An Ontology-based Cooperative Environment for Real-world Agents

H. Takeda, K. Iwata, M. Takaai, A. Sawada, and T. Nishida
Graduate School of Information Science,
Nara Institute of Science and Technology
8916-5, Takayama, Ikoma, Nara 630-01, Japan
takeda@is.aist-nara.ac.jp
http://ai-www.aist-nara.ac.jp/

Abstract

In this paper, we propose a framework for communication and cooperation among heterogeneous real-world agents based on shared ontologies. Ontologies give a background of knowledge to share among agents, that is, systems of concepts are defined which are used to communicate to each other. We present three different ontologies, namely, ontologies for object, space, and action. Based on these ontologies, we then show mediation mechanism to generate action sequences for agents from given tasks. Our framework is appropriate for cooperation among heterogeneous real-world agents because of (1) adaptiveness for environments and agents, (2) implicit representation of cooperation, and (3) integration of information and real-world agents. We realized and verified our mediation mechanism by applying task planning for mobile robots and computer-controlled instruments.

Introduction

In recent years, many machines and instruments are computerized, i.e., they have CPUs which enable them to communicate to humans actively as well as to communicate to each other with digital manner. The concept *agent* is a key to integrate human and these various kinds of machines.

In this paper, we propose a cooperative environment for real-world agents in which ontologies and mediation play important roles for cooperation. Ontologies give a background of knowledge to share among agents, that is, systems of concepts are defined which are used to communicate to each other. We present three different ontologies, namely, ontologies for object, space, and action. Mediation is to find appropriate agents for given messages from agents(Wiederhold 1992). Since each agent cannot know what agents are accessible exactly in multi-agent systems, mediation is mandatory function for agent communication. In this paper mediation is done with ontologies and knowledge on ability of agents.

In Section 2, we show ontologies needed for real-world agents to cooperate to each other. In Section 3, we discuss how tasks can be mediated to agents by using ontologies. We show the current implementation of our system in Section 4, and test cases performed by it in Section 5. In Section 6, we compare previous work with our approach. Then we conclude the paper in the last section.

Knowledge for cooperation of agents

Our aim is to establish information infrastructure to cooperate heterogeneous real-world agents at knowledge level(Newell 1982), i.e., to clarify what knowledge is needed for those agents for cooperation.

Need for sharing concepts

The simplest way to accomplish a task with multiple agents is to break down the task and design subtasks each of which is executable for each agent. But this approach is not applicable where tasks are dynamically defined like environments human and agents co-exist.

In order to do it more intelligently, agents should understand what parters are doing or requesting and so on. In other words, agents should have common communication abilities to tell and understand intension. It means that they should share not only protocols and languages to communicate but also concepts used in their communication. The latter is called ontology which a system of concepts shared by agents to communicate to each other(Gruber 1993).

Ontology is defined and used mostly in information agents (For example see (Takeda, Iino, & Nishida 1995)(Cutkosky *et al.* 1993)). The primary concern in such studies was to model objects which agents should handle. Modeling objects is not sufficient to realize communication among real-world agents. Modeling space is also important because they should share space to cooperate each other. Modeling action is another important concept because they should understand what other agents do or request[1]. Therefore there are three ontologies, namely ontologies for object, space, and action (see Figure 1).

[1] Another important concept is one for time. In this paper, time is not explicitly described but embedded as shared actions.

