Ontological Mediation: An Analysis

—

DRAFT

Alistair E. Campbell, Hans Chalupsky, and Stuart C. Shapiro
Department of Computer Science
and Center for Cognitive Science
State University of New York at Buffalo
226 Bell Hall
Buffalo, New York 14260
{aec,hans,shapiro}@cs.buffalo.edu

Feb. 1, 1995

1 Introduction

The central objective of the Knowledge Sharing Initiative [Neches *et al.*, 1991] is to develop techniques, tools, languages, protocols, standards, etc., in order to enable independent and possibly heterogeneous knowledge bases or agents to mutually share their knowledge and expertise. There exists a wide spectrum of possible approaches to this problem. One extreme position in this spectrum might be described as the *standardization approach*, which tries to develop a set of standard languages, ontologies and protocols, which are then to be used by every knowledge agent. Communication between such agents is then relatively simple, because they all speak the same language, use the same conceptualizations and follow the same protocols. The disadvantage of this approach is that it is only applicable to knowledge agents which were developed with these standards in mind; agents that were already existing prior to these standardizations are left out of the picture. In addition, having to conform to a rigid set of guidelines in agent design can be stifling. Doing so reduces every agent's abilities to a predetermined lowest common denominator. For example, if the only language permitted in interagent knowledge communication is KIF [Genesereth and Fikes, 1991], then the agents will be restricted to communicating in only first-order sentences. The other extreme position in the spectrum of approaches can be called the *mediation approach*. It tries to assume as little as possible about the various knowledge agents, while enabling communication between them by providing a special class of agents called *Mediators* or *Facilitators*. These mediators normally speak the language of one particular knowledge agent as well as a common mediator language that lets them easily communicate with mediators of other knowledge agents. The advantage of the mediation approach is that it is applicable to a wide variety of already existing knowledge agents, The main disadvantage is that it involves possibly difficult translation at various levels.

Our main interest lies in the mediation approach, and in this paper we investigate a subproblem thereof. Successful communication between two knowledge agents (or any agents for that matter)

has a wide variety of prerequisites. One is that the agents use compatible communication protocols. The protocol aspect of communication encompasses establishing a communication channel, deciding on a content language, using the proper speech acts, e.g., assertion, query, etc., transmitting actual information, synchronization, error detection and recovery, etc. The Knowledge Query and Manipulation Language, KQML [Finin and others, 1992], is mainly concerned with that part of the communication between knowledge agents. Another important prerequisite of successful communication is that the exchanged content messages have the proper meaning so that they are understood correctly by the intended recipient. To achieve this, the language of one agent has to be translated into the language of the other. A form of translation — perhaps better called explanation — might be necessary even if the agents speak the same language, because they might have different expertise, use different terminology, etc.

To facilitate that translation part of communication we are investigating the notion of an *ontological mediator* (OM), and the feasibility of implementing a computerized OM. An *ontological mediator* is an agent that enables communication among two or more intelligent agents who either speak different languages or use different ontologies. Unlike KQML mediators who treat the messages of the agents as uninterpreted strings, OMs are to involve themselves in the meanings of the messages being sent among agents.

We need mediators *because* there is no common framework within which the community is developing knowledge agents. The interaction between specialized knowledge agents and users presupposes that the users already understand the meta-language required for knowledge queries, and will completely understand the responses they receive. When a human user doesn't understand a response, they will issue a new query in an attempt to gain clarification. Automated interaction between autonomous knowledge agents, however, was never intended. Mediators bridge the gaps between agents. They determine what clarifying questions to ask. They rearrange the structure of queries and responses to smooth out inconsistencies and prevent miscommunications.

Wiederhold justifies the use of mediators (called SoD's) this way [Wiederhold *et al.*, 1990]:

> Today, without the knowledge encoded in SoDs, the methods for retrieving the *best* information are explicitly specified by the user. It is likely to require distinct methods for multiple domains. Both in database and Prolog access styles, these specifications require knowledge of each the [sic] underlying domains and their structure. In today's database languages a sensible specification is likely impossible to state, so that all the data has to be retrieved into memory, and then processed and reduced by the application programs. (p. 67)

2 What is an Ontology

The primary task of an OM is to mediate between agents using different ontologies, so, what are these ontologies? Or better, what do we mean when we say *ontology*? Let us start out with a few definitions from the literature and then give our definition of it.

From Webster's on-line dictionary:

> on.tol.o.gy noun [NL ontologia, fr. ont- + -logia -logy] 1: a branch of metaphysics relating to the nature and relations of being 2: a particular theory about the nature of being or the kinds of existence.

This definition characterizes ontology the way it is standardly used in philosophy.

In artificial intelligence (AI) circles people are less concerned with the actual nature of existence or reality than with the *modeling* of certain aspects of that reality. They use the term more along the lines of Tom Gruber, one of the principle designers of Ontolingua. He writes [Gruber, 1994, p.1]:

> An **ontology** is an explicit specification of a conceptualization. The term is borrowed from philosophy, where an Ontology is a systematic account of Existence. For AI systems, what "exists" is that which can be represented. When the knowledge of a domain is represented in a declarative formalism, the set of objects that can be represented is called the universe of discourse. [...] Formally, an ontology is the statement of a logical theory.

Gruber's characterization of ontology qua logical theory actually subsumes more than what AI researchers usually consider as part of an ontology. In his view an ontology consists of a representational vocabulary with precise definitions of the meanings of the terms of this vocabulary *plus* a set of formal axioms that constrain the interpretation and well-formed use of these terms.

Most commonly however, the representational vocabulary is the only aspect considered to be part of an ontology. Such representational vocabularies are usually defined as *taxonomic class hierarchies*. For example, the taxonomy of a general natural language understanding system would have very general classes such as *abstract things* and *concrete things* at the top of the hierarchy, bottoming out at specific common nouns, verbs and adjectives. The ontology, together with the specific individuals of each taxonomic class, constitute the agent's knowledge base. The constraint axioms omitted from the specification of the ontology are added to the AI system separately.

In the following we will use the term ontology in this narrower sense. Hence, for us the primary function of an OM is to translate between differing taxonomies of communicating agents. One of the foci of our future research in this area will be to incorporate semantic constraints in interagent reasoning, as it would seem necessary if we wish to have agents understand each other's meanings.

