AGENT ORGANIZATION AND COMMUNICATION WITH MULTIPLE ONTOLOGIES

HIDEAKI TAKEDA
Graduate School of Information Science
Nara Institute of Science and Technology (NAIST)
8916-5, Takayama, Ikoma, Nara 630-01, Japan

and

KENJI IINO
Graduate School of Information Science
Nara Institute of Science and Technology (NAIST)
8916-5, Takayama, Ikoma, Nara 630-01, Japan

and

TOYOAKI NISHIDA
Graduate School of Information Science
Nara Institute of Science and Technology (NAIST)
8916-5, Takayama, Ikoma, Nara 630-01, Japan

ABSTRACT

In this paper, we discuss how ontology plays roles in building a distributed and heterogeneous knowledge-base system. First we discuss relationship between ontology and agents in the Knowledgeable Community which is a framework of knowledge sharing and reuse based on a multi-agent architecture. Ontology is a minimum requirement for each agent to join the Knowledgeable Community. Second we explain mediation by ontology to show how ontology is used in the Knowledgeable Community. A special agent called *mediator* analyzes undirected messages and infer candidates of recipient agents by consulting ontology and relationship between ontology and agents. Third we model ontology as combination of aspects each of which can represent a way of conceptualization. Aspects are combined either as *combination aspect* which means integration of aspects or *category aspect* which means choice of aspects. Since ontology by aspect allows heterogeneous and multiple descriptions for phenomenon in the world, it is appropriate for heterogeneous knowledge-base systems. We also show translation of messages as a way of interpreting multiple aspects. A translation agent can translate a message with some aspect to one with another aspect by analyzing dependency of aspects. Mediation and translation of messages are important to build agents easily and naturally because less knowledge on other agents is requested for each agent.

Keywords: multi-agnet system, ontology, mediation, agent communication, KQML, KIF

1. Introduction

Large scale Knowledge base is indispensable to put AI theories to work in the real world. There are two approaches to realize large scale knowledge bases. One is to build a large scale knowledge base system such as Cyc [8, 10]. The other is to provide a framework of knowledge sharing and reuse by common languages and ontologies among different systems. The purpose of our project called the Knowledgeable Community[12, 13] is to provide a framework of knowledge sharing and reuse based on a multi-agent architecture.

It is most important but difficult for knowledge sharing and reuse how to use different concept structures together. Each system or each agent adopts its own concept structure that is the way how concepts are defined and associated to each other. Each concept structure has a domain that its concepts cover and a perspective that is policy how to describe the domain as concepts. When two agents with different concept structures try to communicate to each other, difference in concept structure disturbs their communication. There are mainly two reasons for it. One is that it is difficult for each agent to know what concepts are used and to know which agent to talk to. The other is, even if they find relationship among their concept structures, it is difficult for each agent to interpret concept structure of the other agent.

We provide ontologies which are vessels to put concepts and their relations used in each agent to solve the first problem. Each agent is required to declare its ontology explicitly. We realize mediation mechanism using ontology that can suggest possible agents to respond the given undirected messages.

For the second problem, we allow multiple aspects for concepts and provide relations among different aspects which are used to translate information from one aspect to the other.

In Section 2, we show our approach of building a large-scale knowledge-base system based on a multi-agent architecture. In particular we identify relationship between agents and ontologies. Then we show mediation of undirected messages as application of use of ontology in Section 3. While we assume a single ontology until Section 3, we introduce *aspect* as component of ontology and define ontology as structure of aspects in Section 4. This definition allows us co-existence of multiple ontologies. We also define relationship between aspects, and show how representation in an aspect can be translated into representation in other aspect. Section 5 discusses some related work, and Section 6 summarizes the paper.

2. Ontology in the Knowledgeable Community

The Knowledgeable Community is an artificial community of cooperating agents. Each agent is supposed to represent some computing ability like problem solver or database, or human interaction, or mixture of computing ability and human interaction.

As the first step to realize the Knowledgeable Community, we are now working on building KC_0 system. Agents in the KC_0 are required to have the following abilities;

Communication ability Each agent can communicate to other agents. It is an indispensable requirement to be a member of the community. This requirement implies that each agent should share communication languages (a unique language is not required).

Declaration of its concept structure Each agent should declare its concept structure, because it enables other agents to guess what kind of concepts an agent can deal with.

Declaration of its processing ability Each agent should declare what it can do or at least what it is expected to do.

The first requirement is realized by providing KQML[2] and KIF[4] as standard protocols (actually KQML is mandatory, but KIF is recommended).

The second requirement (declaration of agents' concept structures) is realized by sharing ontology, i.e., each agent has part of ontology[1]. Then other agents can guess relation between itself and the other agent by comparing their part of ontology.

In KC_0, ontologies are built using *frame ontology* as meta-ontology, i.e., there are classes, relations, and hierarchy of classes. Figure 1 shows ontology of travel plan ontology used in our prototype system KC-Kansai.

In the Knowledgeable Community, we provide ontology server as manager of ontology and relationship between ontology and agents. Its main functions are

(i) accepts a pair of an agent and a part of ontology used by the agent,

(ii) keep ontology consistent,

(iii) reply part of ontology associated to requested ontology,

(iv) reply part of ontology associated to requested agents,

(v) reply a set of agents associated to requested ontology.

3. Agent Communication with a Single Ontology

In this section, we assume a single ontology. Since it means that all agents agree usage of all concepts in the ontology, there are no confusion to interpret messages, i.e., only a single interpretation exists for every message.

