submitted for journal publication, January 1999

Agent Communication with Differentiated Ontologies:

eight new measures of description compatibility

Peter C. Weinstein and William P. Birmingham
Artificial Intelligence Laboratories
Department of Electrical Engineering and Computer Science
University of Michigan
1101 Beal Ave., Ann Arbor, MI 48109-2110
{peterw, wpb}@eecs.umich.edu

Abstract

We propose an approach to achieve appropriate exchange of services and data in distributed systems subject to semantic heterogeneity. We assume *differentiated ontologies:* that terms have formal definitions as concepts related to other concepts, that local concepts inherit from concepts that are shared, and that most or all primitives are shared. We then develop measures of *description compatibility* using the structure of the source and target definitions. We evaluate these measures by generating description-logic ontologies in artificial worlds. In our simulations, the "meaning" of a concept is its denotation in a finite universe of instances. The accuracy of the description-compatibility measures can thus be judged by their success in predicting the overlap of concept denotations. Description compatibility can be used to guide agent search for services across communities that subscribe to differentiated ontologies.

1 Introduction

We are interested in the semantics of agent communication in large, distributed systems developed in a decentralized manner. In these systems, semantic heterogeneity is inevitable. “Semantics” is the association of terms with the world; “semantic heterogeneity” exists when term meanings are not coherent throughout the system. We can be more precise within the simplifying framework of model theory: *semantically heterogeneous terminology* is a set of symbols with inconsistent denotations (the same symbol references different sets of objects), redundant denotations (different symbols reference identical sets of objects), or both.

Semantic interoperability—the appropriate exchange of data and services in semantically heterogeneous systems—requires some form of translation between terminologies. Most research in this area focuses on how to represent *mappings,* which identify correspondences between items in two terminologies. Mappings may be represented as conditional rules (e.g., Chang and Garcia-Molina (Chang and Garcia-Molina 1998)), functions (e.g., Chawathe et. al. (Chawathe, Garcia-Molina et al. 1994)), logic (e.g., Guha (Guha 1992)), or as sets of tables and procedures (e.g., Weinstein and Birmingham (Weinstein and Birmingham 1998)). Most often, mapping is between local terminologies and a global terminology, since this configuration requires a number of mappings linear in the number of local terminologies rather than a quadratic number of mappings. Unfortunately, it is often quite difficult to specify global terminologies that are adequate for all intended uses. Furthermore, although there have been preliminary attempts to automate the specification of mappings (Bright, Hurson et al. 1994; Lehmann and Cohn 1994; Campbell and Shapiro 1995), this process is usually mostly manual. Typically, manual specification of mappings is extremely expensive for real-world problems (Heiler 1995).

This paper describes measures of the syntactic correspondence between the definitions of pairs of terms. These “description-compatibility measures” can be used to search for satisfactory mappings, or to bypass mappings altogether. The algorithms that implement the measures vary in the complexity of their computation and the knowledge that they require about the domain. Some of the compatibility measures utilize *matching:* the identification of maximal one-to-one correspondences between elements of the compared definitions.

We use simulations to study the accuracy of our compatibility measures. In the context of real-world applications, it is not possible to calculate the meaning of a term. In our simulations, however, we can calculate meaning. First, we generate artificial sets of terminology definitions that describe a finite universe of artificial objects. We then define a term's meaning in a literal model-theoretic way, as the set of objects that satisfies the term's definition. Given any two terms, we can calculate the “semantic overlap” of their meanings using the intersection of their denoted sets. These semantic measures then provide an authoritative basis for evaluating the accuracy of the syntax-based compatibility measures.

The remainder of this paper is organized as follows. We define our measures of description compatibility in Section 6, the focus of the paper. The preceding sections are requisite for a precise understanding of the measures and the experiments. Section 2 defines “differentiated ontologies”, structures that contain term definitions that support calculation of description compatibility. Section 3 describes the model of agent interaction that underlies our experiments. Sections 4 and 5 describe the artificial ontologies that provide data for our experiments, including the representation language, and the process of generating the ontologies. Section 6 presents the description-compatibility measures, including algorithms to construct matchings between elements of concept definitions. Section 7 analyzes the performance of the syntactic measures by comparison to the measure of semantic overlap defined in Section 3. Section 8 is a discussion, and Section 9 concludes.

2 Differentiated Ontologies

To enable calculation of the compatibility measures, we make certain assumptions about the syntactic representation of terms to be mapped. Structures that satisfy these assumptions are *differentiated ontologies.*

Assumption 1: Terms are defined as concepts in formal ontologies.

Ontologies are sets of concept definitions. Formal ontologies use logic to define concepts by their relations to other concepts. For example, chair might be defined as a-kind-of furniture that a person can sit-upon.[1] The concept chair denotes the set of objects that are chairs; the instance chair001 denotes a particular chair object. The logic must, like description logic, support computational inference of subsumption (Woods 1991). Concept A subsumes B if and only if every instance of B is always an instance of A. Highchair, for example, might be defined as a-kind-of furniture that a toddler can sit-upon; by comparing the definitions for chair and highchair and person and toddler, respectively, description logic can deduce that highchair is a-kind-of chair. (See Weinstein and Birmingham (Weinstein and Birmingham 1998) for an expanded description of "formal ontology"; in this paper, Section 4 specifies the representation that we use in our simulations.)

The structure of ontological definition provides a basis for syntactic comparison. The precision required by logic, and in particular the support for subsumption inference, is required to enable calculation of the compatibility measures.

Whereas Assumption 1 is fundamental to our enterprise, the following assumptions are "soft" in that they can be violated, but increasing frequency of violation yields increasingly poor results for our description-compatibility measures.

Assumption 2: Concepts inherit definitional structure from concepts that are shared.

This assumption identifies the form of commonality that we use to enable automatic mapping. Note that mapping between concepts in differentiated ontologies is equivalent to mapping between concepts in a single ontology that provides separate spaces for concept names.

Assumption 3: Primitive concepts and relations are shared.

Non-primitive definitions have both necessary and sufficient conditions, but primitives have only necessary conditions. For example, given the primitive ($C \Rightarrow D$), we cannot infer that instance $j \in C$ unless this is asserted explicitly. Clearly, there can be no basis for matching a primitive element of a concept definition to any element of a concept in an ontology where the primitive is undefined.

We expect differentiated ontologies to occur in two important contexts. First, architects may design infrastructures for multi-agent systems that include one or more generic ontologies to which all agents are expected to subscribe. As systems change over time, developers will create local versions of ontologies that are specialized to help meet the needs of particular user communities. Over time, large ontological structures will develop, and communities of agents will be defined by the ontologies they use to communicate. Agents will interact with agents in their own community, but may use the techniques

[1] We use font to identify concept and relation names in a real or hypothetical ontology.

described in this paper to seek services from agents in other communities. We expect that mapping will be more effective between ontologies whose concepts share a relatively high percentage of definitional structure, compared to mapping between more remote ontologies. The experiments described in this paper are designed within this context of agent communication.