3 A comparison of ontologies

A number of computer-based ontologies are used by existing knowledge agents to taxonomize the world and restrict the kinds of assertions that can be made about objects. We expect that communicating agents may use ontologies that differ. However, since any two rational agents who live in the real world have the common experience of real world, we expect to find that ontologies describing the same portion of reality will have significant overlap. We have collected a few ontologies from various sources in order to determine whether the extent to which ontologies overlap meets with our expectations.

3.1 Integrating ontologies

In building interlingua for communicating agents, one approach is to merge separate ontologies into one ontology that is consistent with both agent's world view. In order to determine whether this is feasible, we must examine the structure of ontologies. If they are identical except for concepts at the bottom of the taxonomy,

The WordNet lexical database system provides relational information about synonym groups (*synsets*). The synsets for nouns are organized in a hierarchy whose top starting with {"entity"}.

The Penman [Penman, 1989] system is designed to support natural language understanding. Part of this system is the Penman Upper Model, a LOOM [MacGregor, 1988] taxonomy that drives

the natural language generation engine. Similarly, the CYC knowledge base [Lenat and Guha, 1989] is an ontology designed for general-purpose artificial intelligence systems. Since

We compare the upper levels of these three systems. Figure 1 illustrates the very top levels of the CYC and Penman system. In CYC, the top levels are in a relatively sparse tangled hierarcy. The Penman system, on the other hand, employs a dense tree. Notice also that there is very little obvious overlap in top-level concepts between CYC and Penman. Clearly, these are two orthogonal ways of taxonomizing the world.

It is not necessary for ontologies to match at every level, however. If two agents are discussing technical issues, the vocabulary will remain at the lower levels of their respective ontologies. It does not matter how the two agents partition their concepts as long as they mean the same things by use of particular concepts in dialogue.

Thus, a top-down approach to ontological augmentation, where agents discuss the classification of a concept beginning with the most general and abstract items in their taxonomies, will not always succeed because of a lack of a common frame of reference at that level. By the same token, a bottom-up approach, is not guaranteed either. Instead, if a common frame of reference can be found nested in both ontologies, an iterative graph traversal algorithm (growing both up and down) is the most successful heuristic. Our algorithm employs this approach by finding immediate subclasses and immediate superclasses of a noun concept, then expanding the search by part-whole relationship information.

In WordNet, however, The top level, "entity," is divided into ten synsets, each of which has multiple words. The same word may belong to more than one sense, actually occupying multiple places in the ontology. Wordnet is much more dense and tangled hierarchy than both CYC and Penman. We omit a diagram of its top level for that reason.

Many approaches to the problem of heterogeneous knowledge sources involve merging two or more sources of knowledge into one large knowledge base. This is also known as building an interlingua. A major obstacle to building an interlingua is the task of translating each source into the knowledge representation language to be used in the interlingua. When ontologies are merged, it is often done by hand [Knight and Luk, 1994] It is not yet clear how to automate this process in the general case. Even if ontologies are structurally identical but use different words, there may not be direct translations between words at the same level of the ontology. The spreading search heuristic may lead to a solution for particular concepts. In addition to this basic paradigm we also require a model of agent-mediatior-agent *explanation dialogue.*

Another problem related to merging is that of making use of common on-line resouces, including machine-readable dictionaries. Given dictionaries such as the Longman's Dictionary of Contemporary English (LDOCE), or Roget's International Thesarus (RIT), one may want to build up a lexicon that can be used by natural language understanding systems. Such a project has been attempted by [McHale and Crowter, 1994]. In order to build a complete lexicon, certain thematic roles for words have to be extracted from the dictionary. However, these roles are not explicity represented in the LDOCE. A pattern matching heuristic is employed on sample sentences in the LDOCE to determine the ways words are used in the context of actual sentences. It can then be determined, for example, which verbs require direct or indirect objects. Additionally, their algorithm will attempt to correlate word senses between the LDOCE and RIT, essentially merging concepts from different sources and representing their common meanings. When a mediator understands and represents explicitly the *meanings* of words, it is able to make proper translations.

Must translations be exact? Except in very precise and technical areas, the most intelligent of agents—humans—do not always mean the exactly the same thing, even when they use the same word. However, humans share enough of the meanings of words to allow them to communicate