Then the next problem is mediating messages to appropriate agents. Even if all agents agree to use a single ontology, they can not interpret all of the ontology but can only deal with some part of the ontology. Every message should be processed by agents whose ontology matches ontology of the message. On the other hand, since it is not mandatory for agents to have knowledge on other agents, it is natural for agents to make messages whose recipients are not determined. Therefore it is an important task for the Knowledgeable Community to *mediate* such undirected messages.

[1]Ontology can be multiple as long as they are connected with each other. We will discuss this issue in Section 4

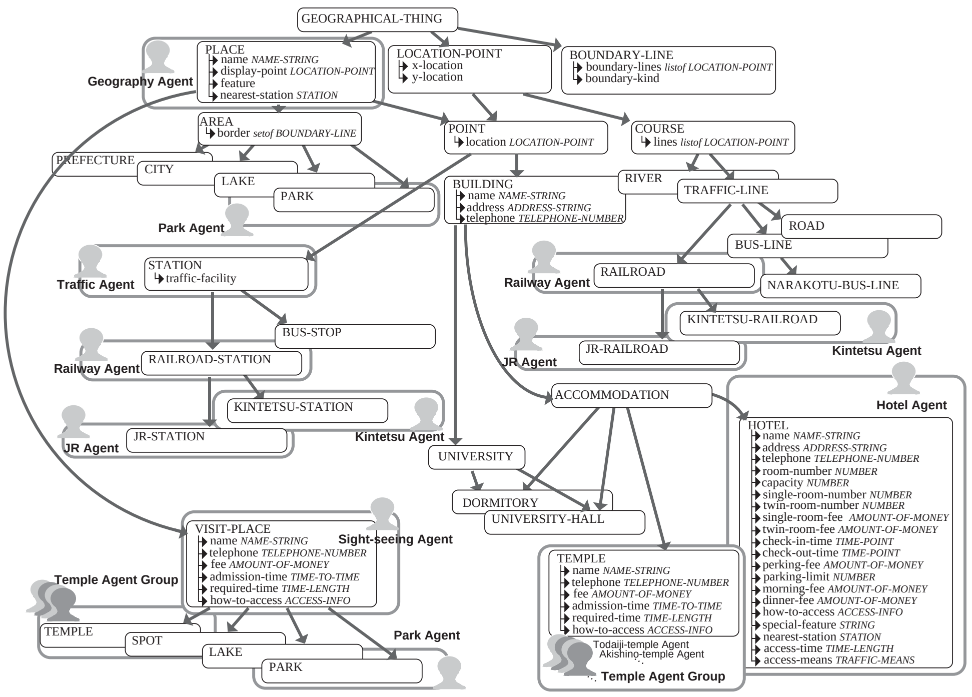

Figure 1: An Example of Ontology

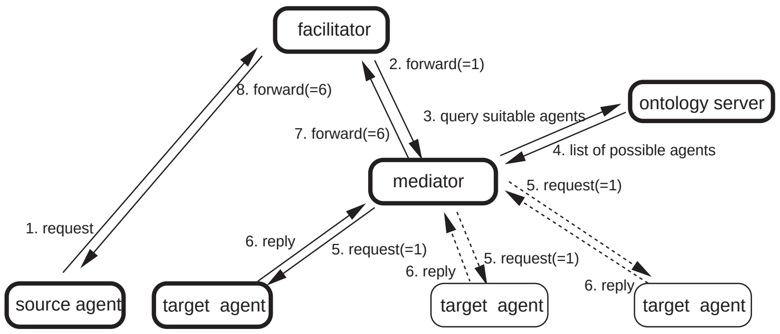

Figure 2: Mediation by Ontology

Since all agents are mapped into the ontology, we can determine appropriate agents for undirected messages by using the ontology. The ontology server knows ontology and relations between ontology and agents. It can reply candidates of agents associated to concepts in the given message by searching concept hierarchy for feasible agents.

Mediation Procedure The mediation is performed by the facilitator[14, 1], mediator and the ontology server (see Figure 2).

All message transmission is made through facilitators. When the recipient is specified, the facilitator will just pass the message to the recipient. Otherwise, the facilitator will forward the message to the mediating agent (we call *mediator*) which will determine the recipient based on knowledge about agents and ontology.

The mediator can determine feasible recipient agents by consulting the ontology server, and send out the message to each of the feasible recipient agents repeatedly until the successful replying message is returned.

The ontology server analyzes received messages to suggest feasible agents. First it detects main classes of messages by checking predicates in contents of messages. Then, it searches agents associated to main classes, then subclasses of them, and superclasses of them in order. Finally the mediator returns a list of candidate agents associated to the given message.

Figure 3. shows how a messages is forwarded for mediation. They are represented in KIF (Knowledge Interchange Format)[3, 4] and KQML (Knowledge Query and Manipulation Language)[2]. Message numbers in this figure are corresponding to those in Figure 2. Note that the content of the original message is not unchanged, but that only KQML performative types are changed in the process.