Figure 1: Three types of concepts

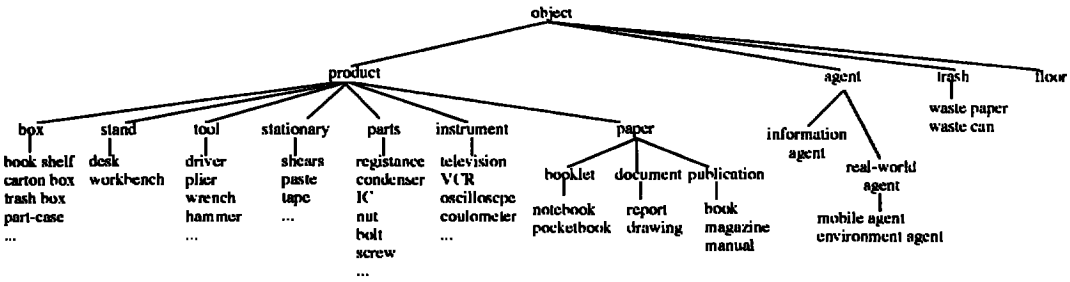

Figure 2: An example for object ontology

Concept for object

The environments are usually fulfilled with a lot of objects, and tasks are usually related to some of these objects. They should share concepts for objects, otherwise they cannot tell what they recognize or handle to other agents.

Difficulty lies that what they can understand are different because the way they can perceive are different. It depends on abilities for sensing, acting, and information processing.

Agents should understand objects to communicate to each other, but a number and kinds of objects may be different. Attributes of objects agents can understand are also different because they should be directly or indirectly associated to attributes which can be detected by agents' sensors.

The policy for making shared concepts is using abstraction levels to represent objects. We build taxonomy of objects as hierarchy of *is-a* relations. Figure 2 shows an example of this ontology which represent objects in workshop. It does not mean that all agents can understand all objects in this figure. Most agents can only understand subsets of those objects because recognition abilities are limited. For example, some agent can recognize a box but cannot recognize difference between a trash box and a parts-case, because it can only detect whether it is a box or not. It is sufficient for this agent to understand concept *box* and its higher concepts.

We provide current position, default position, color, weight for attributes which are common for all objects. Descriptions of attributes have also levels of abstraction. For example, in most abstract description of weight, there are only three values, i.e., *light, middle,* and *heavy.* In other abstract level, it can be represented by more symbols, or numerically. The level of abstraction is determined by abilities of sensing and pattern recognition.

Concept for space

The next important concept for cooperation is concept for space. Since agents are working in the physical world, they should understand space, that is, where they are, where they are moving for, where the target object exists, and so on. Especially it is important for agents to work together. According to sensing and processing abilities of agents, they can have different ways to represent space. For example, agents which move by programmed paths would represent space by paths and points to stop. Some agents cannot have absolute positions but relative position.

We provide the following two types of representation as shared space ontology.

Representation with preposition Relative position is described as combination of preposition and object which is represented as object ontology(Herkovits 1986). We provide seven prepositions, i.e., *at, on, in, in-front-of, behind, to-the-right-of,* and *to-the-left-of.* For example, a position in front of the rack agent is represented as `in-front-of (rack-agent)`. Actual position is determined by agents who interpret representation.

Representation with action Relative position can be also represented by action. For example, it

```
((<action-name>
 (<attribute-name> <attribute-value>)
 (<attribute-name> <attribute-value>)
  ...)
 (<constraints among attributes>)
 (<decomposed-action-name>
  (<attribute-name> <attribute-value>)
  (<attribute-name> <attribute-value>)
   ...)
 (<decomposed-action-name> ...)
  ...
)
```

Figure 3: Syntax for action concept

```
((fetch  (object       ?object-1)
         (from_place   (on ?place-2))
         (to_place     (on ?place-3))
         (subject      ?agent-4))
         (from         (at ?agent-4))
         (to           (in_front_of ?place-3)))
((= (on ?place-2) (:current-place ?object-1)))
(move (subject       ?agent-4)
         (from         (at            ?place-4))
         (to           (in_front_of ?place-2)))
(bring  (object       ?object-1)
         (from_place   (on          ?place-2))
         (to_place     ?place-3)
         (subject      ?agent-4)
         (from         (at          ?agent-4)
         (to           (in_front_of ?place-3))))
```

Figure 4: An example of action concept

is useful for agents who want to achieve action to describe space as "where you can look at the rack" or "where you can meet Agent X." The actual position may be different according to which agent would take action, because ability of action agent can do may be different. No matter actual position differs, it is sufficient to understand positions where such action can be done.