We also believe that differentiated ontologies will have an important role in the integration of heterogeneous data. In this context, differentiated ontologies will be manually constructed as a less-demanding alternative to a global terminology. Each local schema will be mapped to one ontology in a family of ontologies. For example, we generated a knowledge base of library metadata by mapping from a standard library format, MARC (Machine Readable Cataloging), to an ontology that models bibliographic relationships between works (Weinstein and Birmingham 1998). We were able to specify the mappings in a straightforward way by the simple expedient of permitting the ontology to resemble MARC somewhat more than it should. In the future, we plan to map from a variety of metadata standards to a family of differentiated ontologies. Queries can be represented as concepts, and measures similar to those described in this paper will then support an imprecise query service that integrates data from the different sources.

3 Simulation Scenario

An agent in need of a service might use a variety of strategies to search for agents capable of providing the service. Frequently, broker agents play important roles in multi-agent systems; they know about available services and, in various ways, act to help requesting and providing agents to coordinate (Kuokka and Harada 1995).

To structure our analysis of agent communication, we focus on a straightforward pattern of agent interaction where agents ask brokers for service recommendations, as pictured in Figure 1. Agent A formulates the request as an ontological concept definition, which, in this context, we call a *service description.* In the figure, we represent service descriptions as triangles. Broker agents know about services available in a community of agents, which are also represented by service descriptions. These descriptions may have been defined by agents as a means of advertising (Weinstein and Birmingham 1998). The broker applies some sort of filter to select a set of candidate services, then compares each candidate service description to the request. In the figure, the thickness of the gray lines indicates the strength of the mapping. The broker then recommends the service whose description maps most strongly to the request.

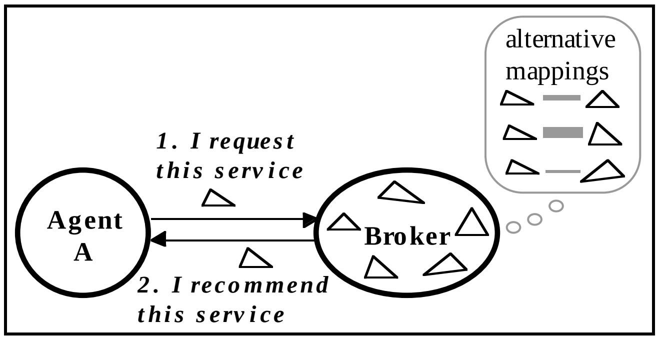


Figure 1: Model of agent interaction

In this model, the broker recommends a service description, which is an ontology concept, rather than a particular service-providing agent. It is possible that many agents can provide the service (presumably, the broker or some other agent knows those agents providing each service). In our simulations, instances of agent service provision are represented as ontology instances. Thus, the meaning of an agent's service description is the set of particular behaviors that the agent might execute to provide the service.

Accordingly, we measure the quality of the recommendation as the probability of *success*, which we define as occurring when an instance selected from the recommended set also satisfies the request:

$$\begin{aligned} P(\mathit{Success}) &= P(j \in C^I \mid j \in D^I) && (1) \\ &= P(j \in C^I \cap D^I) / P(j \in D^I) \end{aligned}$$

where C is the request, D the recommendation, C^I and D^I the extensions of C and D, respectively, under logical interpretation *I*, and j is an instance selected uniformly randomly from D^I. Figure 2 illustrates Definition (1) with a Venn diagram.

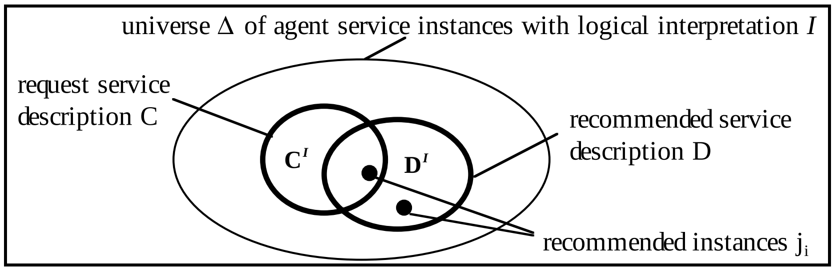

universe Δ of agent service instances with logical interpretation I
request service description C
C^I
D^I
recommended service description D
recommended instances j_i

Figure 2: Measuring semantic overlap using concept denotations

In our simulations, the universe of objects is large, but finite. Thus, we operationalize our semantic measure for recommendations (SEMR) as:

$$\mathrm{SEMR}(C, D) = |C^I \cap D^I| \;/\; |D^I| \,. \qquad (2)$$

For some purposes, a symmetric measure of semantic overlap is appropriate. This measure is independent of the pragmatic roles of concepts C and D, and is calculated by:

$$\mathrm{SEMS}(C, D) = |C^I \cap D^I| \;/\; |C^I \cup D^I| \,. \qquad (3)$$

4 Ontology Representation

We designed the representation of our artificial ontologies in accordance with our experience developing ontologies in complex domains (Weinstein and Birmingham 1998), and our expectations for the sensititivities of our measures of description compatibility. We knew that to match concept definitions we would convert concept definitions to description graphs (Borgida and Patel-Schneider 1994; Cohen and Hirsh 1994), so we thought in terms of graph structure. Thus, we considered it essential to include multiple inheritance among concepts, and believed that it would be important to include relations between role fillers, which would produce cycles within concept description graphs. We excluded less useful features that would add complexity to our simulations without reason to expect an important impact on measures of description compatibility. We discuss the issue of expressivity further in Section 8.2.

Table 2 presents the syntax and semantics for a language that we call Very Basic Description Logic (VBDL). VBDL is similar to CoreClassic (Cohen and Hirsh 1994), except that roles are quantified existentially rather than universally, and relations may inherit from other relations.

Table 2: Syntax and semantics for Very Basic Description Logic (VBDL)

Language Element	Definition Syntax	Notation	Denotation
instance	(new-instance c :primitives :attributes/values)	c	$c \in \Delta$
value		v	$v \in V=\{1,2,\ldots,k\} \subseteq \Delta$
concept	(new-concept C :parents :roles :constraints)	C	$C^I \subseteq \Delta$
primitive	:is-primitive t	P	$P^I \subseteq \Delta$
relation	(new-relation R :parents :domain :range :inverse)	R	$R^I \subseteq \Delta \times \Delta$
conjunction	:parents (C D)	C ¿ D	$C^I \cap D^I$
roles	:roles ([R D]*)	C.R:D	$\{c \in \Delta \mid \exists d \in \Delta.\ c \in C^I,\ d \in D^I,\ (c,d) \in R^I\}$
type constraints	:constraints ([R_1 R_2 R_3]*)	$C.R_1$:($C.R_2$ $C.R_3$)	$\{c \in \Delta \mid \exists d,e \in \Delta.\ c \in C^I,\ (c,d) \in R_2^I,\ (c,e) \in R_3^I,\ (d,e) \in R_1^I\}$
value constraints	:constraints ([R_1 R_2 v]*[R_1 v R_2]*)	$C.R_1$:($C.R_2$ v)	$\{c \in \Delta \mid \exists d \in \Delta,\ v \in V.\ c \in C^I,\ (c,d) \in R_2^I,\ (d,v) \in R_1^I\}$

Concept definitions include “roles” and “constraints”. Roles include a relation, and a concept that restricts the set of instances that can satisfy the relation, called a “type restriction”. Instances that satisfy a concept role are “fillers”. Constraints include a relation that must be satisfied by one or two of the concept's fillers. We call constraints between two fillers “type constraints”. “Value constraints” require that a filler be a number that is in the desired numeric relation to a given constant. Relations do not have roles, but can have domain or range type restrictions, and inverses. The “:parent” argument for new concepts identifies concepts from which the new concept inherits roles and constraints (and the same applies analogously to relations).