4. Multiple Aspects

In this section, we introduce *aspect* as component of ontology in order to realize multiple ontologies.

```
(ask-one :content (and (hotel ?x)
                       (name ?x "Nara-hotel")
                       (nearest-station ?x ?y))
         :aspect ?y :language KIF)
```

(a) Message 1

```
(broker-all :content
  (ask-one :content (and (hotel ?x)
                         (name ?x "Nara-hotel")
                         (nearest-station ?x ?y))
           :aspect ?y :language KIF)
  :reply-with q1)
```

(b) Message 2

```
(recommend-all :content
  (ask-one :content (and (hotel ?x)
                         (name ?x "Nara-hotel")
                         (nearest-station ?x ?y))
           :aspect ?y :language KIF)
  :reply-with q1)
```

(c) Message 3

```
(ask-one :content (and (hotel ?x)
                       (name ?x "Nara-hotel")
                       (nearest-station ?x ?y))
         :receiver sight-seeing-agent
         :aspect ?y :language KIF)
:reply-with q1)
```

(d) Message 4

Figure 3: Forwarded Massages for Mediation

4.1. Aspects

In the previous section, we assume a single ontology. But it is not easy to build large ontologies in fact. One of the reasons is that we often confuse concepts from different conceptualization when we try to build large ontologies.

For example, Figure 4 shows how concept "temple" is modeled differently. One may think temple as item in textbook of history, so "founded year", "sect", "historical events", and so on are used with "temple". One may think temple as place for religion, so "doctrine", "parishioners", "branch temples", "chief priest" are concepts related to "temple". Of course, some conecpts are appeared in more than one aspect like "name" and "location".

Mixture of concepts from different conceptualization often confuse us. Some different concepts in different aspects may denote the same fact, for example "chief priest" in *aspect-for-temple-as-religious-place* and "representative" in *aspect-for-temple-as-public-organization* may be the same. On the other hand, concepts used in more than one aspect may denote different facts, for example "name" is used in common, but its meaning is either religious name or historical name or official name with according to aspect it is used. In such cases, the more concepts are collected, the less clear are meanings of concepts.

Each concept is meaningful if and only if concept is used in proper way, that is, concept is used with concepts which come from the same conceptualization. We call this unit *aspect.* We can say that an aspect is a consistent view for conceptualization. Then ontology can be composed of some aspects.

We use various aspects, for example in engineering we use aspects like dynamics aspects, kinematics aspects. To model the commonsense world of traveling, we may use aspects like traffic aspect or geography aspect.

There are two issues on aspect. One issue is what should be in aspect. Aspect should have a vocabulary to describe phenomena in its domain. It should also

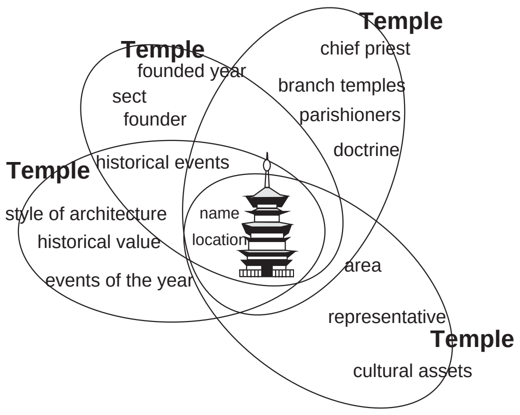

Figure 4: What is Ontology for Temple?

have a theory which associates concepts in its vocabulary. And the theory should be consistent. In the other words, aspect is what we can conceptualize the world without inconsistency[2].

The other issue is how to compose aspects from other aspects.

We provide two types of basic connections among aspects. One is *combination* aspect. This is just integration of aspects for different domains. For example, one of the ways to build *aspect-for-travel* is to combine *aspect-for-hotel* and *aspect-for-traffic*. In this case, concepts like "railway" which is in aspect-for-traffic do not exist in aspect-for-hotel, because domain of modeling is different from each other. In aspect-for-travel, concepts like "tour" are defined using concepts from both aspects.

The other is *category* aspect. This is collection of aspects which share domains but come from different conceptualization.

When a temple is modeled differently we have just shown, we can assume there is a *category-aspect-for-temple*. This aspect has some specific aspects for temple like *aspect-for-temple-as-history-textbook* and *aspect-for-temple-as-religious-place* as component. Since component aspects share domains, it is reasonable (but not mandatory) that there are relations among concepts in different component aspects. Such relations are contents of the category aspects.

Since combination and category aspects can use other combination or category aspects as component, we can construct large aspects from relatively small aspects. We call such relatively large aspects as ontologies.

Figure 5 is an example how different ontologies can be defined with sharing aspects.

[2]The other important content of aspect is *intention*, because aspect is abstraction of the target world by some *intention*. But intention itself is seldom written explicitly. We have tried to represent intention of aspect of object modeling as *function*[16].

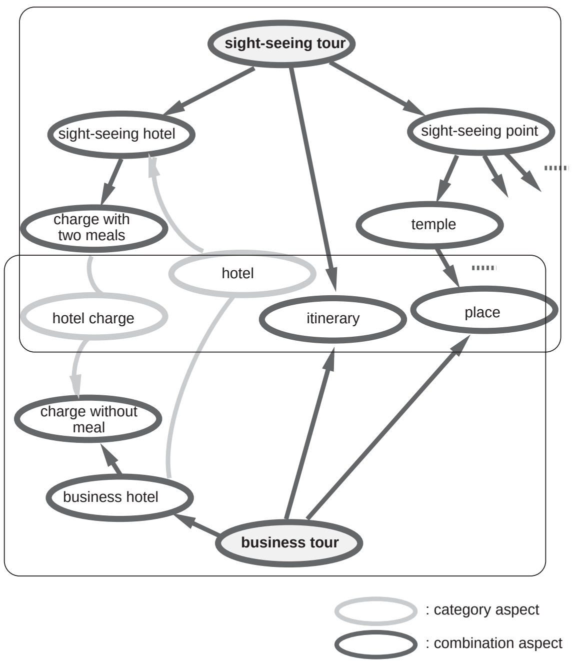

Figure 5: Multiple and Sharable Ontologies

Two aspects can share aspects in their consititution, or be connected by category aspects. We call these two aspects are compatible. That is, they may share or transfer information to each other.