We describe position with combination of an action-related term and object(s). For example, `viewpoint(rack)` means a position where the agent can look at the rack, and `meetingpoint(agent1, agent2)` means where agent1 and agent2 can meet.

Concept for action

The last category for shared concepts is concept for action. Agents should understand what the other agents are doing in order to cooperate with them. Like the other categories, concepts which an agent can understand are also different according to ability of the agent itself, in particular concepts which are directly associated to its physical action. But more abstract concepts can be shared among agents. Concepts associated to agents' physical actions should be related to more abstract concepts shared by them in order to understand each other.

In this paper, we provide nine primitive actions, i.e., *move, grasp, release, find, hold, open, close, h_handover*, and *r_handover*. *h_handover* means the action to hand object someone, and *r_handover* means the action to receive objects handed from someone.

Definition of concept for action consists of name, attributes like subject and starting-point, and constraints among attributes. Constraints are represented as sharing of attribute values. Relation among concepts is decomposition relation, i.e., an action can have an sequence of action which can achieve the original action. Relation among decomposed actions are represented as constraints among attributes. Figure 3 shows syntax for definition of action concepts, and Figure 4 shows an example of definition of action concept.

Mediation

In this section, we discuss how to realize communication among agents with different ontologies. We introduce mediators which break down and translate tasks to be able to understand agents(Takeda, Iino, & Nishida 1995). In this paper, we assume that tasks are given as actions by human.

The function of mediators is to bridge a gap between tasks specified by human and actions which can be done by agents. Since in most cases, tasks should be performed by multiple agents, tasks are decomposed into subtasks to distribute to agents. Ontologies have two roles in this process. Firstly, it is used to understand the given tasks. Since given tasks are what humans want agents to do, they are insufficient and incomplete for specifying a sequence of actions. Ontologies supply information on environments and agents to complete task descriptions. Secondly, it is used to distribute tasks to agents. As we mentioned in the previous section, each agent has its own ontology which is dependent to their physical and information ability. But shared ontologies integrate these agent ontologies using abstraction. Tasks can be translated to a set of local tasks each of which is understandable by some agent by using multiple abstraction levels in ontologies.

We provide four processes to process the given tasks in mediation (see Figure 5.

Supplement of object attributes If necessary attributes of objects are missing in a task description, the mediator can add these attributes using default values in object ontology.

Assignment of agents The mediator assigns agents to perform actions to realize the task. It is done by consulting knowledge on agent abilities which is represented by object, space, and action ontologies.

Action decomposition The mediator decomposes the given action into actions each of which can be executable by some agents. Decomposition of action is

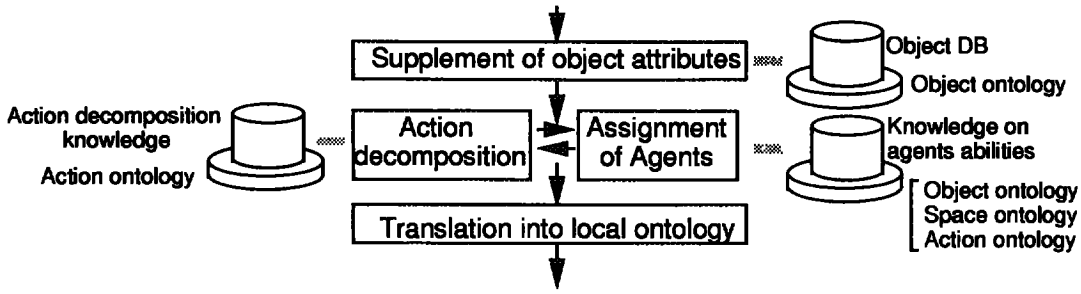

Figure 5: Mediation flow

done by consulting action ontology. Action decomposition and agent assignment are done simultaneously because action decomposition restricts agent assignment and vice versa.