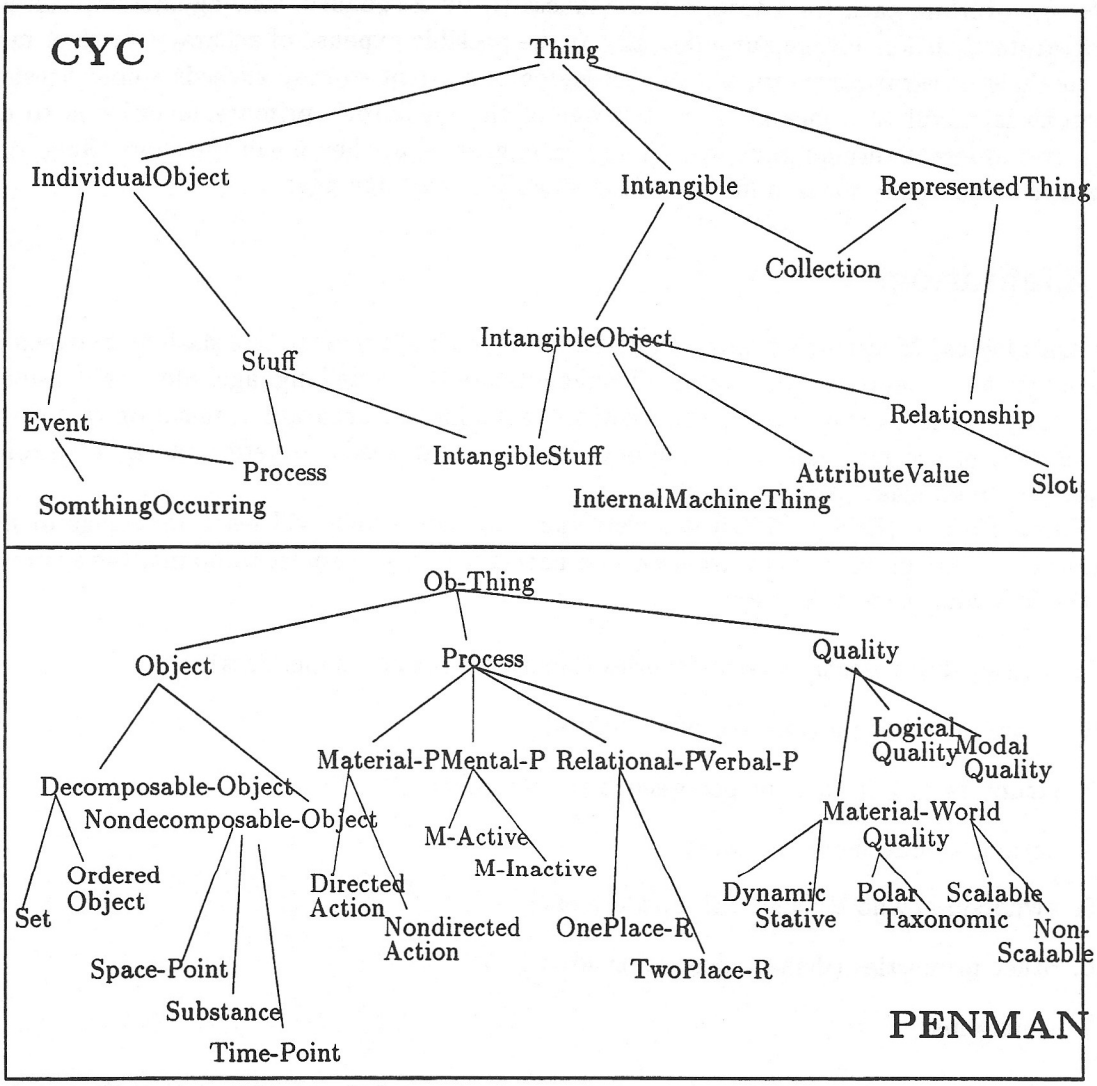

Figure 1: A comparison of ontologies

effectively, and understand *enough.* One theory that describes this phenomena is set forth in [Lehmann and Cohn, 1993]. Every concept is described by a two-part system, the EGG, which formally defines the concept, and the YOLK, which is a set of core exemplars, either formally defined, or enumerated. In order for effective communication to occur about a particular EGG/YOLK concept, both agents must agree on the YOLK, or agree to disagree about the meanings of the terms they are using. In addition, the EGG/YOLK formalism allows for a much finer distinction between concepts. There is a partial order of 46 distinct ways in which the EGGS and YOLKS of two concepts can overlap, each from total disjointedness to total equality. The agents can therefore work to integrate their knowledge automatically at the possible expense of accuracy, or they may choose not to allow integration except where the degree of concept overlap exceeds some threshold. This approach is useful to a mediator because one of the mediator's primary facilities is to determine when two different terms match well enough in context that they mean the same thing, or when to identical terms really mean different things to each knowledge agent.

4 Definitions

The Ontological Mediator's primary function is to provide accurate translations between sentences of one agent and sentences of another. The agents use the same language, but might use open-class words unfamiliar to each other. The key to providing an accurate translation is to understand the content of the message. Most importantly, an agent needs to determine a definition for each unfamiliar open-class word it hears.

Karen Ehrlich [Ehrlich, 1994] is developing a system which will learn meanings of new words from the context in which they are used in a narrative. With respect to nouns, the system searches for the following narrative cues:

1. relationship to basic-level categories (identity, subclass, superclass)
2. function (what purpose the noun serves)
3. structure (what parts or possessions the noun has)
4. actions (what the noun does)
5. ownership (who or what can own a noun)
6. other properties (default size, color, etc...)
7. synonyms

Rather than read a narrative, an ontological mediator can ask questions of its agents. Given a new word, its task is to find a place for it in the agent's ontology. The mediator needs to ask the right questions in order to make this placement correctly.

Following the work of Ehrlich, we have begin to develop a system which interfaces with the WordNet ontology and asks the user questions about a given noun. It asks the user to specify class inclusions and part-whole relationships about the word. It then displays possible words the given noun might match. If the user verifies a match, the system will place the new word in the corresponding synset, and update its internal database with this new information. Currently, our interface to WordNet is read-only, but even if it were read-write, we don't believe it is the mediator's job to change the ontologies of its agents directly. If a word match cannot be found, but the user

is satisfied that the given noun is a sibling of the hyponyms of a WordNet word, the system will modify its internal database with a new synset of that sense containing just the new word. After this system is more fully developed, future queries of the system regarding the given noun will reveal its placement in the taxonomy.

The question/answer approach to determining the ontological status of a new word places a great deal of trust in the user's (or question-answering agent's) competence. For example, the agent must know the meaning of each word the mediator presents; otherwise important clues as to the new word's placement might go unnoticed. In our informal experiments to date, we have discovered that the user must almost know *a priori* the classification of sibling words in the destination synset. Otherwise, the user may cause the mediator to digress in a direction that ultimately fails to classify the new word. The user needs to be intelligent enough to realize when such a digression has occurred and to tell the mediator to back up to a point before the digression began.

5 An Ontology of Problems

Suppose we had to design an OM for the mediation between two particular software agents. What sort of problems would we encounter? Here is a first categorization:

Translation problems center around the translation of sentences from the language of one agent into that of another.

Protocol problems center around how the OM is bound into a dialogue. Should it always translate or only when it detects misunderstanding? How can it lead a clarification dialogue? How does it inform the agents about translation problems? Etc.

Design problems center around the general architecture of the OM. How much does it have to know to be useful? Does it have to know everything in advance? Should it be able to learn?

Below we will investigate each of these categories in more detail.

5.1 Translation Problems

The study of language is usually concerned with *syntax*, *semantics* and *pragmatics*. If one wants to translate from one language into another one is confronted with problems along all those dimensions. Let us assume for the moment that our OM has to mediate between two agents who use a formal language with the same syntax, e.g., the language of first-order predicate calculus, and that pragmatic aspects are of minor concern, e.g., the only speech acts used by the agents are *tell* and *ask*. Hence, "all" our OM has to be concerned with is the semantics of individual utterances i.e., questions and assertions. Another way to characterize these assumptions is to say that we are confronted with a *dialect problem*, that is both agents use the same basic sentence structure, but some of their words mean different things, or one agent uses words not known by the other agent.