In the following sections, we first describe aspect in a logical framework, and then in a programming framework. Finally, we show how communication between different aspects can be established.

4.2. A Logical Formalization of Aspect

Since our basic policy is to define aspects constructively, we start from defining atomic aspect and then define more complicated aspects.

We assume a first-order language L_E, and predicate *aspect* of L_E. L is a first-order language which is the same to L_E except predicate *aspect* is removed.

4.2.1. Aspect Theory

First we define *atomic aspect*, aspect which does not depend on any other aspects.

Definition 1 *An atomic aspect A with a consistent theory $T(A)$ of a first language L and with a unique name $aspect(A)$ satisfies the following formula.*

$$aspect(A) \leftrightarrow T(A)$$

$aspect(A)$ is an identifier of aspect which has a similar effect to the second argument of predicate `ist` in Ref. [9], and a modal operator in Ref. [11].

Then, we introduce L^m_E and L^m as modal extension of L_E and L respectively. Here we assume domain of individuals are always the same regardless of possible worlds. In the following discussion, we assume this language L^m_E.

A combination aspect is simply defined as follows.

Definition 2 *$T(A_{COM}(A_1, \ldots, A_n))$, aspect theory of combination aspect for aspect $A_1, \ldots, A_n$, is a consitent theory defined as follows;*

$$T(A_{COM}(A_1, \ldots, A_n)) = aspect(A_1) \wedge \ldots \wedge aspect(A_n) \wedge I(A_1, \ldots, A_n)$$

where $I(A_1, \ldots, A_n)$ is a set of formula of language $L^m \cup \{aspect(A_1), \ldots, aspect(A_n)\}$, which means inter-aspect theory among $A_1, \ldots A_n$.

Apparently, it would cause unexpected results if some of aspect theories share predicates. We ideally assume that if the same predicates appear in some aspects, there should share some concepts in them[3]. In such cases, it should be represented by category aspects.

On the other hand, a category aspect is more complicated because it does not imply that both of aspect theories are *always* true. In order to represent a category aspect, we introduce modal operators $\Box$ (necessity) and $\Diamond$ (possibility) and assume S4 modal system. Then a category aspect for two aspects is define as follows.

[3] Of course, it is too strict in practise. In programming approach we allows the same predicates in different meanings.

Definition 3 *$T(A_{CAT}(A_1, \ldots, A_n))$, aspect theory of category aspect for aspect $A_1, \ldots, A_n$, is a consistent theory defined as follows;*

$$T(A_{CAT}(A_1, \ldots, A_n)) = \Diamond aspect(A_1) \wedge \ldots \wedge \Diamond aspect(A_n) \wedge I(A_1, \ldots, A_n)$$

where $I(A_1, \ldots, A_n)$ is a set of formula of language $L^m \cup \{aspect(A_1), \ldots, aspect(A_n)\}$, which means inter-aspect theory among $A_1, \ldots A_n$.

$I(A_1, \ldots, A_n)$ is again an inter-aspect theory among $A_1, \ldots, A_2$.

Since we can use combination and category aspects as component of aspects, we can define hierarchical aspects using combination and category aspects. In other words, An aspect A is represented $A = f(A_1, \ldots, A_n)$ where $A_1, \ldots, A_n$ are aspects and function f is composed by A_{COM} and A_{CAT}.

4.2.2. Inter-aspect Relations

Then we can define relations between aspect, **inclusion** and **strict inclusion**.

Definition 4 *An aspect A is **included** in aspect B if and only if $aspect(B) \vdash \Diamond aspect(A)$.*

Definition 5 *An aspect A is **strictly included** in aspect B if and only if $aspect(B) \vdash aspect(A)$.*

Note that there are two reasons for these relations, i.e., one is composition or category relations among aspects and the other is logical implication. Strict inclusion corresponds *weaker-than* relation in Ref. [11].

Similarly, relations between formula and aspect are defined.

Definition 6 *A formula f is **included** in aspect A if and only if $aspect(A) \vdash \Diamond f$.*

Definition 7 *A formula f is **strictly included** in aspect A if and only if $aspect(A) \vdash f$.*

These definitions mean that there are two types of interpretation of aspect theories. One is represented as *strict inclusion* which is traditional way of inter-theory relation. The other is *inclusion* which takes account of all alternatives of theories. By having two types of interpretation, we can deal with both strictly a single representation and variety of representations.

Theorem 1 *If aspect A is strictly included in aspect B, then A is included in aspect B.*

Another relation is **compatibility** which is criteria two aspects are related to each other[4].

Definition 8 *Aspect A and B is **compatible** if one of the following condition is satisfied;*

[4]Term *compatible* is borrowed from Ref.[7].