To meet the requirements of our simulations, we implemented a customized, very fast VDBL system. This implementation makes the following simplifications:

1) Roles are always filled with exactly one filler (they are “attributes”).
2) Numeric fillers belong to a finite and known set of values.
3) Concept definitions are not recursive (a concept cannot be a type restriction in its own definition).
4) All relations have inverses.
5) Assertions about instances are monotonic.
6) All instances are accessible to all contexts.

“Contexts” restrict accessibility within an ontological structure. In this research, we are not concerned with issues related to parallel execution on distributed platforms. Therefore, we found it convenient to implement differentiated ontologies in a single structure. We use contexts to hide concepts and relations that would not be accessible in a distributed system.

Every concept and relation is associated with some context, and is always accessed from the perspective of some context. Contexts are partially ordered, i.e., accessibility among contexts is transitive and non-symmetric. For example, if ontology B contains concepts specialized from ontology A, concepts in ontology A will be visible from the context of ontology B, but not vice versa. Figure 3 illustrates a fairly common occurrence, where a concept defined in ontology B is classified between two ontology A concepts, but remains invisible from the perspective of context A.

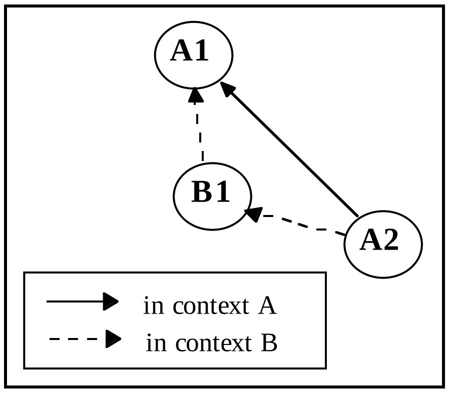


Figure 3: Direct subsumption links in multiple contexts

5 Ontology Generation

This section describes the generation of the artificial ontologies that provide data to evaluate our compatibility measures. We start with a set of world "objects" that the ontologies will describe. These objects are hypothetical instances of service provision. We call the representations of objects in the description-logic ontologies "instances". Our description-logic concepts are service descriptions. We also start with a tree of "communities" that shape ontology differentiation. We think of communities as organized groups of users with increasingly specialized interests; they bias the generation of new concepts to utilize increasingly small subsets of the attributes that describe the objects. We generate one ontology for each community.

The insight that drives concept generation within ontologies is this: we create terms that distinguish certain objects from other objects. Figure 4 illustrates. The axes represent the relations and attributes that define a space of potential services. We pick an object, find the least general concepts that describe the object, pick one, and add to its definition to create a new, relatively specialized concept that includes the target object.

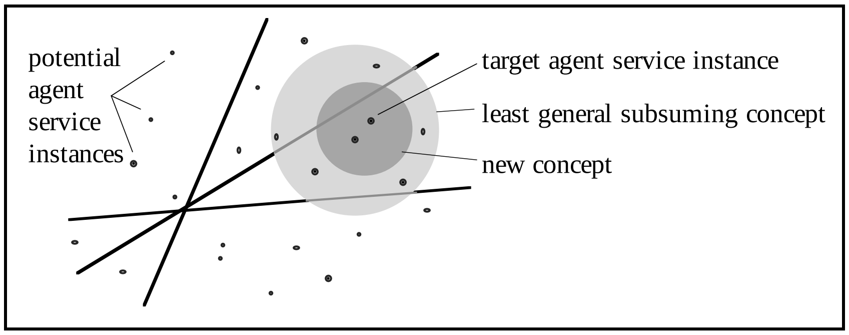

potential agent service instances
target agent service instance
least general subsuming concept
new concept

Figure 4: A new concept distinguishes a new service from an old

The following subsections describe objects, communities, and concept generation in detail. We illustrate with examples and data from Experiment 15, the basis of most of the results reported in Section 7 (we used Experiments 1 through 14 to develop the process of ontology generation).

5.1 Objects

Our simulated worlds must have some structure that the ontologies can model, and the description compatibility measures can exploit to produce interesting behavior. This structure should be simple, to avoid obscuring the nature of the results, yet plausible as a highly abstracted model of the real world.

Objects in our simulations are characterized by relations to numeric values, which we call "attributes", and by relations to other objects. These attributes can be thought of as describing any kind of real world characteristic: for example, the color of an object, its structure, behavior, or even less inherent attributes such as function, or context. Our object attributes have the following characteristics:

(a) Some values are relatively rare.

(b) Attribute values are correlated with certain other attributes.

(c) Objects have values for different sets of attributes.

(d) Values can be interpreted as referring to other objects, which we explain below.

Experiment 15 has objects with structures as illustrated by Figure 5. Each object has five "clusters" (L1 through L5), each with three slots for attributes. Three of the clusters have attributes with values. Objects are associated with values by relations of the form has-LxAy, where x is an integer that identifies the value's cluster, and y is an integer that identifies the attribute within the cluster. For example, Figure 5 includes the role <HAS-L1A3 OBJ00123 4>.

	L1	L2	L3	L4	L5
	A1 A2 A3 ...				
OBJ00123	4 5 4	- - -	1 1 3	3 4 5	- - -

Figure 5: Objects have clusters of attribute values

We generate the values randomly with a multivariate normal distribution. The values are correlated within clusters. These clustered correlations represent the structure of the world: for example, we expect tall men to also be relatively heavy. Or, say that we are concerned with agent services in digital libraries (Weinstein and Birmingham 1998), and OBJ00123 is one of a group of services that provide access to educational videos. We would expect that the intended audience of the videos and their conceptual-level would strongly covary, thus hypothetically supporting mappings between services for elementary-school, and those for reading-grade-level-2.

Attribute values are also significant in a second way, as cluster indices that constrain links to other objects that enrich the object's description with further detail. In Figure 6, for example, attribute A2 in cluster L1 of object OBJ00123 has the value 5; this has the consequence that OBJ00123 will have a "described-by" link from that value only to an object with values in the 5th cluster in some other object (in this case, OBJ32789). To continue our previous example, say that attribute A2 of cluster L1 identifies object OBJ00123 as a live video access service. Because OBJ00123 delivers video, we expect it to have values for certain other attributes that describe different aspects of the service; for example, compression-format, delivery-rate, and so on. These values are supplied by a link to another object, OBJ32789.

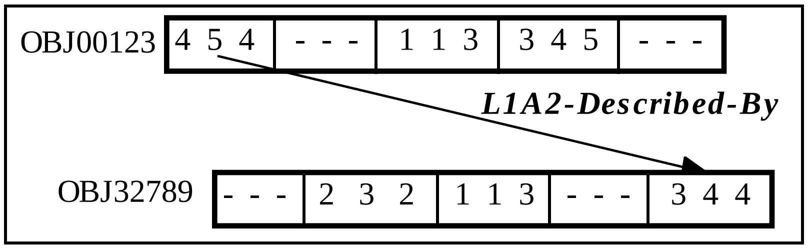


Figure 6: A value references a cluster in another object

In our simulations, every value of every object has a relation in the form LxAy-Described-By to some other object, as illustrated in the figure. A parameterized "cliquing bias" determines the probability that an object will be linked to objects that are also linked to each other, when it is possible to do so. This bias creates some structure in the pattern of relations between objects; otherwise, the greater the number of objects, the less likely such cliques would be.