Let us further assume that both agents conceptualize their domain in ontologies consisting of individuals, classes of individuals, properties of individuals, and relations between individuals (i.e., both agents use a first-order conceptualization).

In such a first-order conceptualization we usually write sentences such as `Apple(ap1)`, or `Red(ap1)` to make assertions about the domain under consideration. If we have two different conceptualizations C_1 and C_2 of a particular domain, with languages L_{C_1} and L_{C_2}, and we want to translate from one conceptualization into the other, we need a translation function $T : L_{C_1} \rightarrow L_{C_2}$ such that (ideally) for every sentence $s_1 \in L_{C_1}$ we have $T(s_1) = s_2$ where s_2 has the "same meaning"

as s_1. The quotes should indicate that it is not immediately obvious how to define our intuitions about "same meaning". Intuitively, s_1 and s_2 should express the same concept (to use another vague term), however, the denotations or meanings of sentences in a standard first-order logic are truth values, and two sentences s_1 and s_2 are generally not semantically equivalent just because they have the same truth value. Moreover, in general it will not be possible to map sentences from one conceptualization to another in a 1-to-1 fashion, rather it might take several sentences in one language to express the same meaning as one sentence of another language.

If we were to build an OM for two particular knowledge agents, then one of our primary concerns would be the definition of a translation function T for that situation. The task of T is to translate between the different conceptualizations used by the individual knowledge agents. Before we go on to attempt a categorization of a variety of differences between conceptualizations, we want to make the following (somewhat obvious) observations:

Domain overlap: Only if the domains conceptualized by the two agents *overlap* do we have a translation problem. Put differently, only if there is a set of sentences (or assertions, or beliefs) of agent A that "talk about the same thing" as a set of sentences (or assertions, or beliefs) of agent B might we have to translate between these sets of sentences. If all agent A knows about are apples and oranges, and all agent B knows about are hammers and sickles, then there is obviously nothing for them to communicate about.

Domain difference: Only if the domains conceptualized by the two agents *differ* do we have a translation problem. If agent A conceptualizes the exact same aspects of the domain as are conceptualized by agent B, then, even if they conceptualize these aspects differently, there is no need for them to communicate because nothing new could result from such communication.

There are situations where even without domain difference there is need for communication, for example, if one agent just wants to delegate some work to some other agent even though it could do it all by itself.

5.1.1 Categorizing Differences

In the following let us assume that we have two knowledge agents with non-identical but overlapping conceptualizations of a domain, that is, according to our observations above translation might actually be necessary. What categories of differences in the two conceptualizations are we likely to encounter?

Naming differences: The two conceptualizations might use different names for the same domain object (synonyms), or the same name for different objects (homonyms). Domain objects might be individuals as well as functions, classes and relations. This is different from the categorization given in [Goh *et al.*, 1994] where differences between relation or attribute names of a database schema are categorized as *schematic* while differences between instance names (values) are categorized as *semantic*.

Structural differences: Structural differences are syntactic differences such as different order of arguments in a relation, or encoding of a domain object in the relation name, for example, `GreaterThanApple1`.

Taxonomic differences: Knowledge agents often structure a domain with help of taxonomic or class hierarchies. Such hierarchies might differ, for example, in the class hierarchy of Cyc

[Lenat and Guha, 1989] `Person` is a subclass of `Process` which is rather unusual and likely to differ from many other taxonomies.

Different expertise: An expert knowledge agent might have a much richer domain model than another agent, making it difficult to translate the expert terminology into something understandable by the "lay" agent. On the other hand this very expertise is probably the reason why a lay agent might want to communicate with the expert.

Different expressiveness: The representation language employed by one agent might be more expressive than that of another, for example, an agent using the language of first-order predicate calculus will be able to use universal quantification, while a relational database agent will not be able to do that.

Different definitional constraints: For example, one agent might define a `bachelor` as an unmarried man, while another might define it as an unmarried man who is not a celibate priest.

More... :

The categories above are not mutually exclusive, for example, instances of different definitional constraints might as well be characterized as a taxonomic differences, or as naming differences where two different domain objects or concepts have the same name. Moreover, differences of conceptualizations will usually be a colorful mixture of all of the above.

5.1.2 A Simple Example Domain

To make our investigation slightly more concrete, we turn to a very simple example domain which is shown in Figure 2.

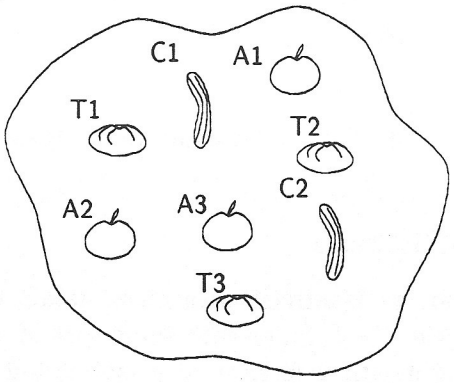

Figure 2: Veggie World: A simple example domain

5.1.3 Naming Differences

Figure 3 illustrates naming differences between two possible conceptualizations of Veggie World.

Conceptualization 1	Conceptualization 2
`Apple(ap1)`	`A(i1)`
`Apple(ap2)`	`A(i2)`
`Tomato(to1)`	`T(i3)`
`Tomato(to2)`	`T(i4)`

Figure 3: Two simple conceptualizations

5.1.4 Taxonomic Differences

Figure 4 illustrates taxonomic differences between two possible conceptualizations of Veggie World.