(i) *A and B is the same aspect,*

(ii) *there exists aspect C which has both A and B as componet,*

(iii) *there exist compatible aspect A' and B' are componets of A and B respectively.*

Definition 9 *Formula f is* ***compatible*** *with aspect A if and only if there exists aspect B in which f is and B is compatible with A.*

Compatibility assure neither consistency nor translatability between aspect theories, but denotes existence of connection between aspects.

4.2.3. Inter-aspect Theory: Aspect-level Relations

Characteristics of category aspect varies according to its inter-aspect theory. One type of category aspect is *compact.*

Definition 10 *If $I(A_1, \ldots, A_n)$ of a category aspect satisfies the following formula, it is called* ***compact category****.*

$$I(A_1, \ldots, A_n) \vdash \Box(aspect(A_1) \vee \ldots \vee aspect(A_n))$$

Intuitively, all of theories can be true and either of them should be true at any time in compact category aspects. It means that aspect $A_1, \ldots, A_n$ is sufficient to define the category.

Theorem 2 *If a compact category aspect A which has exactly two componets A_1 and A_2, and A_1 is strictly included in A_2, then A_1 is strictly included in A.*

Actually relation between A and A_1 is stronger, i.e., $aspect(A) \vdash \Box aspect(A_1)$. We can say that any formula in A_1 is *rigid* in A by defining *rigidness* as follows;

Definition 11 *A formula f is rigid in aspect A if and only if $aspect(A) \vdash \Box f$.*

Theorem 3 *If a compact category aspect A which has exactly two componets A_1 and A_2 and A_1 is strictly included in A_2, then any formula which satisfies A_1 is rigid in A.*

Rigidness in ontology is discussed in Ref.[7].

The other type of category aspect is *exclusive.*

Definition 12 *If $I(A_1, \ldots, A_n)$ of a category aspect satisfies the following formula, it is called* ***exclusive category****.*

$$I(A_1, \ldots, A_n) \vdash \bigwedge_{k=1,\ldots,n-1} \bigwedge_{l=k+1,\ldots,n} \neg\Diamond(aspect(A_k) \wedge aspect(A_l))$$

4.2.4. Inter-aspect Theory: Object-level Relations

We also describe relations between formulae in different componet aspects by inter-aspect theories. We call such relations *object-level* relations, while we call relations described in Section 3.3 *aspect-level* relations. For example, p in aspect A should imply q in aspect B can be written as follows;

$$\Diamond(aspect(A) \wedge p) \rightarrow \Box(aspect(B) \rightarrow q)$$

More generally a rule "If f_1 in A_1 is true, then f_2 in A_2 should be true" is described as;

$$\Diamond(aspect(A_1) \wedge f_1) \rightarrow \Box(aspect(A_2) \rightarrow f_2)$$

If p in A and q in B should be equivalent to each other, then

$$(\Diamond(aspect(A) \wedge p) \rightarrow \Box(aspect(B) \rightarrow q)) \wedge (\Diamond(aspect(B) \wedge q) \rightarrow \Box(aspect(A) \rightarrow p))$$

Theorem 4 *If a proposition p is equivalent in all componet aspects of a compact category aspect A, p is rigid in A.*

4.3. ASPECTOL*: A Language for Aspects*

Here we show a language of aspects called ASPECTOL(Aspect-based Ontology Description Language), which is an extension of Ontolingua-like ontology definition (see [5]). Syntax of ASPECTOL is shown in Figure 6.

Definition of an atomic aspect consists of declaration of aspect name and definitions of classes, relations, and functions. Figure 7(d)(e) are examples of atomic aspects. Definition of a combination aspect is definition of an atomic aspect and declaration of including aspects (see Figure 7(b)(c))[5]. Definition of a category aspect consists of a set of translation formulae. A translation formula is defined between two aspects in a category aspect, and is defined as `define-translation` which describes logical relation between concepts in both aspects (see Figure 7(a)). It can represent a class of direct relations between formulae in two aspects which we have discussed in the previous section. A left hand side of an implication formula is a formula of the aspect of the first argument and a right hand side is a formula of of the aspect of the second argument.

4.4. Translation among Aspects for Agent Communication

Our approach to deal with the inter-aspect relations is to translate messages among agents with different aspects[6]. Translation formulae are used by a *translation agent* which converts information from one aspect to the other. As we mentioned, if there is only a single ontology, there are no ambiguity to interpret messages. Since we now allow multiple aspects, it is needed to specify aspects the message is based on and sometimes to translate messages from an aspect to the other. Therefore, a translation agent is inserted between the mediator and target agents (see Figure 8). Mediation is performed with almost the same way to a single ontology. The difference is that a step to select aspects is inserted to the procedure.

[5] Definition of classes should not be written directly in compination aspect but be in including aspects. It is just for programming convention.

[6] We discussed more direct use of multiple aspects in Ref. [15]

```
(define-aspect  aspect-name
    Ontoligua definitions are here
    ...
)
```

(a) atomic aspect

```
(define-aspect  aspect-name
  (:use  componet-aspct1, ...)
  (:rename (aspect!predicate . new-name) ...)
)
    Ontoligua definitions are here
    ...
```

(b) combination aspect

```
(define-category-aspect  aspect-name
  (:use  componet-aspct1, ...)
  (:category-type  category-type)
  (:rename (aspect!predicate . new-name) ...)
  (define-translation  aspect-name
     (=> atmoic-formula1 atmoic-formula2)
     ((:query-procedure  nil or t
       :inform-procedure  nil or t
        translation-rule
         ...
       ))
      ...
     )
  (define-translation ...)
   ...
   Ontoligua definitions are here
)
```

(c) category aspect

Figure 6: Definition of aspect

There are two types of messages, i.e., one is informative message as `tell` KQML performative type, and the other is query message as `ask-one` KQML performative. Translation for informative messages is just to translate the given messages, while translation for query messages is to translate answer messages in addition to the given messages (see Figure 9).