Translation into local ontology All information before this process is represented by the shared ontologies. Before sending out the decomposed messages to agents, the mediator translates each message into one in the local ontology to the receiver agent.

Implementation of Mediators

In this section, we show our preliminary implementation of agents and mediators. Characteristics of our implementation is to use techniques for information agents to both agents and mediators discussed in Ref. (Takeda, Iino, & Nishida 1995).

Types of real-world agents

We used four different real-world agents to demonstrate our approach. Two of them are mobile robots and the others are computer-controlled machines.

Mobile Agent Kappa1a: Kappa1a is a mobile robot which has a stereo CCD camera with two rotational freedoms, and two six-freedom manipulators with hands (see Figure 6). There are two modes for navigation. One is map-based navigation in which the robot can move along with predefined paths and stop at predefined stopping points. In this mode, the robot can avoid obstacles automatically. The other is direct operation in which commands like *move*, *stop*, and *turn* are directly executed. The manipulators are operated either by tele-operating or by commands. There are an infrared radar and eighteen supersonic sensors. There is an on-board computer with a wireless Ethernet. It has interfaces for all actuators and sensors and an interface for TCP/IP by which it can communicate to other on-board computers and Unix computers on the network.

Mobile Agent Kappa1b Kappa1b is the same to Kappa1a except no manipulators are equipped.

Computer-controlled Rack It is an automated rack which can store and take out objects by command in computers (see Figure 7).

Computer-controlled door It is an automated door system which can open and close by command in computers.

Agentification

Those robots and machines are agentified as KQML agents(Finin *et al.* 1994). KQML(Knowledge Query and Manipulation Language) is a protocol for exchanging information and knowledge among agents. KQML is mainly designed for knowledge sharing through agent communication. A KQML message consists of a message type called *performative* like *ask, tell* and *subscribe*, and a number of parameters like *sender, receiver, content* and *language*. For example, a message content is written as a value of parameter *content*. We mostly use KIF (Knowledge Interchange Format)(Genesereth & Fikes 1992) as language to describe message contents. KIF is a language for interchange of knowledge and based on the first-order predicate logic.

Each robot or machine agent consists of three sub-agents, namely *KQML handling sub-agent* which parses and generate KQML messages, *database sub-agent* which holds status of agent itself and environments, and *hardware controlling sub-agent* which sends commands to actuators and to obtain sensor values.

We provide some pure information agents also as KQML agents to offer information processing to those real-world agents. *Image processing agent* receives an images and a color name, and an answer where the color is located in the image. *User-interface agent* accepts users' queries and returns the results. *Object database agent* manages information on objects' attributes like positions based on object ontology, and the content is updated by messages from other agents.

Implementation of mediator

A mediator is implemented as two agents, namely, a *planner* and a *translator*. The planner performs the first three processes listed in Section 3.

