Agent Communication with Multiple Ontologies

Hideaki Takeda, Kenji Iino, and Toyoaki Nishida
{takeda, kenji-i, nishida}@is.aist-nara.ac.jp
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 agent 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 *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 as ontology of a heterogeneous knowledge-base system. 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 make it easy to build each agent because it is required to have less knowledge on other agents.

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 [3, 5]. 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[6, 7] 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 how they are connected in the other agent. The other is, if they find relationship among their concept structures, it is difficult for each agent to interpret concept structure of the other agent.

We provide ontology 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 multi-agent architecture. In particular we identify relationship between agents and ontology. In Section 3, we show mediation of undirected messages as application of ontology. In this section, we assume a single ontology. In Section 4, we introduce *aspect* as component of ontology and define ontology as structure of aspects. 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. Agents in the Knowledgeable Community are required to have the following abilities;

Communication ability Each agent can communicate to other agents. It is a minimum 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 second point is realized by sharing ontology, i.e., each agent has part of ontology. Then other agents can guess relation between itself and the other agent by comparing their part of ontology.

In this paper, we restrict ontology as *frame ontology*, i.e., there are classes, relations, and hierarchy of classes. Figure 2 shows ontology of travel plan ontology used in our prototype system KC-Kansai.

Ontology is used when an agent is created and added to the Knowledgeable Community. First, ontology are consulted by agent programmers when they program new agents. If they find the same or the similar concepts to those that they intend to write, they can use these existing concepts in their agent programs. Second, when a new agent which introduces new concepts and their relations is added, the fragment of ontology is also added to the existing ontology.

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

1. accepts a pair of part of ontology and agent using it,
2. keep ontology consistent,
3. reply part of ontology associated to asked ontology,
4. reply part of ontology associated to asked agents,
5. reply a set of agents associated to asked ontology.