Table 3 summarizes the parameters that control the generation of objects in our simulations. To guarantee that the attribute correlations are valid for multivariate normal distributions, we generate them from a parameterization of the normal distribution as a composition of ellipses. The mean attribute value is not listed in the table because it is determined by the number of clusters. Finally, the ontologies are generated initially with a set of 10,000 objects. To reduce noise from small numbers, we subsequently repopulate the ontologies with different sets of objects 49 times.

Table 3: Parameters controlling object generation

Parameter	**Settings for Experiment 15**
Number of clusters	5
Number of attributes in each cluster	3
Standard deviation of values	1.25
Attribute value correlations	Within clusters: ρ(A1, A2) = 0.840 ρ(A1, A3) = 0.799 ρ(A2, A3) = 0.559 Across clusters: ρ(Ay, Az) = 0
Type relation cliquing bias	0.5
Number of objects	10,000 per universe, 50 universes

5.2 Communities

We use a tree of "communities" to model the specialization of perspective that may be expected of agent communities. Each community has a set of attributes that are preferred over other attributes when generating concepts, as discussed in the next section. Each community's preferred attributes are a subset of attributes preferred by its parent community. At the top of the community tree all attributes are equally preferred.

Figure 7 illustrates the tree of communities used in Experiment 15. Each community label starts with the attributes that are preferred within each cluster. If an "L" is present, the following digits indicate clusters that are preferred. Communities that have equal preferences are distinguished with a small "a" or "b". Note however, that concepts in the ontologies associated with these communities will nevertheless be differentiated, as they inherit structure from different sets of concepts (generated by communities with different preferences).

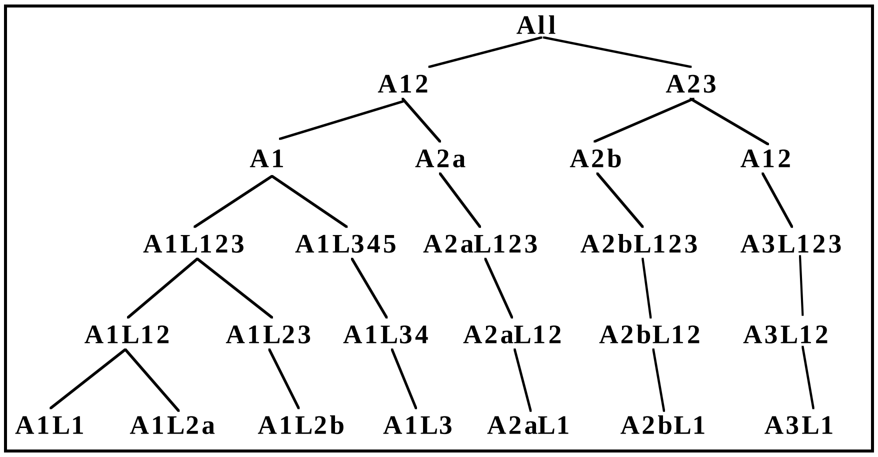


Figure 7: Tree of communities

The community tree is filled flush left to provide maximum variation in the number of shared and unshared ontologies when comparing concepts. For example, comparisons between concepts in the ontologies associated with communities A1L1 and A1L2a benefit from shared structure inherited from concepts in ontologies A1L12, A1L123, A1, A12, and the top ontology. Comparisons between A1L1 and A3L1, meanwhile, have only the top ontology in common.

5.3 Concept Generation

We define new concepts in such a way as to distinguish certain objects from other objects. Generating a new concept involves the following steps: select a target object, identify the most specific concepts ("parents") that describe the target, then define a new concept that specializes one or more parents using one of the methods described below. By design, the new concept must exclude a desired percentage of the instances of the specialized parents. Finally, identify instances that are members of the new concept.

Prior to concept generation, we create a "Top" ontology whose contents are inherited by all ontologies, including the "All" ontology at the top of the tree in Figure 7. The "Top" ontology includes the following: (1) a description-logic instance for each world object, asserted to be a member of the primitive concept Service, (2) a tree of value relations including has, has-Lx for each cluster, and has-LxAy for each cluster/attribute, (3) a tree of type relations including described-by, Lx-described-by, and LxAy described-by, (4) isomorphic trees of inverse relations, with of and describes at the top, (5) a Number concept, and (6) numeric relations (EQ, LT, etc.) for value comparisons. Instance names are of the form OBJxxxxx, where xxxxx is a unique integer. Most importantly, note that the trees of relations in the Top ontology model the structure of attributes within clusters, since attributes within a cluster have a more proximate common ancestor than do relations in different clusters.

Subsequent to concept generation, we delete many of the concepts from the interior of the ontologies. This yields a structure that has relatively realistic variation in the number of additional roles that distinguish concepts from their parents.

In more detail, we generate each concept as follows:

1) Select a target object to be "named" by the current community. If it has not already been selected for naming, it must have a value for at least one attribute favored by the current

community. Otherwise, it has already been named either by the current community, or one of its parents.

2) Determine the least-general concepts that describe the object. Consider only concepts that are accessible to the current community. Order these potential parent concepts in a way that is probabilistically biased such that concepts defined in the current or proximate ontologies are more likely to be picked for specialization than concepts defined in more remote ontologies.

3) If the object is not subsumed by any accessible concept, create a new child of the primitive Service concept that has one role.

4) Otherwise, pick one of the alternative methods described below for specializing concept definitions. Except for method (a), specialize the first of the potential parent concepts that has not already been tried. In all cases, the new concept must exclude a certain percentage of a sample of instances that are members of the parent concepts. This sample, called the "prior members", includes all instances that have previously been "named" by the current community or its parents, and a random sample of additional instances if necessary in order to obtain a sample of adequate size. The methods for defining new concepts include:

   a) Multiple inheritance: Conjoin two or more of the least-general-subsuming concepts. If combining the definitions of two parents does not distinguish the target from enough of the prior members, conjoin another parent, and so on.

   b) Add a value-relation role. The new role constrains its filler to be either greater or lesser than the mean value. We first try roles for attributes preferred by the current community. Of those, we first use attributes with relatively extreme values. If the new role does not successfully distinguish the target from enough prior members, other attributes are tried. If the method does not work for any attribute favored by the community, we try attributes favored by the parent community, and so on.

   c) Add a type-relation role. We define a new type relation corresponding to the most locally preferred attribute that does not already have a type relation in the parent concept's definition. The relation's name is of the form [Lx][Ay-]described-by[.<concept>], where the bracketed elements are optional. The goal is to be as unrestrictive as possible while distinguishing the target from enough prior members. Thus, we start with the most general type relation (described-by), and test increasingly specific concepts as range restrictions. If none of these relations succeed in distinguishing the target, we specialize the relation to refer only to fillers of a single cluster; and if this does not work, we add the attribute identifier as well. The new relation is classified in the same ontology as the new concept.

   d) Restrict a value filler. Replaces a numeric constraint in a value-relation role with a more restrictive constraint.

   e) Restrict a type filler. Replaces a type-relation role with a role with a more specific type restriction.

   f) Add a constraint between value fillers. One filler must be greater than, less than, or equal to another.

   g) Add a constraint between type fillers. One filler instance must be linked by a type relation to another filler. The new constraint adds a described-by relation between the fillers, if that suffices to distinguish the target from enough prior members. Otherwise, it requires a more specific LxAy-described-by relation between the fillers.