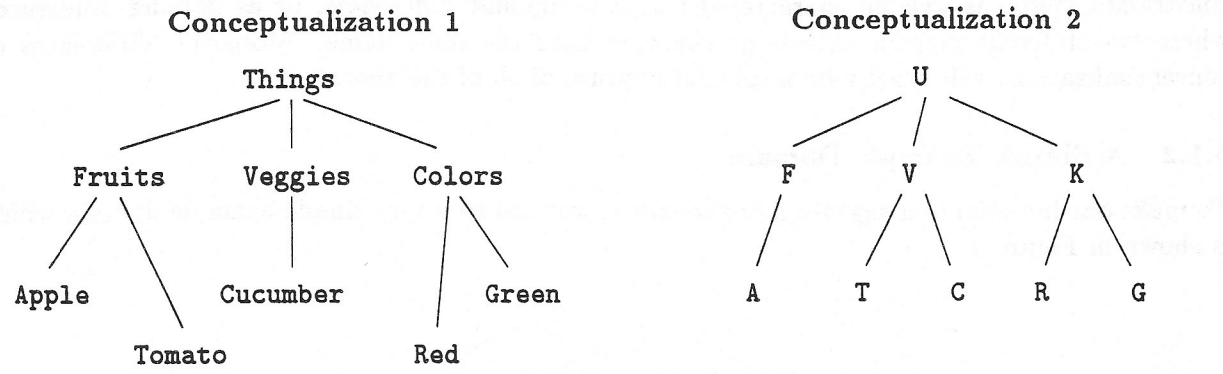

Figure 4: Two taxonomies about Veggie World

5.1.5 Identification of Individuals

Individuals in the domain can be identified with their name, e.g., *Bill Clinton*, with a definite description, e.g., *the president of the U.S.*, or with some sort of deixtic reference, e.g., *the guy over there next to Hillary*. Ignoring deixtic reference, at least the following problems can occur:

1. The sets of individuals known to each agent only partly overlap, hence, an agent can refer to an individual that is not known to the other agent, and hence unidentifiable.
2. The agents might have different names for the same individual.
3. The agents might have the same name for different individuals.
4. The agents might use different definite descriptions for the same individual, for example, *the last Friday of this month* vs. *next Friday*.

5. The agents might use the same definite descriptions for different individuals, for example, *the first floor* meaning either the floor at the same level as the ground or one floor above the ground.

5.1.6 Identification of Properties and Relations

5.2 Protocol Problems

Interagent communication is based on a message passing protocol in which agents send messages to other agents, and receive messages from other agents. Each message has an explicit sender and an explicit receiver. The communication pathway can be any medium; we are thinking of a physical network connection employing standard data protocols such as TCP/IP. These protocols are separate from the protocols discussed here. We presume that both agents can employ the same language in the sense that they employ the same syntax, and use the same closed-class vocabulary. Open-class words may differ between communicating knowledge agents.

An ontological mediator sits like a two-way filter between two particular communicating knowledge agents. It serves as a translator for messages the agents send to each other. In our current model, every pair of knowledge agents will need its own translator. The OM monitors at least the syntactic content of sending agent's messages to ensure they will be received without misunderstanding. The translation problems mentioned in the previous section apply to the OM's ability to perform this task.

It is a requirement that agents who wish to communicate have ontologies that overlap. It is not necessary for the agents to share the same terms for every item of discourse, but at least they must share some intensional objects. In other words, if two agents' ontologies have nothing in common, they can have nothing to discuss with one another. Attempts by agent A to communicate with agent B when B doesn't share any of A's ontology need to be politely interrupted by the mediator.

Given that the communicating agents have some common ground in their ontologies, An ontological mediator can be used to extend the ontologies of either agent. If one agent begins to use a term unknown to the other agent, the mediator will introduce the term, and attempt to explain the term using language the other agent already understands. If the other agent is not satisfied by the explanation, or needs further information, it may, via the mediator, query the first agent. The ensuing meta-discussion ends when both agents are satisfied that the second agent understands the term well enough to use it appropriately in the context of communication. An appropriate task for the OM will be to determine when the meta-discussion has exhausted its usefulness for the agents, and it is time to continue the original communication.

A prerequisite to this process is that the mediator know something about the ontologies of both agents. In order to explain a term or translate terms between ontologies, the OM must be able to compare pieces of ontology from both agents until it finds sufficient similarity. It can extract pieces of an agent's ontology remotely by querying the agent, or it could store the complete ontologies of both agents in its own memory. The obvious advantage to maintaining local copies is reduced overhead, but it comes at the price of having to monitor dialog between agents and determine when ontologies are modified.

5.3 Design Problems

6 Terminological Representation and Translation

An ontological mediator needs to have knowledge of the base-level category words used by the agents whose communication it is mediating. We presume that the communicating agents speak the same language, and barring any evidence to the contrary, they use the same words to speak of objects known to both agents. Often, however, different agents will use different words to represent the same object, or the same word to represent different objects.

We have developed a representation scheme for ontologies and a mechanism for translating sentences when the agents use different words to represent the same object. Words used by a speaking agent are translated into the equivalent words used a listening agent. The agents themselves do not represent this meta-ontological translation knowledge. Rather, it is stored in the knowledge base of the Ontological Mediator. Ultimately, when agent A wants to sent a message to agent B, A will send the message to the OM for appropriate translation, then the OM will forward the message with translated terms to agent B.

Meta-ontological knowledge about the terms used by two agents is formed in the belief space of the ontological mediator by a set of asserted propositions of the form "Agent X believes that expression T denotes object O." Each agent is represented as an individual base node in the OM's semantic network. To specify a specific term translation, two propositions are asserted in OM's belief space: "Agent X believes that expression T_1 denotes object O_i," and "Agent Y believes that expression T_2 denotes object O_i." The O_i's in the OM represent the same concept for which the communicating agents use different terms. An O_i can be an individual or a class.

Often it is the case that a word from the ontology of the speaking agent has no equivalent term in the ontology of the listener. One example is when the speaker and listener disagree as to the classification of a concept in their unified domain of discourse. In such cases, a definition must be built for the listener in terms for which the listener and speaker have previously agreed on definitions. In Veggie World, two agents can disagree on whether to classify tomato as vegetables or as fruits. The ontological mediator can build translations that are more complicated than simple term replacements. By analyzing the ontologies of both agents, including semantics, the mediator builds and stores set-theoretic translations from speaker to listener in terms the listener understands. For example, in the case of Veggie World, The OM deduces from the structure of the ontology and the meanings of the terms that the class denoted by "v" in the second taxonomy is the same as the class denoted by "veggie" union the class denoted by "tomato" in the first taxonomy (see figure 4) and that "v" therefore translates (loosely) to "veggie" + "tomato" The OM had previously determined that "t" and "tomato" refer to the same class in order to perform this inference.

The following demonstration illustrates the OM's ability to use translations when each agent has an equivalent term for each of the other's terms, and both agents agree on taxonomic classifications. Demonstration of more complicated translation is in progress.