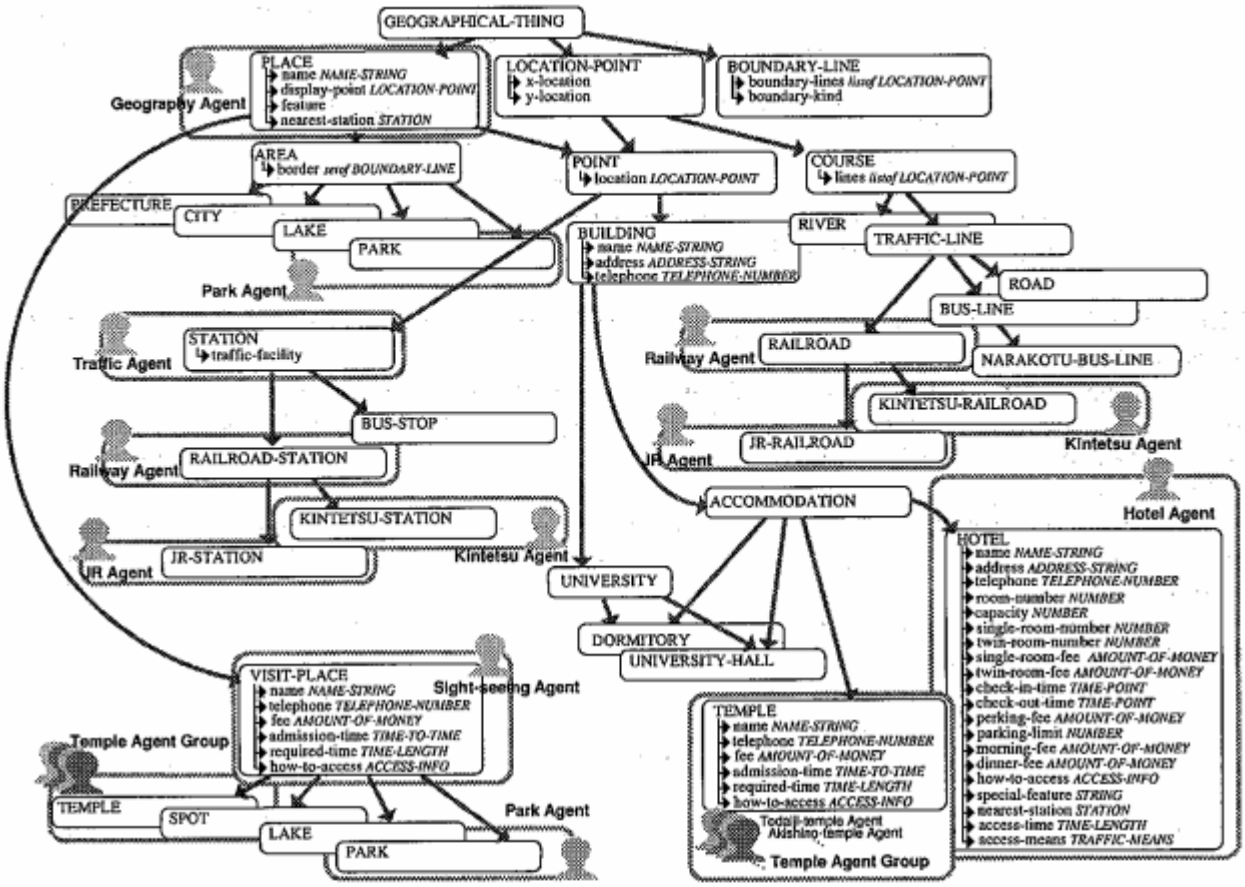


Figure 1: An Example of Ontology

3 Agent Communication with A Single Ontology

In this section, we assume a single ontology. Since only a single ontology exists 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. Though all agents agree to use a single ontology, they can not interpret all part of the ontology but can only deal with some part of the ontology. As we mentioned every agent has its domain, i.e., 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.

Since all agents are mapped into the ontology, we can determine appropriate agents for undirected messages by using the ontology.

As we mentioned, there is an ontology server that knows ontology and relations between ontology and agents. The ontology server 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, 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, 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. 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.

4.1 Aspects

There are many possible ways to describe even a single phenomenon. We should provide mechanism to allow different ways of conceptualization. We call a consistent view of conceptualization *aspect*. Every aspect provides us a framework to recognize some phenomena in the real world. We use various aspects to understand things in the real world. For examples, dynamics aspect provides concepts to understand how things move, and an aspect like *traffic aspect* to use train time-table book. The size of domain of