The new concept is named after the object that it distinguishes, in the form COBJxxxxx[s]. For convenience, in cases where the object has already been named, we add a letter that gives the concept a unique name.

5) If the method chosen in (4) does not succeed in distinguishing the target object from enough of the prior members, then, if the selected object has multiple least-general-subsuming concepts, the same method is tried with another potential parent concept. If the method fails for all the potential parents, then another method is tried, and so on until some method succeeds.

Figure 8 illustrates concept specialization using method (f), by showing the call to the new-concept function. The target instance is OBJ00175, the community is A1L123, the parent concept is COBJ00135A, and the new element added to the parent's definition is the LT constraint between the values that fill the HAS-L5A1 and HAS-L2A2 roles, respectively. Figure 9 show the concept definition that results from merging the new constraint with the parent concept's definition.

```
(NEW-CONCEPT 'COBJ00175 'A1L123
        :PARENTS '(COBJ00135A)
        :CONSTRAINTS '((LT HAS-L5A1 HAS-L2A2))
        :ANNOTATIONS '(CCM4-VAL-FILLERS-RESTRICTION))
```

Figure 8: Specializing a concept definition

```
A1L123:COBJ00175
        <ROLE: HAS-L2A2 NUMBER>
        <ROLE: DESCRIBED-BY.COBJ00001 COBJ00001>
        <ROLE: L5-DESCRIBED-BY.COBJ00022 COBJ00022>
        <ROLE: HAS-L5A1 NUMBER>
        <CONSTRAINT: LT (<RELATION: HAS-L2A2>) 3>
        <CONSTRAINT: LT (<RELATION: HAS-L5A1>) 3>
        <CONSTRAINT: LT (<RELATION: HAS-L5A1>) (<RELATION: HAS-L2A2>)>
```

Figure 9: A concept definition

Table 4 summarizes the parameters that control the generation of concepts. The actual frequencies of concept definition using the alternative methods do not match the parameters, because frequently most methods fail. The parameters can be adjusted, though, to obtain roughly the desired proportions.

Table 4: Parameters controlling concept generation

Parameter	Settings for Experiment 15
Probability of picking a target object that is not already "named"	0.5
Relative probability of picking previously named objects from objects named by the current community, rather than from objects named by the current community's parent (also used to bias prioritization of least-general-subsuming-concepts as potential parents).	2.0
The probability of selecting each subclassing method for defining new concepts.	.08 .50 .08 .10 .08 .08 .08
The percentage of prior members that must be distinguished from the target object.	60%
The percentage of concepts that have children that are subsequently deleted from the ontologies (the "thinning" percentage).	50%

Table 5 describes the ontology structure produced for Experiment 15. Generating 70 concepts for each community, then thinning half of the concepts with children yields 1327 concepts. The typical (median) concept has three ancestors before reaching the generic Service concept (counting its longest path to Service), one parent, two value-relation roles and one type-relation role, and 50 members.

Table 5: Concept characteristics summary (1327 concepts, 10000 instances)

Characteristic	Average	Median	Minimum	Maximum
Concept depth	2.9	3	1	7
Number of parents	1.7	1	1	9
Number of roles	3.5	3	1	8
value-relation roles (number fillers)	1.9	2	0	6
type-relation roles (instance fillers)	1.6	1	0	7
Number of member instances	198.5	53	1	3940

35 percent of the concepts have children. 264 of the concepts are used as type-restrictions in other concept definitions. The "members ratio" compares concepts' number of member instances to the number of members of each of its parents. Typically, specializing a concept results in excluding 80 or 90 percent of the parents' members.

Experiment 15 has a preponderance of value relations compared to type relations. In Section 7, we present some results from Experiment 16, in which this relative proportion is reversed. The typical concept in Experiment 16 is deeper and has more roles compared to Experiment 15.

6 Measures of Description Compatibility

In this section, we define measures of description compatibility between pairs of concept definitions. The functions that produce these measures vary with respect to their cost of computation, and the knowledge of the domain that they require. We present eight measures that we believe are interesting, out of a much larger number with which we experimented. In Section 7, we compare the output of the compatibility functions against SEMR, our measure of semantic overlap (Equation (2) in Section 3).

For presentation purposes, we organize the measures into three groups. Two "foundational" functions are inexpensive and universally applicable. Both utilize inheritance links in the ontologies, and one, SHRR (see Table 6) also considers the structure of the concept definitions. SHRR is surprisingly effective, as we will see in Section 7.

The "matching-based" functions build and evaluate one-to-one correspondences between elements of concept definitions represented as graphs. These measures require more calculation compared to the foundational measures. The advantage of matchings is that they support analyses of both similarities and differences in the syntactic structure of the compared definitions.

The "probabilistic" functions require domain-specific knowledge of the semantics of the elements of the concept definitions, which they use to calculate the likelihood that arbitrary instances in the domain will satisfy certain definitions. Thus, we characterize these functions as *semantics laden.* One of the probabilistic functions is also matching-based. In our simple simulation domain, we achieve very strong results with these functions (see Section 7). PSHR and PMHR are practical in the sense that, we believe, it will be possible to calculate them in real-world systems. The third (PCIR) makes unrealistic assumptions about the agreement of concept-definition semantics across differentiated ontologies. We include it for theoretical reasons because its derivation directly parallels our calculation of semantic overlap.

Table 6 summarizes the measures that we define in detail in the remainder of the section. The measure names are capitalized in association with their abbreviations, which we employ extensively. All of the abbreviations terminate with an "R" to distinguish these measures from their symmetric counterparts. These measures are appropriate for the request/recommendation scenario pictured in Figure 1; the "R" stands for "Recommendation". They all have some factor associated with either the source or the target concept, which in the symmetric versions of the compatibility measures is associated with both the source and the target.

Table 6: Summary of description compatibility measures

Measure name	**Measure basis**	**Requires matching**	**Semantics-laden**
Least-general Common Subsumer (LCSR)	Distance to least-general subsuming concept	No	No
SHared Rooted roles (SHRR)	Percent of shared rooted roles	No	No
eXponentiated DIFferences (XDIFR)	Exponentiated unmatched edges	Yes	No
Rooted ISOmorphism (RISOR)	Matching edges weighted by distance from root	Yes	A little
Rooted Isomorphism with Partial Differences (RIPDR)	Rooted match edge similarity, unmatched partial edge differences	Yes	A little
Probabilistic Shared-roles Heuristic (PSHR)	Conjoined parents	No	Yes
Probabilistic Matching Heuristic (PMHR)	Matching structure	Yes	Yes
Probabilistic Conjoined Intersection (PCIR)	Conjoined concepts	No	Yes (unrealistic)

None of our compatibility measures exploit concept or relation names that happen to be the same in different ontologies. We give concepts unique names for convenience; because our relations may have the same name, we must always specify a context in order to access them. Systems that maintain multiple ontologies usually define distinct namespaces by concatenating the ontology name with the concept or relation name (as in Stanford's library of Ontolingua ontologies (Farquhar, Fikes et al. 1996)).

