Ontology-supported Agent Communication

Hideaki Takeda, Kenji Iino, and Toyoaki Nishida
phone: +81-7437-2-5262, fax: +81-7437-2-5269
{takeda, kenji-i, nishida}@is.aist-nara.ac.jp
Graduate School of Information Science, Nara Institute of Science and Technology
8916-5, Takayama, Ikoma, Nara 630-01, Japan

Abstract

Building ontology is one of key issues for knowledge sharing which is important technology to realize large-scale knowledge-base systems. In this paper, we propose a computational framework for constructive ontologies called *aspect*. We first formalize aspect in a logical framework and then define it as a programming language.

Aspect is representation of conceptualization which consists of a vocabulary and a theory, and ontologies are built as composition of aspects. Two types of composition of aspects are provided, one is combination of aspects which is just union of aspects, and the other is category of aspects which links different but domain-sharing aspects. Using them, we can represent not only relations among different aspects but also a set of aspects which either of them can be used if needed.

We show a logical formalization of aspect by using modal logic. Category aspect is modeled using modal operator $\diamond$. We also formalize characteristics for aspects like *compatible* and *rigid* which can be used as criteria for ontology.

We also provide ASPECTOL, a programming language for aspect by extending Ontolingua. We then 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: knowledge representation, ontology, knowledge-sharing, context, model-based easoning.

Introduction

Large scale Knowledge base (VLKB) is indispensable to put AI theories to work in the real world. One approach for VLKB is *knowledge sharing* approach in which technology to share knowledge is proposed instead of VLKB system itself like Cyc project. Technology to share knowledge is summarized as environments for communication among participating systems. There three types of protocol for their communication. The first is a language to describe queries for exchanging knowledge and information, which is realized for example as KQML (Knowledge Query and Manipulation Language)(Finin *et al.* 1994). The second is a language to describe syntax of those queries, for which KIF (Knowledge Interchange Format)(Genesereth 1991; Genesereth and Fikes 1992) is proposed. The third is a language to describe semantics of those queries, which is called *common ontology* and Ontolingua(Gruber 1992) is proposed for this purpose. There are significant difference between the first two protocols and the last one, i.e., while the first two protocols are concerning syntactical aspect of messages and as a result just defining a format to communicate, the last one is committing contents of messages and therefore we have to provide not only a format but also ontologies themselves. Although building ontologies is crucial part of knowledge sharing, technology to build ontologies, in particular, large ontologies has not been established yet.

In this paper, we focus on technology to build ontologies for knowledge sharing. We propose *aspect* which is representation of conceptualization and ontologies can be build by composing aspects. There are two types of composition of aspects, one is combination of aspects which is just union of aspects, and the other is category of aspects which represents a set of aspects for alternatives. By using these types of composition of aspects, we can construct multiple ontologies which can co-exist at the same time.

Building ontology is how to describe complicated worlds in computer, and is recognized both in model-based reasoning and context-based reasoning recently, i.e, aspect corresponds *model* and *context* respectively. In model-based reasoning (for example (Falkenhainer and Forbus 1991)), the problem is how to select or how to integrate existing models. Since models are usually given in advance in traditional mode-based approach, they do not care about how to construct models. In the other hand, research on *context*(McCarthy 1993)(Guha 1991), they emphasize dynamics of context in reasoning. Although they discuss how to describe relations among context and how to lift rule to one context to the other, they do not mention how to prepare contexts. In

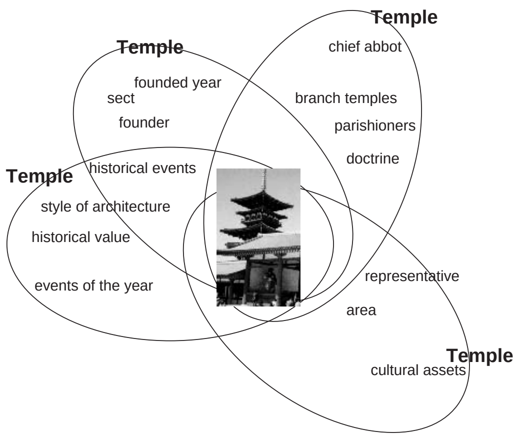

Figure 1: What is Ontology for Temple?

this paper, we discuss how to provide aspects, in particular *constructively.* To discuss consitituion of aspect would imply relationship among aspects and reasoning with aspects.