6.1 Demonstration

This demonstration uses the Semantic Network Processing System [Shapiro and Group, 1992] (SNePS), version 2.1 as the knowledge representation and reasoning system of all three agents (speaker, listener, and mediator). In this early prototype, only one computer process is running. The belief space of each agent is maintained separately, and a message passing scheme simulates

having the agents on multiple processes or multiple computers. The '*' is the SNePS prompt. The demo run has been edited for brevity.

The first few commands reset the semantic network and load appropriate translation support functions written in Common Lisp.

```
* (resetnet t)

Net reset

* (^ (load "/projects/aec/om2-term-translate.lisp"))
; Loading /projects/aec/om2-term-translate.lisp.

(T)
```

The next few commands create belief spaces, called "contexts" for the communicating agents A and B, and a context for the ontological mediator. Finally, SNePS relations are created to support the networks.

```
* (set-context () a)

((ASSERTIONS NIL) (RESTRICTION NIL) (NAMED (A DEFAULT-DEFAULTCT)))

* (set-context () b)

((ASSERTIONS NIL) (RESTRICTION NIL) (NAMED (B A DEFAULT-DEFAULTCT)))

* (set-context () om)

((ASSERTIONS NIL) (RESTRICTION NIL) (NAMED (OM B A DEFAULT-DEFAULTCT)))

* (define subclass superclass member class agent act object expression)

(SUBCLASS SUPERCLASS MEMBER CLASS AGENT ACT OBJECT EXPRESSION)
```

Now we create an ontology for agent A, complete with individual members of each class.

```
* (set-default-context a)

((ASSERTIONS NIL) (RESTRICTION NIL) (NAMED (OM B A DEFAULT-DEFAULTCT)))

* (assert subclass apple superclass fruits)
* (assert subclass tomato superclass fruits)
* (assert subclass cucumber superclass veggies)
* (assert subclass red superclass colors)
* (assert subclass green superclass colors)
* (assert subclass fruits superclass things)
* (assert subclass veggies superclass things)
```

```
* (assert subclass colors superclass things)
* (assert member ap1 class apple)
* (assert member ap2 class apple)
* (assert member ap3 class apple)
* (assert member to1 class tomato)
* (assert member to2 class tomato)
* (assert member to3 class tomato)
* (assert member cu1 class cucumber)
* (assert member cu2 class cucumber)
```

The next command lists the terms in A's ontology. The `find-terms` function searches A's belief space for all asserted propositions of the forms above, and returns a list of all the terms found. Then, we show A's ontology as the list of asserted propositions in A's context.

```
* (^ (find-terms 'a))

(AP1 AP2 AP3 CU1 CU2 TO1 TO2 TO3 APPLE CUCUMBER GREEN RED TOMATO COLORS FRUITS
 THINGS VEGGIES)

* (^ (show-ontology 'a))

((ASSERT SUBCLASS CUCUMBER SUPERCLASS VEGGIES)
 (ASSERT SUBCLASS VEGGIES SUPERCLASS THINGS)
 (ASSERT SUBCLASS FRUITS SUPERCLASS THINGS)
 (ASSERT SUBCLASS COLORS SUPERCLASS THINGS)
 (ASSERT SUBCLASS TOMATO SUPERCLASS FRUITS)
 (ASSERT SUBCLASS APPLE SUPERCLASS FRUITS)
 (ASSERT SUBCLASS RED SUPERCLASS COLORS)
 (ASSERT SUBCLASS GREEN SUPERCLASS COLORS) (ASSERT MEMBER TO3 CLASS TOMATO)
 (ASSERT MEMBER TO2 CLASS TOMATO) (ASSERT MEMBER TO1 CLASS TOMATO)
 (ASSERT MEMBER CU2 CLASS CUCUMBER) (ASSERT MEMBER CU1 CLASS CUCUMBER)
 (ASSERT MEMBER AP3 CLASS APPLE) (ASSERT MEMBER AP2 CLASS APPLE)
 (ASSERT MEMBER AP1 CLASS APPLE))
```

We now enter the meta-ontological knowledge about terms into the belief space of the ontological mediator (context "om").

```
* (set-default-context om)

((ASSERTIONS NIL) (RESTRICTION NIL) (NAMED (OM B DEFAULT-DEFAULTCT)))

* (assert agent a act believe object (build expression things object #o))
* (assert agent b act believe object (build expression u object *o))

* (assert agent a act believe object (build expression fruits object #o))
* (assert agent b act believe object (build expression f object *o))
```

```
* (assert agent a act believe object (build expression veggies object #o))
* (assert agent b act believe object (build expression v object *o))

* (assert agent a act believe object (build expression colors object #o))
* (assert agent b act believe object (build expression k object *o))

* (assert agent a act believe object (build expression apple object #o))
* (assert agent b act believe object (build expression a object *o))

* (assert agent a act believe object (build expression tomato object #o))
* (assert agent b act believe object (build expression t object *o))

* (assert agent a act believe object (build expression cucumber object #o))
* (assert agent b act believe object (build expression c object *o))

* (assert agent a act believe object (build expression red object #o))
* (assert agent b act believe object (build expression r object *o))

* (assert agent a act believe object (build expression green object #o))
* (assert agent b act believe object (build expression g object *o))

* (assert agent a act believe object (build expression ap1 object #o))
* (assert agent b act believe object (build expression i0 object *o))

* (assert agent a act believe object (build expression ap2 object #o))
* (assert agent b act believe object (build expression i1 object *o))

* (assert agent a act believe object (build expression ap3 object #o))
* (assert agent b act believe object (build expression i2 object *o))

* (assert agent a act believe object (build expression to1 object #o))
* (assert agent b act believe object (build expression i3 object *o))

* (assert agent a act believe object (build expression to2 object #o))
* (assert agent b act believe object (build expression i4 object *o))

* (assert agent a act believe object (build expression cu1 object #o))
* (assert agent b act believe object (build expression i5 object *o))

* (assert agent a act believe object (build expression cu2 object #o))
* (assert agent b act believe object (build expression i6 object *o))
```

Now that the translation knowledge has been entered, we can translate individual terms and sentences. Translations of individual terms are printed as a list, but this is merely an artifact of SNePS. It does not affect the operation of translation.

```
* (^ (base-translate 'a 'b 'red))

(R)

* (^ (base-translate 'b 'a 'k))

(COLORS)

* (^ (base-translate 'b 'a '(assert subclass c superclass v)))

(ASSERT SUBCLASS CUCUMBER SUPERCLASS VEGGIES)

* (^ (base-translate 'a 'b '(assert member to1 class ?x)))

(ASSERT MEMBER I3 CLASS (? X))
```