6.1 Foundational Measures

Our first measure of description compatibility is a function of path distance to the most proximate concept that subsumes both of the concepts being compared (LCSR, for "Least Common Subsumer"). Let $L_1, \ldots L_n \in P$ be the set of concepts that subsume C and D. Let $|p(C, L_i)|$ be the number of links necessary to transverse the graph of inheritance links from C to L_i. Then

$$\mathrm{LCSR}(C, D) = \lambda^{\min_i (|p(C, L_i)|)} \quad , \qquad (4)$$

where $\lambda \in (0, 1)$ is an "attenuation factor" that models the expected percentage of instances in the denotation of a parent concept that are also members of a child's concept. In our simulations, λ is set to be the same as the percentage of prior members that must be distinguished from the target object (Table 4).

Measures of similarity based on path distance in a graph have a long history in artificial intelligence, dating back to spreading activation in semantic nets as proposed by Quillian (Quillian 1968). Our LCSR measure is more disciplined than spreading activation in the sense that all of the links must be inheritance relations, and this is generally true of the use of path-distance measures today (as in Bright et. al. (Bright, Hurson et al. 1994)).

Nevertheless, path-distance measures are fragile because they are sensitive to the degree of detail in the ontological structure. For example, defining the new concept B1 in Figure 3 decreases LCSR(A1, A2), although the meaning of these concepts has not changed. In our simulations, LCSR is quite accurate if we set the "thinning percentage" (Table 4) to be zero, since most concept specializations add a single role or constraint, but loses accuracy as the thinning percentage increases.

Our next measure quantifies the portion of definitional structure that is inherited from shared concepts (SHRR for "SHared Rooted roles"). "Rooted" roles are roles of the concepts being compared, as opposed to roles of other concepts in the definitions of the concepts being compared (the term comes from the representation of concepts as description graphs, although these graphs are not required to calculate SHRR). Let $L_1, \ldots L_n \in P'$ be the set of most specific concepts that subsume C and D, i.e., there is no i, j such that L_i subsumes L_j. Let $R(L_i)$ be the set of roles (relation plus filler) of L_i, and let $R' \subseteq \{R(L_1) \cup \ldots \cup R(L_n)\}$ such that no role in R' subsumes another. Let $|R'|$ be the number of roles in R', and let $|R_C|$ be the number of roles in the source concept C. Then

$$\mathrm{SHRR}(C, D) = |R'| \,/\, |R_C| \ . \qquad (5)$$

SHRR is especially interesting to us because it best captures the leverage provided by the assumption of differentiated ontologies. Like LCSR, it uses inheritance links to identify concepts that are the ancestors of both the source and target concepts. Unlike LCSR, SHRR also utilizes the definitions of the source and target concepts, and is thus a far more robust measure.

6.2 Matching-based Measures

The foundational measures identify correspondences between concept definitions due to structure inherited from common parents. Matching-based measures also identify correspondences between definitional structure that develops separately in differentiated ontologies. We call this additional isomorphism identified by matching *serendipitous correspondence*.

In this section, we first describe how we construct matchings. We then define several measures of syntactic correspondence that evaluate syntactic similarities and differences revealed by the matchings.

6.2.1 Building Matchings

A *matching* is a one-to-one correspondence between the concepts and relations of two concept definitions. To build matchings, we represent the concept definitions as directed graphs. This approach has two advantages: the algorithms are independent of the idiosyncrasies of particular concept representation languages, and we can utilize the large body of research on manipulating graphs.

We call the problem of finding an optimal matching Rooted Weighted-Edge-Match Graph Matching (RWEMGM). This problem is a restricted version of Weighted-Edge-Match Graph Matching (WEMGM) (Gold and Rangarajan 1996), which has been the focus of substantial research in the field of computational vision. Our solution for RWMGM, however, is most similar to algorithms for constructing analogies (Gentner 1990).

We represent concepts as *description graphs* (Borgida and Patel-Schneider 1994; Cohen and Hirsh 1994). We define the description graph for a concept C as:

$$G(C) = < V, E, v_0 , k > ,$$

where

- V is a set of vertices. Each vertex, $v \in V$, is associated with a concept d, v = < d >.
- E is a set of edges. Each edge, $e \in E$, is described as $e = < v_i, r, v_j >$, where $v_i, v_j \in V$ and r is a relation.
- v_0 is the vertex associated with the concept C, called the “root” of the description graph G.
- k is the maximum length of a path of edges from the root v_0 to any other vertex.

For every concept C and maximum path length k there is a unique description graph. To produce the graph we create a root node for the concept C, add edges for each role and constraint on role fillers, then recursively add subgraphs for each type restriction concept. If the path length from the root to a vertex reaches k, we halt the recursion. Figure 10 illustrates the description graph for the concept defined in Figure 9, COBJ00175, which is typical for our experiments with respect to size and complexity. The figure does not show inverse relations, which are present in the graph for every relation.

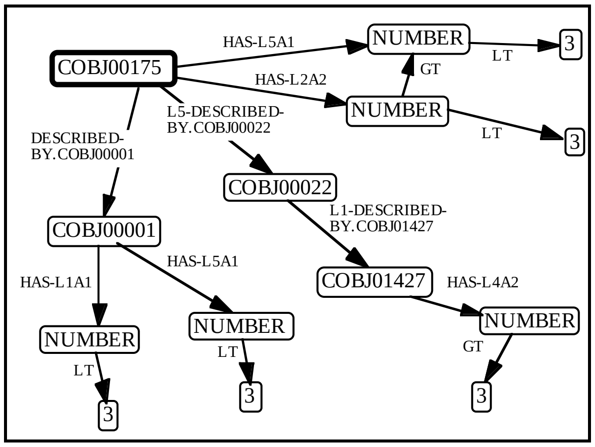


Figure 10: A description graph

We represent matchings using graphs of matches, where a *match* is a pairing of a concept (or relation) from the source graph with a concept (relation) from the target graph. We define a matching of two concepts as:

$$M(C, D) = < V_M, E_M, v_M, G(C), G(D) >$$

- V_M is a set of matches between vertices. Let H(C) and H(D) be subgraphs of G(C) and G(D), respectively. Let BV be a bijection between vertices, BV: $V_{H(C)} \rightarrow V_{H(D)}$. Then $V_M = \{v::BV(v), \text{ for all } v \in V_{H(C)} \}$, where "::" indicates a match.
- E_M is a set of matches between edges. Let BE be a bijection between edges, BE: $E_{H(C)} \rightarrow E_{H(D)}$. $E_M = \{e::BE(e), \text{ for all } e \in E_{H(C)} \}$.
- v_M is the root match, $v_0::BV(v_0)$.
- G(C) and G(D) are description graphs for the source and target concepts. We include these in the formal definition of matchings because analyses of differences between concepts are interesting both for elements that are matched, and those that are not matched.

Figure 11 illustrates a graph of matches in the matching between COBJ00175 and another concept, COBJ00223C. Again, inverse relations are not shown. This matching is also typical of matchings built in our simulation experiments.

