9(b), the curve decreases more slowly than the curve for the initial episodes. This phenomenon shows that the agent learned the new environment as a target task by using the transferred knowledge.

The value of DCS for learning curve in the “with transfer” condition is greater than that in the “without transfer” condition. This result indicates that the final state of learning with transfer is 1% worse than the case of without transfer. This DCS value is considered small enough for effectiveness.

### *C. Results with Different State Spaces*

The results for learning curves are shown in Fig. 9(c). In this experiment, the result also exhibits a large value for JS,

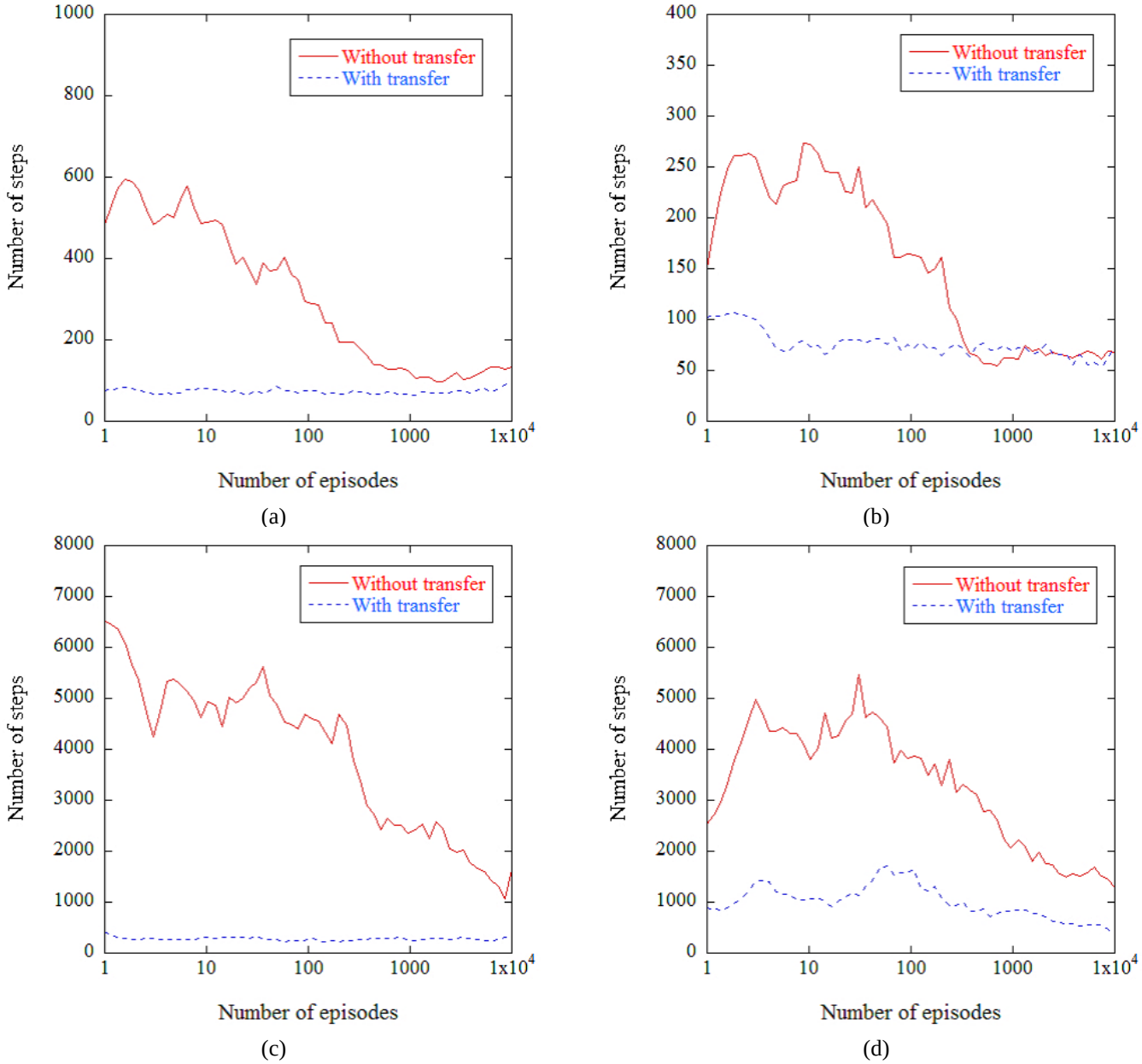



Fig. 9: Comparison of learning curves between "without transfer" and "with transfer". (a) Result of self-transfer condition. In this experimental condition, obtained knowledge is common knowledge of all experimental conditions. (b) Result of experiment with different actions spaces. (c) Result of experiment with different state spaces. (d) Result of experiment with heterogeneous state spaces.

4433.01 steps, which is an improvement rate from the JS of 94% relative to the case without transfer. Additionally, the DCS value is also excellent, with improvement of 81%. Together, these phenomena mean that the performance of the agent in the "with transfer" condition is greater than performance of the agent in the "without transfer" condition at the final state of learning. The main reason for this is the adjustment of learning parameters, such as $\alpha$, $\gamma$, and $T$. In reinforcement learning in a dynamic environment, the agent's behavior is sensitive to tuning of the learning parameters. Such sensitivity is clearly seen in this experimental result, where the performance of the agent in the "without transfer" condition does not reach the performance of the agent in the "with transfer" condition.