B has no ontology as yet.

```
* (^ (show-ontology 'b))

NIL
```

We can now determine what B's ontology would look like if it agreed with A, but used the terms given in the translation. This merely shows the ontology. It does not actually create an ontology for agent B.

```
* (^ (translate-ontology 'a 'b))

((ASSERT SUBCLASS C SUPERCLASS V) (ASSERT SUBCLASS V SUPERCLASS U)
 (ASSERT SUBCLASS F SUPERCLASS U) (ASSERT SUBCLASS K SUPERCLASS U)
 (ASSERT SUBCLASS T SUPERCLASS F) (ASSERT SUBCLASS A SUPERCLASS F)
 (ASSERT SUBCLASS R SUPERCLASS K) (ASSERT SUBCLASS G SUPERCLASS K)
 (ASSERT MEMBER TO3 CLASS T) (ASSERT MEMBER I4 CLASS T)
 (ASSERT MEMBER I3 CLASS T) (ASSERT MEMBER I6 CLASS C)
 (ASSERT MEMBER I5 CLASS C) (ASSERT MEMBER I2 CLASS A)
 (ASSERT MEMBER I1 CLASS A) (ASSERT MEMBER I0 CLASS A))
```

We finally give B its ontology by explicitly sending it to B. Message sending at the current stage of development simply calls the ontological mediator to translate a message from the sender's dialect to the listener's dialect. The mediator then "forwards" the message by asserting the translation in the listener's belief space. In the future, a true multi-platform networked model will be created, where agents will actively receive messages, and messages to be sent will be filtered through a truly separate OM. This demonstration ends by showing B's ontology, as represented in the semantic network.

```
* (^ (multiple-message-send 'a 'b (find-ontology-structure 'a)))

OM received message from A: (ASSERT SUBCLASS CUCUMBER SUPERCLASS VEGGIES)
                 sent to B: (ASSERT SUBCLASS C SUPERCLASS V)
OM received message from A: (ASSERT SUBCLASS VEGGIES SUPERCLASS THINGS)
                 sent to B: (ASSERT SUBCLASS V SUPERCLASS U)
OM received message from A: (ASSERT SUBCLASS FRUITS SUPERCLASS THINGS)
                 sent to B: (ASSERT SUBCLASS F SUPERCLASS U)
OM received message from A: (ASSERT SUBCLASS COLORS SUPERCLASS THINGS)
                 sent to B: (ASSERT SUBCLASS K SUPERCLASS U)
OM received message from A: (ASSERT SUBCLASS TOMATO SUPERCLASS FRUITS)
                 sent to B: (ASSERT SUBCLASS T SUPERCLASS F)
OM received message from A: (ASSERT SUBCLASS APPLE SUPERCLASS FRUITS)
                 sent to B: (ASSERT SUBCLASS A SUPERCLASS F)
OM received message from A: (ASSERT SUBCLASS RED SUPERCLASS COLORS)
                 sent to B: (ASSERT SUBCLASS R SUPERCLASS K)
OM received message from A: (ASSERT SUBCLASS GREEN SUPERCLASS COLORS)
                 sent to B: (ASSERT SUBCLASS G SUPERCLASS K)
OM received message from A: (ASSERT MEMBER TO3 CLASS TOMATO)
                 sent to B: (ASSERT MEMBER TO3 CLASS T)
OM received message from A: (ASSERT MEMBER TO2 CLASS TOMATO)
                 sent to B: (ASSERT MEMBER I4 CLASS T)
OM received message from A: (ASSERT MEMBER TO1 CLASS TOMATO)
                 sent to B: (ASSERT MEMBER I3 CLASS T)
OM received message from A: (ASSERT MEMBER CU2 CLASS CUCUMBER)
                 sent to B: (ASSERT MEMBER I6 CLASS C)
OM received message from A: (ASSERT MEMBER CU1 CLASS CUCUMBER)
                 sent to B: (ASSERT MEMBER I5 CLASS C)
OM received message from A: (ASSERT MEMBER AP3 CLASS APPLE)
                 sent to B: (ASSERT MEMBER I2 CLASS A)
OM received message from A: (ASSERT MEMBER AP2 CLASS APPLE)
                 sent to B: (ASSERT MEMBER I1 CLASS A)
OM received message from A: (ASSERT MEMBER AP1 CLASS APPLE)
                 sent to B: (ASSERT MEMBER IO CLASS A)

* (^ (show-ontology 'b))

((ASSERT SUBCLASS C SUPERCLASS V) (ASSERT SUBCLASS V SUPERCLASS U)
 (ASSERT SUBCLASS K SUPERCLASS U) (ASSERT SUBCLASS F SUPERCLASS U)
 (ASSERT SUBCLASS R SUPERCLASS K) (ASSERT SUBCLASS G SUPERCLASS K)
 (ASSERT SUBCLASS T SUPERCLASS F) (ASSERT SUBCLASS A SUPERCLASS F)
 (ASSERT MEMBER TO3 CLASS T) (ASSERT MEMBER I4 CLASS T)
 (ASSERT MEMBER I3 CLASS T) (ASSERT MEMBER I6 CLASS C)
 (ASSERT MEMBER I5 CLASS C) (ASSERT MEMBER I2 CLASS A)
 (ASSERT MEMBER I1 CLASS A) (ASSERT MEMBER IO CLASS A))
```

7 Notes, To Do, etc.

- The section on Protocol problems is a rough sketch. More thought and detail needs to be given.
- We can only hope to translate between expressions that cover the same domain.
- Are we only mediating between different taxonomies (ontologies in the narrower sense), or are we trying to give complete semantically correct translations? For example, one system uses time points to model time, the other uses intervals. They might use the same taxonomies, but the temporal assertions about individuals might be completely incompatible. This answer to this question is still unclear, and probably needs to be addressed soon.
- How could we give a formal characterization that a translation is correct, i.e., that all aspects of something got translated appropriately? One method could be to have the agents talk. If the agent notices incorrect/incoherent usage, it could flag the mediator or the other agent. This would trigger meta-communication to resolve the misunderstanding. See protocol problems above.
- Between what languages do we translate? Perhaps SNePSUL and KIF. This would require SNePS to adopt a standard ontology for its knowledge, something we're not sure should/could be done. Do we distinguish between the KRL of an agent and its public communication languge? We think not; By public, we mean a language that has been adopted for interagent communication — theoretically any language could be used, although the interlingua of choice seems KIF. A better answer to this question might be, "Only if the agent itself does." Do we only translate between public communication languages?