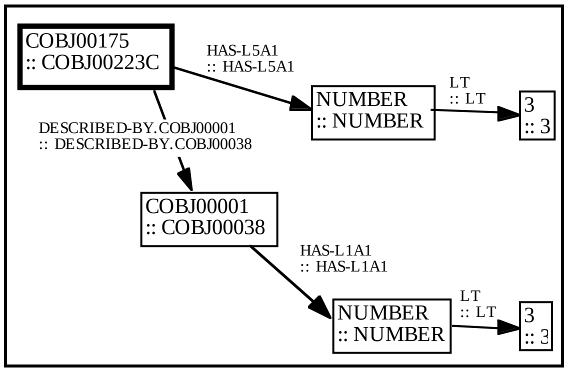


Figure 11: A graph of matches in a matching

The WEMGM (Weighted-Edge-Match Graph Matching) problem takes two graphs whose edges are more or less compatible with edges from the other graph, and determines the optimal one-to-one correspondence to maximize the total compatibility of matched edges (Gold and Rangarajan 1996). This problem is NP-complete, which can be proved by transformation from the Largest Common Subgraph problem (Garey and Johnson 1979). Given an instance of Largest Common Subgraph, let all edge compatibilities have weight = 1, and require total edge compatibility equal to the required number of isomorphic edges in the original. Many polynomial-time approximation algorithms exist for WEMGM, however (see Rangarajan (Rangarajan 1996) for a particularly comprehensive list).

Thinking that we would avoid reinventing the wheel, we used the "Graduated Assignment" algorithm (Gold and Rangarajan 1996) to generate sets of "soft" vertex matches (matches in [0,1] rather than {0, 1}), and the Hungarian method for maximizing network flow to convert the soft matches into discrete matches (Lawler 1976). Unfortunately, this approach has two drawbacks: graduated assignment is slow, and occasionally the resulting matches are not what we call "rooted".

A rooted graph of matches includes the root match, and is connected: every vertex can be reached by a path of edges from every other vertex. To motivate the need for rooted graphs, compare two edges constructed with natural-language concepts and relations:

E1: < BARBARA DRINKS WATER >

and

E2: < CHARLES SWIMS-IN WATER > .

The definition of "water" might expand to a sizable description graph containing all sorts of knowledge about water, but if we are interested in comparing Barbara and Charles, we must also consider their relation to water.

We define RWEMGM (Rooted Weighted-Edge-Match Graph Matching) as a decision problem in Figure 12. Condition (1) enforces isomorphism, condition (2) ensures that the graph of matches is rooted, and condition (3) seeks to maximize total compatibility. Condition (4) can be used to represent a preference for relatively compact solutions (more edges per vertex), which often have greater face validity as matchings.

Problem Instance

Graphs G=(V1, E1), H=(V2, E2).
A root match ($r1 \in V1$, $r2 \in V2$).
Non-negative real weights wij for every member of $M = E1 \times E2$.
Positive real T.
Positive integer N.

Question

Do there exist subsets $V1' \subseteq V1$, $E1' \subseteq E1$, $V2' \subseteq V2$, $E2' \subseteq E2$, such that:

(1) the subgraphs G'=(V1', E1') and H'=(V2', E2') are isomorphic: |V1'| = |V2'|, |E1'| = |E2'|, and there exists a bijection $f: V1' \rightarrow V2'$ such that $(u,v) \in E1'$ iff $(f(u),f(v)) \in E2'$.
(2) $r1 \in V1'$, $r2 \in V2'$, $f: r1 \rightarrow r2$, and G' and H' are each connected graphs.
(3) $W \geq T$, where W is the total sum of weights of the set $P' \subseteq M$, $P' = \{(u,v), (f(u),f(v))\}$ for all $(u,v) \in E1'$.
(4) $|V1'| < N$.

Figure 12: Rooted Weighted-Edge-Match Graph Matching (RWEMGM)

RWEMGM is also NP-complete, which can be proved by transformation from WEMGM. We provide a sketch. Given an instance of WEMGM, generate an instance of RWEMGM for every possible root match, $(r1, r2) \in V1 \times V2$. In the source and target graphs of each instance of RWEMGM, connect every vertex to the root if it is not already connected to the root. Let all of the compatibilities for these additional edges equal zero. WEMGM will have a solution if and only if one of the RWEMGM instances has a solution.

We use a greedy algorithm to produce high quality, but not necessarily optimal rooted matchings. This algorithm starts with the root match, and repeatedly extends the matching by adding the most-compatible edge match whose edges are incident on a match currently in the matching. If the new edge conflicts with other reachable edges, the matching is cloned and each inconsistent edge is added to one of the copies. Tractability is maintained by limiting the number of alternative matchings.

Figure 13 provides pseudocode for our greedy matching algorithm, including a key that describes the input arguments and the pseudocode variables. In the pseudocode, two edges are *compatible* if the function used to assign edge compatibility weights returns a positive weight. Two edges are *consistent* if their matches do not violate the requirement for a one-to-one correspondence between vertices of the matched subsets of the source and target graphs.

Greedy matching's time complexity is $O(|E|b^2)$, where $|E|$ is the number of edges in the smaller of G1 and G2, and b is the largest number of edges incident on a vertex in either G1 or G2. Our WEMGM approximation algorithm is $O(|E|^2)$. On a Sun UltraSparc (143 Mhz) running compiled Allegro Lisp, building 20,000 matchings with the WEMGM algorithm requires 60 hours. An equivalent run using greedy matching with a maximum of 20 alternative matchings requires 20 minutes.

Greedy matching is also a more accurate predictor of semantic overlap than WEMGM. In more than half of all comparisons, greedy matching identifies more edges connected to the root than WEMGM, and the reverse is true for less than one percent of comparisons. The benefit is not dramatic, however. Apparently WEMGM does identify the matches that matter most.

```
Greedy-matching (G1, G2, CI, f-eval-M, max-M)
initialize A := { (matching M with V={root match}, Q=Possible-extensions(M, root match, CI)) }
do until Q is empty for all M in A
    for each M with non-empty Q
            rank edges in Q, pick the best (B)
            identify edges in Q inconsistent with B (I)
    replace M with |I| matchings: for each edge e∈I, Add-edge-match(M, e, CI)
    rank matchings in A with f-eval-M; drop any with rank > max-M
return the most highly ranked matching

Possible-extensions (M, m, CI)
initialize X := ∅
for every edge (es) out of the source concept in match m
    for every edge (et) out of the target concept in match m
            if es and et are compatible and consistent with other edge-matches in M
    X := X ∪ {es::et}
return X

Add-edge-match(M, e, CI)
if the second match in e (v) is new, add v to V
remove edges inconsistent with e from Q
add Possible-extensions (M, v, CI) to Q
add e to E
return M
```

Key

G1, G2:	The source and target description graphs
CI:	"Compatibility Instructions", including contexts for accessing concepts and relations in the source and target graphs, and functions for evaluating edge, concept, and relation compatibility.
f-eval-M:	A function for evaluating the strength of a matching
max-M:	The maximum number of alternative matchings
A:	The set of alternative matchings
M:	A matching in A
V:	The set of vertices in a matching M
E:	The set of edges in a matching M
Q:	A queue of edge-matches that may be added to a matching M

Figure 13: Greedy matching algorithm

The functions used to assign edge compatibilities determine the results of the matching algorithm. These include functions for estimating edge compatibility, concept compatibility, and relation compatibility.

We calculate the compatibility score of a non-numeric edge-match, S(em), as:

$$S(em) = S(rm) * S(v2m) \qquad (6)$$

where S(rm) is the compatibility score of the relation match, and S(v2m) is the compatibility score of the second vertex match. The edge-match includes two vertex matches and the relation match. The compatibility of the first vertex match is not scored because this match is already part of the matching. Equation (6) is also appropriate when evaluating (rather than constructing) matchings, as long as evaluation proceeds outward from the root.