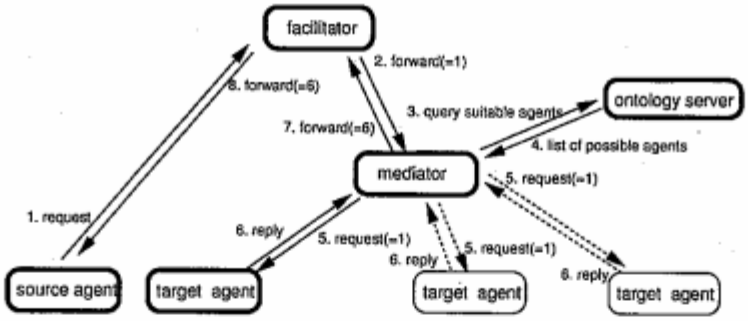


Figure 2: Mediation by Ontology

```
(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

aspects depends on their purposes. Scientific aspects are usually relatively large, because their purposes are to show wide consistent view for the real world. On the other hand, aspects in daily life are relatively small and not well organized to each other, because they are intended to provide a consistent view for specific scenes or conditions. Engineering aspects are seemed in-between.

If there are multiple aspects to conceptualize the same concepts, it is important to provide relations between them. Relations between aspects are not mandatory, but it enables to exchange information between aspects.

An ontology is defined as consistent combination of aspects each of which defines some concepts in the ontology. Multiple ontologies can be co-exist, and they can exchange information if they share the same aspects or some of aspects used in them have relations to each other.

4.2 Theory of Aspect

First we define atomic aspects, a background aspect, and composite aspects.

An atomic aspect is a minimum theory which defines a consistent perspective for modeling the world. An atomic aspect A consists of an aspect name $name(A)$, an *aspect predicate set* $R(A)$ a set of predicates and an *aspect theory* $T(A)$ a logical theory. An aspect theory $T(A)$ is a consistent set of logical formulae in which all predicates are either in $R(A)$ or $R(A_{bg})$ where A_{bg} is a background theory.

A background theory[1] A_{bg} consists of an *aspect predicate set* $R(A_{bg})$ a set of predicates and an *aspect theory* $T(A_{bg})$ a logical theory. An aspect theory $T(A_{bg})$ is a consistent set of logical formulae in which all predicates are in $R(A_{bg})$.

A composite aspect is an aspect which uses two or more aspects to define itself. There are two types of composite aspect, i.e, *combination aspect* and *category aspect*. The former is aspect in which two or more aspect theories are integrated into one. It means that concepts come from two different ways of conceptualization are used together. On the other hand, the latter is aspect in which two or more aspects are gathered for choice of interpretation. It means that one of those aspect theories is used to represent phenomenon in the world. It is used when there are some different types of conceptualization for the same phenomena. Both of composite aspects have inter-aspect theories which describe relationship among aspect theories. An inter-aspect theory for a combination aspect is just an additional theory to the integrated aspect theory, while one for a category aspect works as a mapping function to one aspect to the other.

Appendix shows a tentative formalization of aspect.

4.3 A Language for Aspects

Here we show a computational model of aspects, which is an extension of Ontolingua-like ontology definition (see [1]).

Definition of an atomic aspect consists of declaration of aspect name and definitions of classes, relations, and functions. Figure 4(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 4(b)(c))[2]. 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 4(a)). A left hand side of an implication formula

[1] It should be an atomic aspect, but we distinguish it just for convention.

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

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[3]. 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 ambiguousness 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 5).

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 6).

4.4.1 Function of Translation Agents

The basic function of a translation agent is as follows;

1. The translation agent receives a message from an agent (source agent) to a target agent via the mediator.
2. It analyzes the message and retrieves translation formulae from the ontology server.
3. 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.
4. It waits and receives an answer message from the target agent.
5. 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.

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.

[3]We discussed more direct use of multiple aspects in Ref. [8]

```
(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 4: Examples of Definition of Aspect

Figure 5: A Translation Agent in the Knowledgeable Community

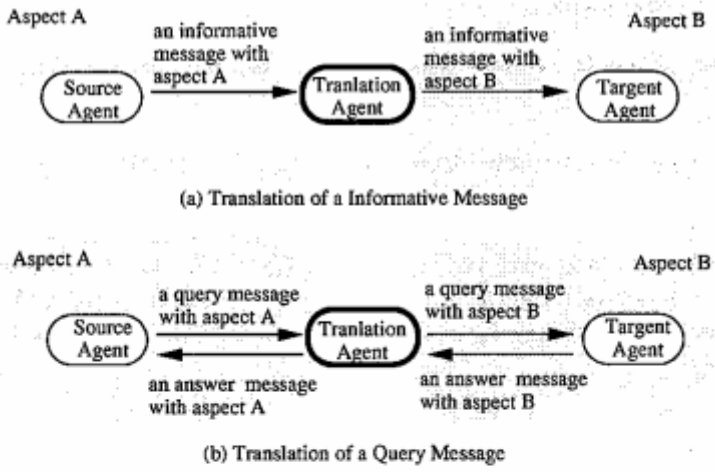
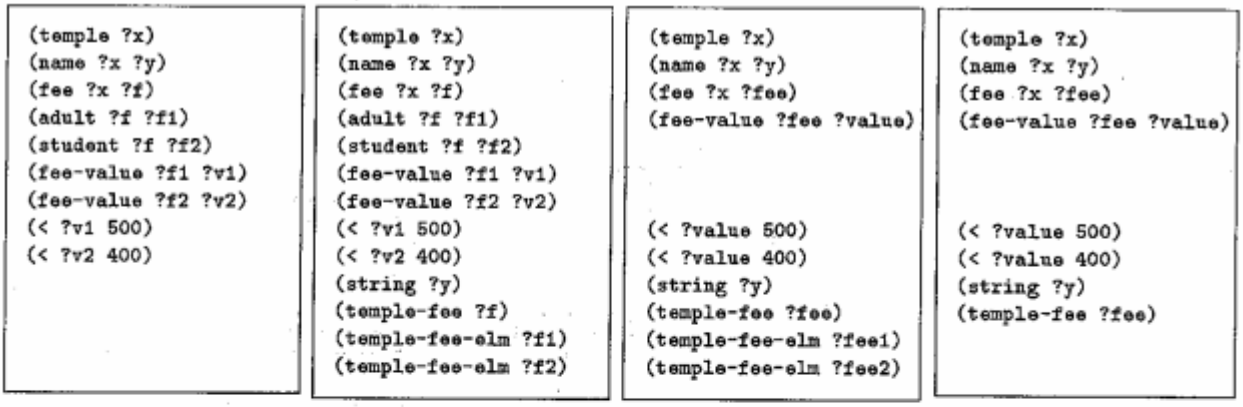