Our purpose is to propose a computational treatment for building ontologies based on aspect. It means that we should propose not only sound formalization of aspect but also environment to write and use ontologies by aspect. For this purpose, the following sections are organized as follows. We will first explain what is concept of aspect briefly in Section 2, then we will show a logical foundation of aspect which is based on modal logic in Section 3. In Section 4, we will propose a programming language ASPECTOL to describe aspects. In Section 5, we will show translation of messages between aspects to demonstrate aspects are used in knowledge-base systems. In Section 6, we will discuss other related work, and we will summarize the paper in Section 7.

Aspect

As we mentioned above, it is not easy task to build large common ontologies. One of the reasons is that we often confuse concepts from different conceptualization when we try to build ontologies.

For example, Figure 1 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 abbot are concepts related to "temple". Mixture of concepts from different conceptualization often confuse us because 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 have and use various aspects, for example in engineering we use aspects like dynamics aspects, kinematics aspects. To model the commonsense world, we may use aspects like traffic 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 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.

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.

Two aspects can share aspects in their consititution, or be connected by category aspects. In such case, 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.

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 *name* of L_E. L is a first-order language which is the same to L_E except predicate *name* is removed.

Aspect Theory

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

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

For convenience of discussion, we define $T'(A)$ of L_E as follows;

$$T'(A) = T(A) \wedge name(A)$$

$name(A)$ works as identifier of aspect which has a similar effect to the second argument of predicate `ist` in Ref. (Guha 1991), and a modal operator in Ref. (Nayak 1994).

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

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 as follows;*

$$T(A_{COM}(A_1, \ldots, A_n)) = \\ T'(A_1) \wedge \ldots \wedge T'(A_n) \wedge I(A_1, \ldots, A_n)$$

And it must be consitent.

$I(A_1, \ldots, A_n)$ is an inter-aspect theory among $A_1, \ldots A_n$. It can be true.

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. 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 $\Box$ and $\Diamond$ and assume S4 modal system. Then a category aspect for two aspects is define as follows.

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

$$T(A_{CAT}(A_1, \ldots, A_n)) = \\ \Diamond T'(A_1) \wedge \ldots \wedge \Diamond T'(A_n) \wedge I(A_1, \ldots, A_n)$$

And it must be consitent.

$I(A_1, \ldots, A_n)$ 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}.

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 $T'(B) \vdash \Diamond T'(A)$.*

Definition 5 *An aspect A is* ***strictly included*** *in aspect B if and only if $T'(B) \vdash T'(A)$.*

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. (Nayak 1994).

Similarly, relations between formula and aspect are defined.

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

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

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.*

Theorem 2 *If aspect A_i is included in aspect A, then $A = f(A_1, \ldots, A_i, \ldots, A_n)$, where function f is composed from A_{COM} and A_{CAT}.*

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

Definition 8 *Aspect A and B is* ***compatible*** *if A and B is the same aspect or there exists aspect C which has both A and B as componet or 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.

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****.*

$$I(A_1, \ldots, A_n) \vdash \Box(T'(A_1) \vee \ldots \vee T'(A_n))$$

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 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 A_1 is strictly included in A.*

Actually relation between A and A_1 is stronger, i.e., $T'(A) \vdash \Box T'(A_1)$. 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 $T(A) \vdash \Box f$.*

Theorem 4 *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.(Guarion *et al.* 1994).