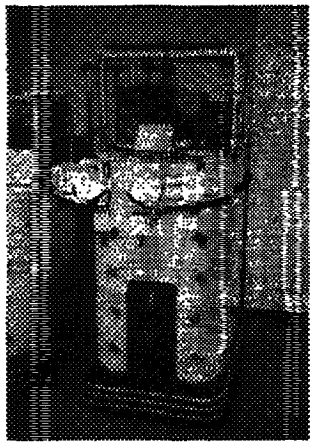

Figure 6: Appearance of Mobile Agent Kappa1a

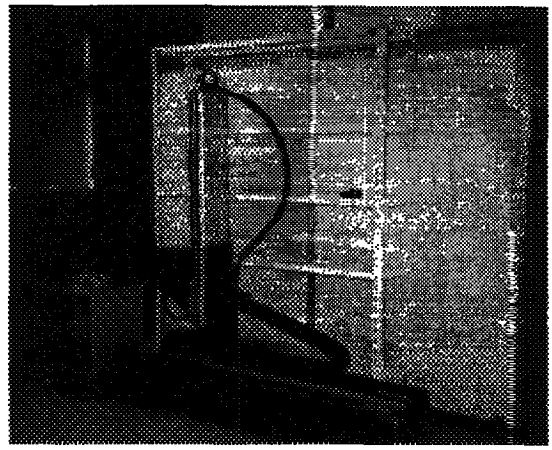

Figure 7: Appearance of the computer-controlled rack

Supplement of object attributes is realized by consulting the object database agent. Decomposition of actions and assignment of agents are invoked recursively. First the mediator tries to find an agent which can execute the given action by consulting knowledge on agents' abilities which is represented with object, space, and action ontologies. If it find an agent, then this process is terminated. Otherwise, it tries to decompose the given action by consulting action ontology. The decomposed actions are associated to each other by sharing their parameters. Then again the mediator tries to find agents which can execute each action. If assignment of agents for all actions are determined, the the current sequence of actions is a solution. Iteration of decomposition and assignment is repeated until all actions are assigned to agents.

The mediator also works as scheduler for action sequences. If sequential actions are supplied, it can keep the sequence by waiting for reply after sending out an action.

The translator performs the last process, i.e., it translates messages written in the shared ontologies into one in local ontologies of the receiver's agent.

Examples

We show examples how the mediator can transform a given task to a sequence of tasks each of which can be executable by an agent.

The situation in the following examples is a very primitive model of an office room where humans and robots work together. There are users, two mobile robots, a computer-controlled rack which can contain books and manuals, a computer-controlled door, desks, and chairs in a room. There are no distinction between space for humans and robots. For example, she can either ask robots to take a book or take one by herself.

Example 1: A simple fetching action

The task is that the user wants to get a manual *Solaris*. Her request is sent to the mediator by the user-interface agent. In the system, it is represented as follows;

```
((fetch (object Solaris))
```

This message is missing important information to determine the task. Then the mediator generates the following representation by consulting action ontology, and then tries to replace variables (denoted *?*) by constants by consulting object ontology.

```
((fetch (object Solaris)(from_place ?place-a)
 (to_place (at operator1))
 (subject ?agent-b)(from ?place-c)(to ?place-d)))
```

This sentence means that agent `?agent-b` at place `?place-c` fetches object `Solaris` from place `place-a` to place `(at operator1)` and stops at place `?place-d`.

First, the mediator consults the object database agents to fulfill insufficient information. In this case, the current position of `Solaris` is found.

Then, since there are no agents to execute action `fetch` by itself, the mediator tries to decompose the action `fetch`. A solution of decomposition of `fetch` is `move` and `bring`. This decomposition generates some constraints between these actions, for example, subject of `move` and subject of `bring` should be the same, the ending place of `move` and the staring place of `bring` should be the same. Since there are no agents to execute `bring` by itself, `bring` is decomposed again, and `handover` and `carry` are obtained instead. Furthermore `handover` is decomposed into `r_handover` and `h_handover`. In this case, `h_handover` is assigned to the rack agent because the object is located in the rack, and the other actions are assigned to the Kappa1b agent because it can move, receive objects from the rack, and carry objects. Since this assignment of agents