Figure 6: Agent Communication with Multiple Aspects



(a) The given message (b) Adding class definitions (c) Applying a translation formula (d) Removing unnecessary literals

Figure 7: An Example of Translation

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 7 shows how translation is applied to messages. In this example, a query message with aspect `temple/B` is expected to be translated into a message with aspect `temple/A` (see Figure 7(a)). Since aspect `temple/B` and `temple/A` use aspect `fee/B` and `fee/A` respectively (see Figure 4 (d)(e)), a category aspect `fee` including both aspects is retrieved (see Figure 4(a)). On the other hand, class definitions of terms are added to the message (see Figure 7(b)). The first translation formula (Line 6 to 10 in Figure 4(a)) is applied to the message and translated into one in Figure 7(c). Finally unnecessary literals are removed (see Figure 7(d)).

5 Related Work

Gruber proposed Ontolingua and discuss how ontology should be written [2]. 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 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 [4]. 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

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, 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 cooprating system because it is required to have less knowledge on other systems.

References

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

[2] 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.

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

[4] 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).

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

[6] 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.

[7] 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.

[8] 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.

Appendix: A Logical Framework for Aspect (tentative)

This is a tentative formalization of aspect just for clarification of discussion in Section 4.

Definition 1 *An atomic aspect A has a consistent theory $T(A)$ of a first language $L(A)$ and has a unique name $name(A)$*

Definition 2 *Aspect set $\mathcal{A}$ is as follows; $\mathcal{A} = \{A : A \text{ is an atomic aspect}\}$*

A unique name means that no other atomic aspect can use the same name. Here we assume every language for atomic aspects shares a domain of 'individuals'. To give a unique name for every predicate for every atomic aspect, another language is defined for an atomic aspect.

Definition 3 *$L(A)'$ for aspect A is a language in which is every predicate p_i in language $L(A)$ is replaced by $name(A).p_i$.*

An aspect theory is also translated into $T(A)'$.

Definition 4 *$T(A)'$ is a theory of language $L(A)$ which is every predicate p_i in language $L(A)$ is replaced by $name(A).p_i$.*

Then we can suppose a language to join all atomic aspects as follows

Definition 5 *L_0 is a first order language which contains every predicate of every atomic aspect $A \in \mathcal{A}$.*

Furthermore, we introduce L_0^m as modal extension of L_0. In the following discussion, we assume this language L_0^m, modal system S5, and use $T(A)'$ instead of $T(A)$.

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

A combination aspect is simply defined as follows.

Definition 7 $T(A_{COM}(A_1, A_3)) = T(A_1) \wedge T(A_2) \wedge I(A_1, A_2)$

$I(A_1, A_2)$ is an inter-aspect theory between A_1 and A_2. It can be false.

On the other hand, a category aspect is more complicated because it does not imply both of aspect theories are true at the same time. In order to represent a category aspect, we introduce modal operators $\Box$ and $\Diamond$ and assume S5 modal system. Then a category aspect for two aspects is define as follows.

Definition 8 $T(A_{CAT}(A_1, A_2)) = \Box(T(A_1) \vee T(A_2)) \wedge \Diamond T(A_1) \wedge \Diamond T(A_2) \wedge I(A_1, A_2)$.

Intuitively, the both theories can be true and either of them should be true at any time.

Definition 9 *An aspect A is included in aspect B if and only if $T(B) \vdash \Diamond T(A)$.*

Since we can use composite aspects as elements of composite 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}.

Theorem 1 *Aspect A_i is included in aspect A if $A = f(A_1, \ldots, A_i, \ldots, A_n)$.*

Definition 10 *Aspect A_1 and A_2 is compatible if and only if there exists an aspect $A = f(\ldots, A_1, \ldots, A_2, \ldots)$*

Definition 11 *Formula f is translative to aspect A if and only if there exists aspect B which includes f and is compatible with A.*