### *D. Results with Heterogeneous Conditions*

For the experiment with heterogeneous conditions, an obvious JS value is present, as shown in Fig. 9(d). The DCS value was high at 49%. These results indicate the effectiveness of HTL in a heterogeneous MARL situation.

However, the learning curve in the "with transfer" condition is unstable in the above experimental conditions. The main cause of this is difficulty of tasks. In this experiment, the task is 3 vs. 1, and all agents are heterogeneous. Moreover,

reused knowledge is transferred from heterogeneous agents. This result indicates that the agents can reuse the knowledge, although the agents require a relearning process for the target task.

## VI. Conclusion

In this paper, we proposed KCF for implementation of MARL, and presented HTL as a transfer learning method suitable for large numbers of heterogeneous learning agents. The HTL method is one of the functions of KCF. We also carried out simulation experiments under four transfer conditions with the pursuit game used for the environment and tasks. The experimental results suggest that HTL can transfer knowledge among heterogeneous agents and several tasks.

For our future work, we plan to demonstrate the effectiveness of HTL by conducting experiments in actual multi-robot learning systems. In the simulations, the action sets and state sets were discrete, and it seems hard to apply discrete sets to real robot systems. Instead, HTL should be applied to continuous sets for real situations. An evaluation system of ontology and an autonomous restructuring mechanism should be developed as new functions. These functions are important because there is an increased probability of choosing the wrong design for ontology because the architecture of ontologies (e.g., instances and classes along with the relations among those factors) depends on the degree of the ontology developer's experience. For application in real-world situations, our proposed system needs a system for autonomous ontology restructuring by agents.

## Acknowledgment

This work was partially supported by the Research Institute for Science and Technology of Tokyo Denki University Grant Number Q14J–01 Japan.

## References

[1] H. Sugiyama, T. Tsujioka, and M. Murata, "Coordination of rescue robots for real-time exploration over disaster areas," In Proc. of the Object Oriented Real-Time Distributed Computing (ISORC) 2008 11th IEEE International Symposium on, pp.170–177. IEEE, 2008.

[2] A. Marino, L. E Parker, G. Antonelli, and F. Caccavale, "A decentralized architecture for multi-robot systems based on the null-space-behavioral control with application to multi-robot border patrolling," Journal of Intelligent & Robotic Systems, Vol. 71, No. 3-4, pp. 423–444, 2013.

[3] R. D'Andrea, "Guest editorial: A revolution in the warehouse: A retrospective on kiva systems and the grand challenges ahead," Automation Science and Engineering, IEEE Transactions on, Vol. 9, No. 4, pp. 638–639, 2012.

[4] M. Tan, "Multi-agent reinforcement learning: Independent vs. cooperative agents," In Proc. of the tenth international conference on machine learning, Vol. 337. Amherst, MA, 1993.

[5] S. Arai, K. Sycara, and T. R Payne, "Experience-based reinforcement learning to acquire effective behavior in a multi-agent domain," In Proc. of the PRICAI 2000 Topics in Artificial Intelligence, pp. 125-135. Springer, 2000.

[6] E. Yang, and D. Gu, A survey on multiagent reinforcement learning towards multi-robot systems, In Proc. of the Computational Intelligence and Games (CIG) 2005 IEEE Symposium on, 2005.

[7] M. J Matarić, "Reinforcement learning in the multi-robot domain. Autonomous Robots," Vol. 4, No. 1, pp. 73–83, 1997.

[8] M. Waibel, M. Beetz, J. Civera, R. D'Andrea, J. Elfring, D. Galaván-López, K. Haüssermann, R. Janssen, J. M M Montiel, A. Perzylo, B. Schießle, M. Tenorth, O. Zweigle, and R. van de Molengraft, "A world wide web for robots RoboEarth," Robotics & Automation Magazine, IEEE, Vol. 18, No. 2, pp. 69–82, June 2011.

[9] G. Hu, W. P. Tay, and Y. Wen, "Cloud robotics: architecture, challenges and applications," Network, IEEE, Vol. 26, No. 3, pp. 21–28, 2012.

[10] H. Kono, K. Sawai, and T. Suzuki, "Convergence Estimation Utilizing Fractal Dimensional Analysis for Reinforcement Learning," In Proc. of the SICE Annual conference 2013, pp.2752–2757, 2013

[11] H. Kono, K. Sawai, and T. Suzuki, "Hierarchical Transfer Learning of Autonomous Robots with Knowledge Abstraction and Hierarchization," In Proc. of the 19th Robotics Symposia, pp.479-484, 2014. (in Japanese)

[12] M. E. Taylor, Transfer in Reinforcement Learning Domains, Vol. 216. Springer, 2009.

[13] G. Boutsioukis, I. Partalas, and I. Vlahavas, "Transfer learning in multi-agent reinforcement learning domains," In Proc. of the Recent Advances in Reinforcement Learning, pp. 249–260. Springer, 2012.

[14] A. Taylor, I. Dusparic, E. Galván-López, S. Clarke, and V. Cahill, "Transfer learning in multi-agent systems through parallel transfer," In Proc. of the Workshop on Theoretically Grounded Transfer Learning at the 30th International Conference on Machine Learning, Vol. 28, 2013.

[15] T. R Gruber, "A translation approach to portable ontology specifications," Knowledge acquisition, Vol. 5, No. 2, pp. 199–220, 1993.