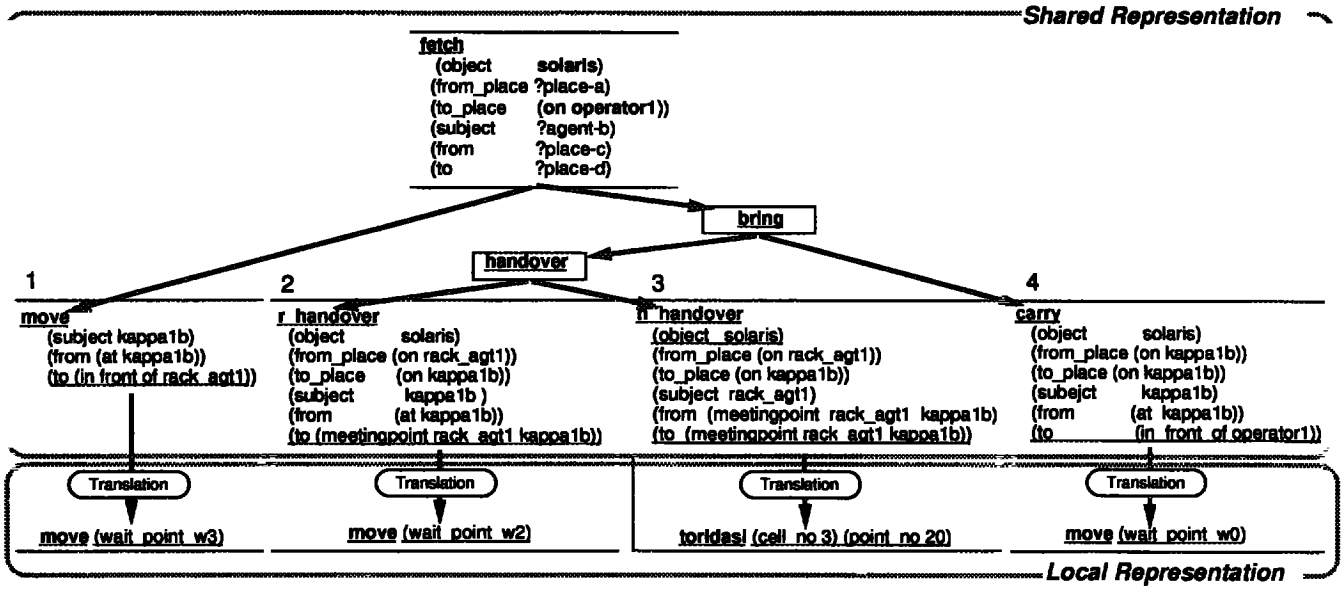

Figure 8: Mediation of action `fetch`(1)

satisfies all constraints among actions, decomposition of action is terminated.

Finally, representation of those actions are translated into local representation which is different among agents (see Figure 8). The actions performed by this decomposition is illustrated in Figure 9.

Example 2: fetch an object at unknown place

In the first example, we could find the object at the place which is registered in the object database agent. Since the user can access objects in the room by herself, it is not guaranteed. In such case, we should find where the object is before deciding how to fetch it.

In the following example, we assume that the current position of the object is unknown but that its default position is known.

In this case, the first solution of decomposition of `fetch` (Example 1) is failed because of lack of the current position, and then it is decomposed into `search` and `fetch`. Action `search` is decomposed into `move` and `find` which are done by a single agent. Since action `move` should be done by a mobile agent and `find` by an agent with camera, `Kappa1a` or `Kappa1b` should be assigned. Unfortunately, these agents do not have image processing ability. This process is down by the image processing agent and the result is returned to mobile agents.

The effect of action `search` is to tell the current position of the object to the object database agent. Then the same decomposition of `fetch` to Example 1 is succeeded because the current position of the object is now available. The total actions are shown in Figure 10. Mediation of action `fetch` including action `search` is shown in Figure 11.

Related Work

Cooperation among multiple real-world agents can be decomposed into two processes, i.e., collaboration and coordination(Noreils 1992). In collaboration, tasks are decomposed into subtasks and subtasks are assigned to agents or a set of agents where it is important to identify structure of tasks. In coordination, schedules of actions planned by agents are adjusted and re-planned to avoid conflicts, where constraint solving in time and space is crucial. Our work mainly concerned with collaboration process, while coordination is realized by a simple planner in the mediator which schedules sequences of actions.

Many studies are done with coordination among multiple real-world agents. For example, (Ishida *et al.* 1994) propose coordination of agents to complement functions among agents by communication. (Schweiger 1994) showed an architecture for forming teamwork of agents dynamically. In these studies, collaboration is out of interest because tasks are given or defined in advance.