4.4.1. Function of Translation Agents

The basic function of a translation agent is as follows;

(i) The translation agent receives a message from an agent (source agent) to a target agent via the mediator.

(ii) It analyzes the message and retrieves translation formulae from the ontology server.

(iii) It composes a new message written in the aspect of the target agent by consulting the translation formulae, and sends the target agent. If the message is a informative one, the procedure ends here.

(iv) It waits and receives an answer message from the target agent.

(v) It composes an answer message written in the aspect of the source agent by consulting the same translation formulae used to compose the original message, and sends it to the source agent.

During the procedure, the translation agent behaves that it can understand a category aspect which includes both aspects in the source and the target agents.

```
(define-category-aspect FEE
                        (fee/a fee/b))
(define-translation FEE
    (=> (fee/A!fee ?fee)(fee/B!fee ?fee))
  ((:query-precedence nil
    :inform-precedence nil
    (-> (fee-value ?fee ?value)
        (and (adult ?fee ?fee1)
             (student ?fee ?fee2)
             (fee-value ?fee1 ?value)
             (fee-value ?fee2 ?value))))
))

(define-translation FEE
    (=> (fee/B!fee ?f) (fee/A!fee ?f))
  ((:query-precedence nil
    :inform-precedence nil
    (-> (and (adult ?fee ?fee1)
             (student ?fee ?fee2)
             (fee-value ?fee1 ?value1)
             (fee-value ?fee2 ?value2)
             (max ?value1 ?value2
                          ?max-value))
        (fee-value ?fee ?max-value))))
  ((:query-precedence nil
    :inform-precedence nil
    (-> (and (adult ?fee ?fee1)
             (fee-value ?fee1 ?value))
        (fee-value ?fee ?value))))
  ((:query-precedence nil
    :inform-precedence nil
    (-> (and (student ?fee ?fee2)
             (fee-value ?fee2 ?value))
        (fee-value ?fee ?value))))
  )
```

(a) Category Aspect `fee`

```
(define-aspect temple/A (TEMPLE)
  (:use fee/A))
(in-aspect temple/A)
(define-class temple (?x)
  :def (and (has-one ?x name)
            (has-one ?x fee)))
(define-function name (?x)
  :-> ?n
  :def (and (temple ?x) (string ?n)))
(define-function temple-fee (?x)
  :-> ?f
  :def (and (temple ?x) (fee ?f)))
```

(b) Combination Aspect `temple/A`

```
(define-aspect temple/B (TEMPLE)
  (:use fee/B))
(in-aspect temple/B)
(define-class temple (?x)
  :def (and (has-one ?x name)
            (has-one ?x fee)))
(define-function name (?x)
  :-> ?n
  :def (and (temple ?x) (string ?n)))
(define-function temple-fee (?x)
  :-> ?f
  :def (and (temple ?x) (fee ?f)))
```

(c) Combination Aspect `temple/B`

```
(define-aspect fee/A (FEE))
(in-aspect fee/A)
(define-class fee (?fee)
  :def (has-one ?fee fee-value))
(define-function fee-value (?fee)
  :-> ?val
  :def (and (fee ?fee) (natural ?val)))
```

(d) Atomic Aspect `fee/A`

```
(define-aspect fee/B (FEE))
(in-aspect fee/B)
(define-class fee (?fee)
  :def (and (has-one ?fee adult)
            (has-one ?fee student)))
(define-class fee-elm (?elm)
  :def (has-one ?elm fee-value))
(define-function fee-value (?elm)
  :-> ?val
  :def (and (fee-elm ?elm)
            (natural ?val)))
(define-function adult (?fee)
  :-> ?elm
  :def (and (fee ?fee) (fee-elm ?elm)))
(define-function student (?fee)
  :-> ?elm
  :def (and (fee ?fee) (fee-elm ?elm)))
(define-function make-fee (?elm1 ?elm2)
  :-> ?fee
  :def (and (fee-elm ?elm1)
            (fee-elm ?elm2)
            (fee ?fee) (adult ?fee ?elm1)
            (student ?fee ?elm2)))
(define-function make-fee-elm (?val)
  :-> ?elm
  :def (and (natural ?val)
            (fee-elm ?elm)))
```