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****.*

$$I(A_1, \ldots, A_n) \vdash \bigwedge_{k=1,\ldots,n-1} \bigwedge_{l=k+1,\ldots,n} \neg\Diamond(T'(A_k) \wedge T'(A_l))$$

Theorem 5 *Suppose a exclusive category aspect A and its componet aspects $A_1, \ldots, A_i, \ldots, A_n$. If A_i is included in A, then*

$$T'(A) \wedge \bigwedge_{k \neq i} \Box\neg name(A_k) \vdash T'(A_i)$$

It means that aspect A_i is strictly included in aspect A if all $name(A_k)(k \neq i)$ are always false.

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;

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

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

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

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

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

Theorem 6 *If a proposition p is equivalent in all componet aspects of a compact category aspect A, p is rigid in A.*

```
(define-category-aspect FEE
  (:use fee/a fee/b)
  (:category-type nil)
  (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))))))
    ...
 )
```

(a) Category Aspect **`fee`**

```
(define-aspect temple/A (TEMPLE)
  (:use fee/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`**

Figure 2: Examples of Definition of Aspect

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 (Gruber 1992)).

Definition of an atomic aspect consists of declaration of aspect name and definitions of classes, relations, and functions.

Definition of a combination aspect includes the same style of definition of an atomic aspect and declaration of component aspects and renaming of predicates. Figure 2(b) is an example of combination aspects.

Definition of a category aspect consists of a set of translation formulae, declaration of component aspects, declaration of renaming of predicates, and declaration of category type. A translation formula is defined between two component aspects in a category aspect, and is defined as `define-translation` which describes logical relation between concepts in both aspects. Figure 2(a) is an example for it. 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. This formulae correspond object-level descriptions in inter-aspect theories which we have discussed in Section 3.4. Declaration of category type is currently either nil or exclusive or compact which we have introduced in Section 3.3.

Translation between Aspects

In order to show how defined aspects work as ontologies, we realized translation of formulae between different aspects. As we mentioned in Section 1, our interest is to provide knowledge sharing technology, especially with heterogeneous agents (participating systems). It is natural that different agents use different aspects. In our definition, if these aspects are compatible, these agents may be able to exchange information, otherwise not. We realized this exchanging procedure as translation agents in our agent-based knowledge-base system Knowledgeable Community(Nishida and Takeda 1993).

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.

The translation agent analyzes the given message based on the ontology and translation formulae among aspects. The translation Procedure is as follows;

1. 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 link them. It collects all included aspects in these aspects by tracing `use` relations. An aspect is a category aspect to join them if it contains both one of included aspects of the message's aspect and one of included aspects of the target agent's aspect. Then it retrieves translation formulae in these category aspects.
2. 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 in the aspect. Then it adds class literals for all terms to the message.
3. 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.
4. Removing unnecessary literals Literals which are not included in the target agent's aspect are removed.

During the procedure, the translation agent behaves as agent which can understand a category aspect linked to both aspects in the source and the target agents.

Figure 3 shows how translation is applied to messages. Two agent **temple/A** and **temple/B** use `temple/A` and `temple/B` aspects respectively. `Temple/A` aspect uses `fee/A` aspect, and `temple/B` aspect uses `fee/A`. `Fee` category aspect uses `fee/A` and `fee/B` aspects.

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.

```
(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)



(< ?v1 500)
(< ?v2 400)
(string ?y)
(temple-fee ?f)
(temple-fee-elm ?f1)
(temple-fee-elm ?f2)
```

(c) Applying a translation formula

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



(< ?v1 500)
(< ?v2 400)
(string ?y)
(temple-fee ?f)
```

(d) Removing unnecesary literals

Figure 3: An Example of Translation

In this example, a query message with aspect `temple/B` is expected to be translated into a message with aspect `temple/A` (see Figure 3(a)). As aspect `temple/B` and `temple/A` use aspect `fee/B` and `fee/A` respectively, a category aspect `fee` including both aspects is retrieved (see Figure 2(a)). On the other hand, class definitions of terms are added to the message (see Figure 3(b)). The first translation formula (Line 8 to 12 in Figure 2(a)) is applied to the message and translated into one in Figure 3(c). Finally unnecessary literals are removed (see Figure 3(d)).