On the other hand, in (Noreils 1992), Noreils proposed a framework for cooperative and autonomous mobile agents in which both collaboration and coordination are considered. In collaboration tasks are decomposed into subtasks and subtasks are allocated through a set of agents. All information for this process is included in task descriptions while our approach divides such information into three categories. It is less

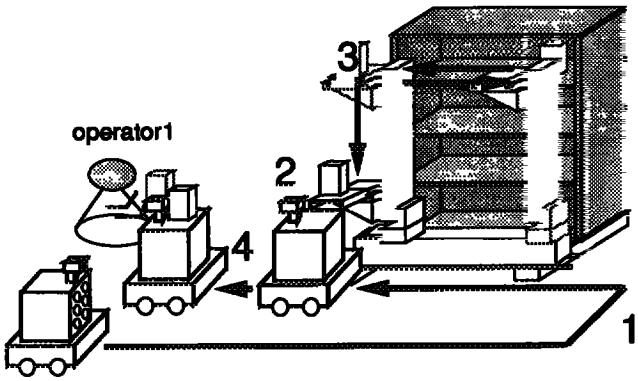

Figure 9: Performance to realize `fetch(1)`

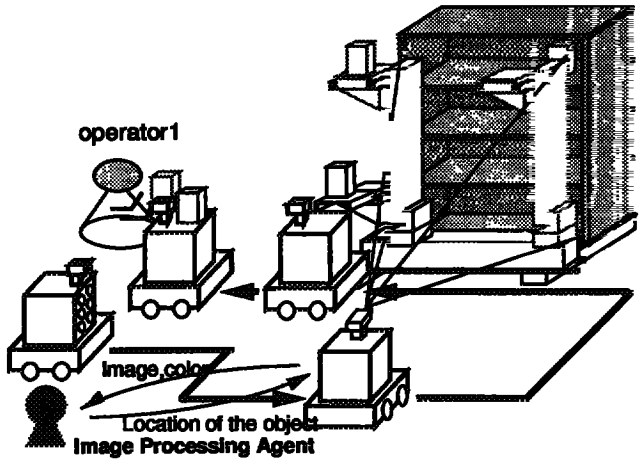

Figure 10: Performance to realize `fetch(2)`

re-usable when increasing environments and agents.

Abstraction is important to deal with real-world information which may be enormous. The use of abstraction levels in resource allocation is shown in (Choueiry & Faltings 1995). The abstraction is used for solving constraints and is generated dynamically, while abstraction in our approach is static. Dynamic abstraction is useful when constraint solving is critical. Our aim for abstraction is rather to narrow problem domains which are possibly huge.

Conclusion and Further Work

In this paper, we clarified what should be shared to realize cooperation among autonomous real-world agents, and built three types of shared ontologies. We then realized mediation for given actions by using those ontologies, which is to organize cooperation among agents. We verified our mediation mechanism by applying task planning of our robots and computer-controlled instruments.

There are three advantages of this approach, i.e., adaptiveness to environments and agents, implicit representation of cooperation, and integration of real-world and information agents.

Ontologies are re-usable for new environments. We describe environments by ontologies which have different abstract levels. Some detail descriptions of ontologies may not be re-usable, but other abstract descriptions are re-usable for new environments. Only we should do is to add new detail descriptions which differ from former environments. It is also easy to add new different agents. If new agents which are different in performance from existing agents, we simply update knowledge for agents' ability. If new agents are different in function, new actions should be added to action ontology.

We realized cooperation among agents without specifying cooperation explicitly. Cooperation is represented implicitly in constraints among decomposed actions and constraints to assign agents derived from knowledge for agents' ability. Cooperation is dynamically determined by solving these constraints. It is desirable for multi-agent systems in the real world because it is unpredictable which agents are available.

There are no distinction between real-world and information agents in our system. We encapsulated real-world agents by technology developed for information agents. It makes easy to integrate real-world agents with information agents which offer various computation facilities.

There some problems to solve. The first one is decentralizing functions of mediation. The current implementation is that mediator manages all processes to mediate actions. But some processes are suitable to be solved by local communication like planning of agents' own schedules. Then negotiation among agents should be needed.