(e) Atomic Aspect `fee/B`

Figure 7: Examples of Definition of Aspect

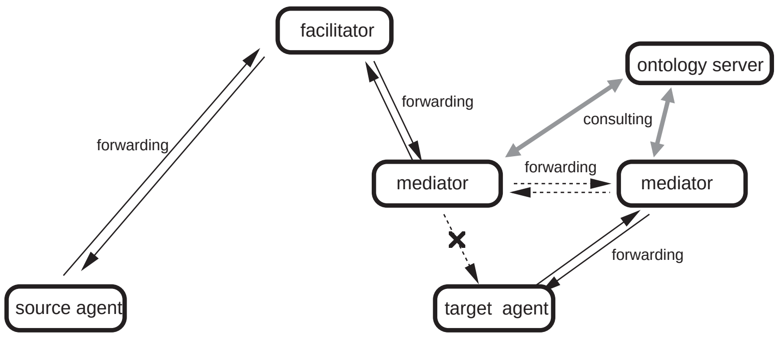

Figure 8: A Translation Agent in the Knowledgeable Community

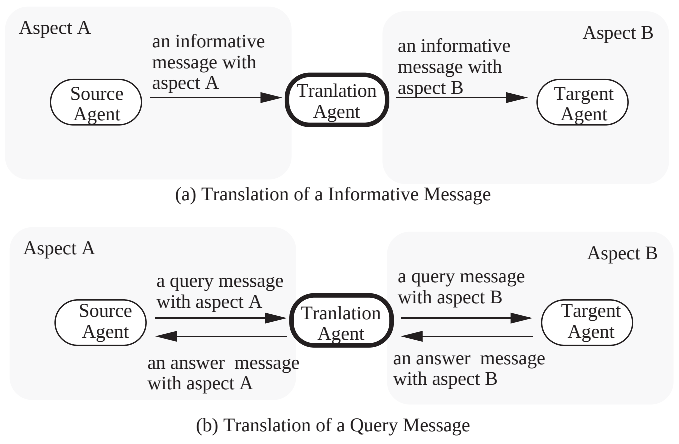
Aspect A
an informative message with aspect A
Source Agent
Tranlation Agent
Aspect B
an informative message with aspect B
Targent Agent
(a) Translation of a Informative Message
Aspect A
a query message with aspect A
Source Agent
an answer message with aspect A
Tranlation Agent
Aspect B
a query message with aspect B
Targent Agent
an answer message with aspect B
(b) Translation of a Query Message

Figure 9: Agent Communication with Multiple Aspects

4.4.2. Translation Procedure

The translation agent analyzes the given message based on the ontology and translation formulae among aspects.

Finding category aspects First, it analyzes the aspect of the given message and the aspect of the target agent to find the category aspects to join them. It collects all including aspects in these aspects by tracing `include` relations. An aspect is a category aspect to join them if it contains both an aspect in the included aspects of the message's aspect and an aspect in included aspect of the target agent's aspect. Then it retrieves translation formulae in these category aspects.

Identifying classes in the message The translation agent analyzes the message and identifies a class of each term in it. If class predicates (unary predicates) are used, classes of terms of their arguments are identified. Otherwise, classes of terms are identified by consulting definition of predicates and functions. Then it adds class literals for all terms to the message.

Applying translation formulae The message is modified by applying appropriate translation formulae. In case of informative messages, if a left hand side of a formula can match the message, the matched part of the message is replaced by the right hand side of the formula. In case of query messages, right hand sides of the formulae are applied. Binding of terms are preserved for translation of the answer message.

Removing unnecessary literals Literals which are not included in the target agent's aspect are removed.

Figure 11 shows how translation is applied to messages. Here we use aspects shown Figure 7 and two agent `temple/A` and `temple/B` use *temple/A* and *temple/B* aspects respectively. Then suppose that `temple/B` agent asks `temple/A` agent adult fee of the temple although `temple/A` agent believes that fee exists but that adult fee does not (see Figure 10).

In this example, a query message with aspect `temple/B` is expected to be translated into a message with aspect `temple/A` (see Figure 11(a)). Since aspect `temple/B` and `temple/A` use aspect `fee/B` and `fee/A` respectively (see Figure 7 (d)(e)), a category aspect `fee` including both aspects is retrieved (see Figure 7(a)). On the other hand, class definitions of terms are added to the message (see Figure 11(b)). The first translation formula (Line 6 to 10 in Figure 7(a)) is applied to the message and translated into one in Figure 11(c). Finally unnecessary literals are removed (see Figure 11(d)).

5. Related Work

Gruber proposed Ontolingua and discuss how ontology should be written [6]. His claim is that a good ontology can yield various formats in representation. The idea behind it is that there is a canonical conceptualization. For example, concept *timepoint* can be used to represent both year/month/day and "year season". But

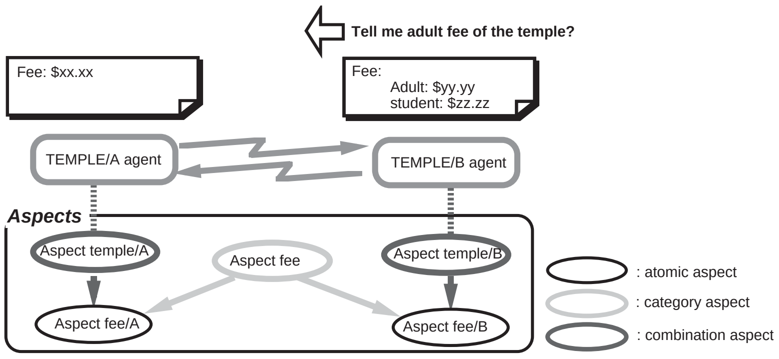

Figure 10: An Example of Communication between Agents

```
(temple ?x)
(name ?x ?y)
(fee ?x ?f)
(adult ?f ?f1)
(student ?f ?f2)
(fee-value ?f1 ?v1)
(fee-value ?f2 ?v2)
(< ?v1 500)
(< ?v2 400)
```