References

[Ehrlich, 1994] Karen Ehrlich. Automatic expansion of vocabulary through natural language contexts. Draft of Ph.D. dissertation, 1994.

[Finin and others, 1992] Tim Finin et al. An overview of KQML: A knowledge query and manipulation language. Unpublished Draft, 1992.

[Finin, 1991] Tim Finin. Re: KB interchange standards. E-Mail message to Gio Wiederhold et al. from Nov.26,91, November 1991.

[Fritzson and Finin, 1991] Rich Fritzson and Tim Finin. Simple knowledge transfer protocol. Unpublished Draft, October 1991.

[Genesereth and Fikes, 1991] Michael R. Genesereth and Richard Fikes. *Knowledge Interchange Format.* Computer Science Department, Stanford University, Stanford, CA 94305, version 2.2 edition, March 1991.

[Goh *et al.*, 1994] Cheng Hian Goh, Stuart E. Madnick, and Michael Siegel. Context interchange: Overcoming the challenges of large-scale interoperable database systems in a dynamic environment. Working Paper WP# 3658-94, CISL WP# 94-01, Sloan School of Management, MIT, Cambridge, MA 02139, February 1994.

[Gruber, 1990] Thomas R. Gruber. The role of standard knowledge representation for sharing knowledge-based technology. Technical Report KSL 90-53, Knowledge Systems Laboratory, Stanford University, Stanford, CA, 1990.

[Gruber, 1991a] Thomas R. Gruber. Ontolingua: A mechanism to support portable ontologies. Technical Report KSL 91-66, Knowledge Systems Laboratory, Stanford University, Stanford, CA, 1991.

[Gruber, 1991b] Thomas R. Gruber. The role of common ontology in achieving sharable, reusable knowledge bases. In J. A. Allen, R. Fikes, and E. Sandewall, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Second International Conference*, pages 601–602, Cambridge, MA, 1991. Morgan Kaufmann.

[Gruber, 1993] Thomas R. Gruber. A translation approach to portable ontologies. *Knowledge Acquisition*, 5(2):199–220, 1993.

[Gruber, 1994] Thomas R. Gruber. Toward principles for the design of ontologies used for knowledge sharing. In Nicola Guarino and Roberto Poli, editors, *Formal Ontology in Conceptual Analysis and Knowledge Representation*. Kluwer Academic, 1994. in preparation, also available as Stanford Knowledge Systems Laboratory Report KSL 93-04.

[Guarino *et al.*, 1994] Nicola Guarino, Massimiliano Carrara, and Pierdaniele Giaretta. An ontology of meta-level categories. LADSEB-CNR Int. Rep. 6/93, ???, 1994. in preparation.

[Guarino, 1994] Nicola Guarino. The ontological level. In R. Casati, B. Smith, and G. White, editors, *Philosophy and the Cognitive Science*. Hölder-Pichler-Tempsky, Vienna, 1994.

[Knight and Luk, 1994] Kevin Knight and Steve K. Luk. Building a large scale knowledge base for machine translation. In *Proceedings of AAAI*, Seattle, WA, August 1994.

[Lehmann and Cohn, 1993] Fritz Lehmann and Anthony G. Cohn. The EGG/YOLK reliability hierarchy: Semantic data integration using sorts with prototypes. Technical Report 93-2, GRandAI Software, 1993.

[Lenat and Guha, 1989] Douglas B. Lenat and R.V. Guha. *Building Large Knowledge-Based Systems*. Addison-Wesley, Reading, MA, 1989.

[MacGregor, 1988] R. MacGregor. A deduction pattern matcher. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, 1988.

[Matos and Martins, 1989] Pedro A. Matos and João P. Martins. SNePSLOG - a logic interface to SNePS. Technical Report 89/03, Instituto Superior Técnico, Technical University of Lisbon, Lisbon, Portugal, 1989.

[McHale and Crowter, 1994] Michael L. McHale and John J. Crowter. Constructing a lexicon from a machine readable dictionary. Technical Report RL-TR-94-178, Rome Laboratory, Air Force Materiel Command, Griffiss Air Force Base, New York, November 1994.

[Neches *et al.*, 1991] Robert Neches, Richard Fikes, Thomas Gruber, Ramesh Patil, Ted Senator, and William R. Swartout. Enabling technology for knowledge sharing. *AI Magazine*, 12(3), Fall 1991.

[Penman, 1989] Penman. The penman documentation. Technical report, USC/Information Sciences Institute, 1989.

[Shapiro and Chalupsky, 1992a] Stuart C. Shapiro and Hans Chalupsky. KQML - issues and review. Final report, Department of Computer Science, State University of New York at Buffalo, Buffalo, NY, July 1992.

[Shapiro and Chalupsky, 1992b] Stuart C. Shapiro and Hans Chalupsky. KQML prototype interface. Final report, Department of Computer Science, State University of New York at Buffalo, Buffalo, NY, May 1992.

[Shapiro and Group, 1992] S. C. Shapiro and The SNePS Implementation Group. *SNePS-2.1 User's Manual.* Department of Computer Science, University at Buffalo, 1992.

[Steele, 1990] Guy L. Steele. *COMMON LISP.* Digital Press, second edition, 1990.

[Wiederhold *et al.*, 1990] Gio Wiederhold, Peter Rathmann, Thierry Barsalou, Byung Suk Lee, and Dallas Quass. Partitioning and composing knowledge. *Information Systems*, 15(1):61–72, 1990.

[Wiederhold *et al.*, 1991] Gio Wiederhold, Tim Finin, and Rich Fritzson. Kqml: Partial report on a proposed knowledge acquisition language for intelligent applications. Unpublished Draft, September 1991.

[Wiederhold, 1992] Gio Wiederhold. Mediators in the architecture of future information systems. *IEEE Computer*, pages 38–49, March 1992.