Coordination level should be considered more seriously for more complicated tasks, for example, synchronization and conflict solving is in-avoidable for tight scheduling of actions.

References

Choueiry, B. Y., and Faltings, B. 1995. Using abstraction for resource allocation. In *Proceedings of the 1995 IEEE International Conference on Robotics and Automation*, 1027–1033.

Cutkosky, M. R.; Engelmore, R. S.; Fikes, R. E.; Genesereth, M. R.; Gruber, T. R.; Mark, W. S.; Tenenbaum, J. M.; and Weber, J. C. 1993. PACT: An experiment in integrating concurrent engineering systems. *IEEE Computer* January 1993:28–38.

Finin, T.; McKay, D.; Fritzson, R.; and McEntire, R. 1994. KQML: An information and knowledge exchange protocol. In Fuchi, K., and Yokoi, T., eds., *Knowledge Building and Knowledge Sharing.* Ohmsha and IOS Press.

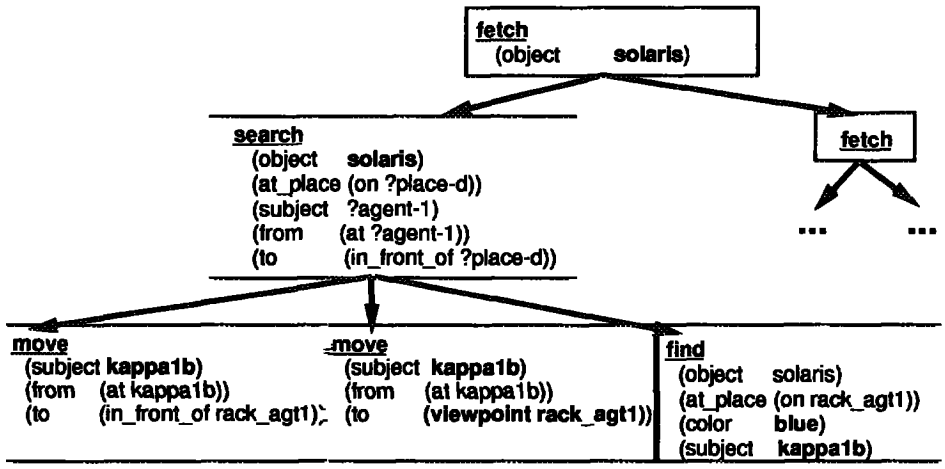

Figure 11: Mediation of action `fetch(2)`

Genesereth, M., and Fikes, R. E. 1992. Knowledge interchange format, version 3.0 reference manual. Technical Report Technical Report Logic-92-1, Computer Science Department, Stanford University.

Gruber, T. R. 1993. Toward principles for the design of ontologies used for knowledge sharing. Technical Report KSL 93-04, Knowledge Systems Laboratory, Stanford University.

Herkovits, A. 1986. *Language and spatial cognition.* Cambridge University Press.

Ishida, Y.; Asama, H.; Tomita, S.; Ozaki, K.; Matsumoto, A.; and Endo, I. 1994. Functional complemnet by cooperation of multiple autonomous robots. In *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, 2476–2481.

Newell, A. 1982. The knowledge level. *Artificial Intelligence* 18:82–127.

Noreils, F. R. 1992. An architecture for cooperative and autonomous mobile robots. In *Proceedings of the 1992 IEEE International Conference on Robotics and Automation*, 2703–2710.

Schweiger, J. 1994. Facilitating teamwork of autonomous systems with a distributed real-time knowledge base. In *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, 2883–2888.

Takeda, H.; Iino, K.; and Nishida, T. 1995. Agent organization and communication with multiple ontologies. *International Journal of Cooperative Information Systems* 4(4):321–337.

Wiederhold, G. 1992. Meidation in the architecture of future information systems. *IEEE Computer* 25(3):38–49.