(a) The given message

```
(temple ?x)
(name ?x ?y)
(fee ?x ?f)
(adult ?f ?f1)
(student ?f ?f2)
(fee-value ?f1 ?v1)
(fee-value ?f2 ?v2)
(< ?v1 500)
(< ?v2 400)
(string ?y)
(temple-fee ?f)
(temple-fee-elm
               ?f1)
(temple-fee-elm
               ?f2)
```

(b) Adding class definitions

```
(temple ?x)
(name ?x ?y)
(fee ?x ?fee)
(fee-value ?fee
           ?value)


(< ?value 500)
(< ?value 400)
(string ?y)
(temple-fee ?fee)
(temple-fee-elm
              ?fee1)
(temple-fee-elm
              ?fee2)
```

(c) Applying a translation formula

```
(temple ?x)
(name ?x ?y)
(fee ?x ?fee)
(fee-value ?fee
           ?value)


(< ?value 500)
(< ?value 400)
(string ?y)
(temple-fee ?fee)
```

(d) Removing unnecessary literals

Figure 11: An Example of Translation

we do not believe that there *always* exists such canonical conceptualization and also it is a great effort to fix such conceptualization even if it exists. On the other hand, since we permit various ways of conceptualization of a single phenomena, we can write ontologies more naturally. In our approach, a phenomena is conceptualized as a network of some aspects each of which represents a way of conceptualization.

Guha proposed idea of *context* to deal with multiple theories [9]. He introduced "*(ist name-of-context formula)*" predicate to denote relationship among context. This predicate corresponds translation formulae in our approach. Although he showed several ways to use this predicates in logical inference, there is no unified way to deal with it. It is also a burden to embed the predicate in logical inference. We adopt translation approach in which interpretation of translation formulae are separated and invoked when they are needed. This approach is applicable if agent is poor in logical inference or even if non-logical.

6. Conclusion and Future Work

We discussed how ontology plays roles in building a distributed and heterogeneous knowledge-base system. Ontology is one of the minimum requirements for each knowledge-base system to join the community of knowledge-base system.

Ontology for a heterogeneous knowledge-base system should be heterogeneous because description according to perspective of each system should be allowed. Since we modeled ontology as combination of aspects each of which can represent a way of concepualization, our model of ontology allows heterogeneous and multiple descriptions for phenomenon in the world. Therefore it is appropriate as ontology of a heterogeneous knowledge-base system.

We also showed translation of messages as a way of interpreting multiple aspects. Combination of mediation and translation of messages makes it easy to build each coorprating system because it is required to have less knowledge on other systems.

Although our framework have these appropriate features for large-scale heterogeneous knowledge-base systems, the current system is not sufficient to realize flexible comminication among various heterogeneous agents In particular some issues are left to solve in mediation and translation. For example, message decomposition is needed to interpret complex messages. Description of translation rules should be improved because it requires much efforts to write rules. We need dynamic generation of rules (for example by learning) to realize flexible translation. We are planning to solve these problems in next work.

References

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

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

[3] M. Genesereth. Knowledge interchange format. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Second International Conference*, pages 599–600. Morgan Kaufmann, 1991.

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

[5] T. R. Gruber. Ontolingua: A mechanism to support portable ontologies. Technical Report KSL 91-66, Stanford University, Knowledge Systems Laboratory, 1992.

[6] T. R. Gruber. Toward principles for the design of ontologies used for knowledge sharing. Technical Report KSL 93-4, Knowledge Systems Laboratory, Stanford University, 1993.

[7] N. Guarion, M. Carrara, and P. Giaretta. Formalizing ontological commitments. In *Proceedings of AAAI-94*, pages 560–567, 1994.

[8] R. V. Guha. Representation of defaults in Cyc. In *Proc. AAAI-90*, pages 608–614, 1990.

[9] R. V. Guha. *Contexts: A Formalization and Some Applications.* PhD thesis, Department of Computer Science, Stadford University, Stanford, CA, 1991. (Available as Report No. STAN-CS-91-1399-Thesis).

[10] R. V. Guha and D. B. Lenat. Cyc: A midterm report. *AI magazine*, pages 32–59, Fall 1990.

[11] P. P. Nayak. Representing multiple theories. In *Proceedings of AAAI-94*, pages 1154–1160, 1994.

[12] T. Nishida. Towards integration of heterogeneous knowledge for highly autonomous analysis of dynamical systems — preliminary report from the PSX project. *Journal of Information Processing*, 15(3):350–363, 1992.

[13] T. Nishida and H. Takeda. Towards the knowledgeable community. In *Proceedings of International Conference on Building and Sharing of Very Large-Scale Knowledge bases '93*, pages 157–166, Tokyo, 1993. Japan Information Processing Development Center.

[14] R. S. Patil, R. E. Fikes, P. F. Patel-Schneider, D. McKay, T. Finin, T. R. Gruber, and R. Neches. The DARPA knowledge sharing effort: Progress report. In C. Rich, B. Nebel, and W. Swartout, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Third International Conference.* Morgan Kaufmann, 1992.

[15] H. Takeda and T. Nishida. Integration of aspects in design processes. In J. S. Gero and F. Sudweeks, editors, *Aritificial Intelligence in Design '94*, pages 309–326. Kluwer Academic Publishers, 1994.

[16] Y. Umeda, H. Takeda, T. Tomiyama, and Y. Yoshikawa. Function, behaviour, and structure. In J. Gero, editor, *Applications of Artificial Intelligence in Engineering V*, volume 1, pages 177–194, Berlin, 1990. Springer-Verlag.