Related Work

Gruber proposed Ontolingua and discussed how ontology should be written (Gruber 1993). 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 phenomenon, we can write ontologies more naturally. In our approach, a phenomenon 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 (Guha 1991). He introduced "`(ist name-of-context formula)`" predicate to denote relationship among context. The advantage of this approach is `ist` is also a first-class object. Since formulae using `ist` predicate and other normal predicates together are allowed, various complicated situations with context can be represented. In our terminology, it is useful to describe object-level relations of aspects. It is difficult to describe aspect-level relations, e.g., to compare theories or to compose theories. Another problem is that his theory is unclear from a computational view. Although he presented various ways of inferences with context like default reasoning and temporal reasoning, each solution for them are isolated and not integrated. It is because concept of context is too wide and vague yet. On the other hand, aspect is clearly defined in a logical framework. And then a specific language for a specific usage is proposed to realize part of such logical definition, for example ASPECTOL is a language for sharing ontologies with multiple agents. Therefore its capability is clearly defined.

Nayak(Nayak 1994) proposed another approach to represent multiple theories. He also used modal logic and each modal operator presents a micro theory. It provides simple but logically sound relation among theories. They discussed inter-theory relations like *weaker-than*. But inter-theory relations they mentioned are too simple, i.e., limited to those for comparison, therefore it is difficult to describe composition of aspects we have discussed here.

Wiederhold(Wiederhold 1994) proposed Domain-Knowledge-Base(DKB) algebra like *DKB-Intersection*. He discussed how to integrate schemes when they share some concepts and realized it as above algebra. It is the same problem we identified as category aspects. Since his solution is limited to scheme integration, our approach gives more general solution, for example we can use category as aspect again.

Conclusion

In this paper, we have developed a computational framework for constructive ontologies as aspect. A key feature of aspect is compositionality, i.e, it can be composed by other aspects. We provide combination and category aspects for composition, and then we can construct large ontologies as aspects composed of other smaller aspects.

Category aspect is a unique idea which links different but domain-sharing conceptualizations. By category aspect, we can represent not only relations among different aspects but also a set of aspects which either of them can be used if needed. Logically category aspect is modeled using modal operator $\diamond$. It means that each component aspect is possible, not always true. According to introduction of modal operators, we also introduced two types of interpretation of which difference is whether possible aspects should be taken into account. We also showed some theorems, e.g., how possible aspects can be always true.

We have also provided a programming language for aspect by extending Ontolingua. We have shown how aspects described in this language is used in the agent-based knowledge-base system. We have been developing VLKB systems with the aspect language as common ontology description language where mediation and translation is supported by using aspect.

References

Falkenhainer, B. and Forbus, K.D. 1991. Compositional modeling: Finding the right model for the job. *Artificial Intelligence* 51.

Finin, Tim; McKay, Don; Fritzson, Rich; and McEntire, Robin 1994. KQML: An information and knowledge exchange protocol. In Fuchi, Kazuhiro and Yokoi, Toshio, editors 1994, *Knowledge Building and Knowledge Sharing.* Ohmsha and IOS Press.

Genesereth, M.R. 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.

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

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

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

Guarion, Nicola; Carrara, Massimiliano; and Giaretta, Pierdaniele 1994. Formalizing ontological commitments. In *Proceedings of AAAI-94.* 560–567.

Guha, Ramanathan V. 1991. *Contexts: A Formalization and Some Applications.* Ph.D. Dissertation, Department of Computer Science, Stadford University, Stanford, CA. (Available as Report No. STAN-CS-91-1399-Thesis).

McCarthy, J. 1993. Notes on formalizing context. In *Proceedings of IJCAI-93.* 555–560.

Nayak, P. Pandurang 1994. Representing multiple theories. In *Proceedings of AAAI-94.* 1154–1160.

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

Wiederhold, Gio 1994. Interoperation, mediation, and ontologies. In *FGCS 94 workshop on Hetrogeneous Cooperative Knowledge-Bases.* ICOT. 33–48.