We evaluate compatibility between edges that represent numeric constraints as:

$$S(ne) = |D(ne1) \cap D(ne2)| \; / \; |D(ne1) \cup D(ne2)| \, , \qquad (7)$$

where D(ne1) is the domain of finite values that satisfy the numeric constraint in the source concept, and D(ne2) is the same for the target concept.

To estimate concept compatibility, we can use any of our description-compatibility measures. In Section 7, we will compare matchings built using the LCSR, SHRR, RISOR, and PCIR measures introduced in Table 6.

We estimate relation compatibility with a modified version of the LCSR compatibility measure. First, we do not permit matches between relations that are specific to different clusters. This modification of the standard LCSR algorithm compensates for inaccurate modeling of our experimental domains by the ontological structure. For example, the relations HAS-L2 and HAS-L3 are both direct children of the HAS relation, and thus, according to LCSR, are highly compatible relations. In our simulations, however, attribute values are not correlated across clusters. It is therefore undesirable to identify correspondences across clusters, since none exist. Second, the relation-compatibility function adjusts the LCSR measure to encourage matches that specify both attribute and cluster over those that specify cluster only.

Finally, we use a very simple function to rank alternative matchings during greedy matching: we sum the compatibilities of the edge-matches in the matching. It would also be possible to use more elaborate evaluation functions for this purpose, including analogs of all of the measures in Table 6.

6.2.2 Evaluating Matchings

The simplest of the matching-based measures focuses on syntactic differences between the concept definitions: in particular, on those edges in the source and target that are not included in the matching. Each unmatched edge diminishes the probability that an arbitrary instance will be a member of both the source and target concept:

$$XDIFR(C, D) = \gamma^{|E(G(C))| - |E(M(C, D))|} \qquad (8)$$

where E(G(C)) is the set of edges in the description graph G for concept C, E(M(C, D)) is the set of edges in the matching between C and D, and $\gamma \in (0, 1)$ is a constant chosen empirically to maximize the association of XDIFR with SEMR in a sample of concept comparisons. γ is exponentiated by the number of edges in the source and the target graphs that are not matched.

The next matching-based measure evaluates correspondences between the structure of the source and target definitions. The RISOR (for “Rooted ISOmorphism”) measure focuses on syntactic similarities, but also models certain differences that can be identified in the context of the similarities:

$$\mathrm{RISOR}(C, D) = (\Sigma_e\, W(e)S(e)) * Q(M(C, D)) / |E(G(C))| \;. \qquad (9)$$

where e is an edge match between concepts C and D, S(e) is the evaluation score of the edge match (see Equation 6), and W(e) is a weighting based on its proximity to the root match. The intuition is that the closer an edge match is to the root match, the more important it is as far as indicating semantic overlap. Q is a function that identifies and penalizes inconsistent numeric constraints. The sum of the weighted edge-match scores is multiplied by the penalty, and divided by the number of edges in the source description graph G(C).

Edge evaluations are weighted by their proximity to the root match, taking into account that multiple paths may connect an edge to the root:

$$W(e) = \min (1, \Sigma_{p\in H(r,v1(e))} (\Pi_{e\in p}S(e))^{\alpha}) \qquad (10)$$

where each p is a member of the set of paths H from the root r to v1(e), the first match of the edge e; $e \in p$ are the edges in each path, and α is a parameter that adjusts the degree to which proximity to the root affects the evaluation. In RISOR, $\alpha = 1$.

The function Q penalizes matchings where inconsistent numeric constraints are put into correspondence. For example, if two concepts are matched and the first has a filler that must be greater than three, while the second has a corresponding filler that must be less than three, then (to the extent that the two fillers must correspond) no instances will be members of the intersection. Q is a function of the matching between C and D, because it is the one-to-one correspondence between the elements of C and D's definitions that permits the inconsistency to be identified:

$$Q(M(C, D)) = \Pi_{v\in Vq(M(C, D))} \min (1, \Sigma_{p\in H(r,v)} (\Pi_{e\in p}S(e))^{\alpha}) \qquad (11)$$

where $V_q(M(C, D))$ is the set of node matches in the matching M(C, D) where there is a contradiction among the numeric constraints incident on the match. The penalty is weighted by the proximity of the offending node match to the root in the same manner as are edge matches in Equation (10).

We could calculate any number of heuristic measures similar to RISOR. For example, setting $\alpha = 0$ in Equation (10) produces an unweighted estimate (that performs very poorly); other values such as $\alpha = 0.75$ or $\alpha = 2$ produce estimates that are slightly less accurate than RISOR. Dividing by the number of edges in the source concept (the denominator in Equation (9)) accounts for differences in definitional structure, and has the advantage that it does not introduce a free parameter such as γ in Equation (8). It makes more sense, however, to use an exponentiated factor to represent differences, since each additional unshared edge contributes proportionately, on average, to diminish the probability of common member instances.

Our objective when designing the next matching-based measure of syntactic correspondence was to squeeze as much information out of the matchings as possible. RIPDR ("Rooted Isomorphism with Partial Differences") combines the weighted rooted numerator of RISOR, including the penalty function, with an exponentiated factor to represent unmatched edges, with the added twist that partially matched edges are included in the exponentiated factor:

$$\mathrm{RIPDR} = (\Sigma_e\, W(e)S(e)) * Q(M(C, D)) * \gamma^{|E(G(C))| - \Sigma i=0to3\beta i|E_i(M(C, D))|} \;. \qquad (12)$$

γ, W(e), S(e), and Q are defined as in Equations (8), (10), (6), and (11), respectively. The $E_i(M(C,D))$ include edge matches that include from zero to three perfect concept and relation matches, and β_0, β_1, β_2, and β_3 are 0.25, 0.50, 0.75, and 1.0, respectively.

We consider RIPDR to be our rococo measure. It is certainly possible to be even fancier, but the return for such efforts rapidly diminishes.

6.3 Probabilistic Measures

The probabilistic compatibility measures require knowledge of the semantics of relations in concept definitions and attribute values in the domain. Using this knowledge, we can estimate the probability that an arbitrary instance is a member of a concept. This capability can be utilized in several interesting ways to construct potent measures of syntactic correspondence.

The procedure for estimating the probability of concept membership represents the concept as a description graph without cycles, then adjusts the estimate for each edge in the graph, and for each edge excluded from the graph to prevent cycles. In our simulations, for example, if a graph includes the relation HAS-L2A3, we multiply the estimated probability by 3/5 because that percentage of objects have values in cluster 2. For the edge <NUMBER GT 3> we multiply by 14/45 because that percentage of values are greater than 3.

More generally, we exploit the following knowledge about the generation of objects and concepts:

- Knowledge of the structure of objects and concepts:
  - That all attributes in 3 of 5 clusters have values.
  - That each attribute can have only one filler.
  - That cycles in concept description graphs include exactly three relations.
- Empirical data on:
  - The frequencies of values.
  - The frequencies of values conditioned on known values of other attributes within a cluster.

With this knowledge, our estimates of the probability of concept membership, $\hat{P}(j \in C)$, achieve very strong correlation with empirical probabilities of concept membership: $r(\hat{P}(j \in C), |C^I| / |\Delta|) \approx 0.90$. (The correlation is often higher, but can be lower depending on the sample of concepts). In realistic systems, we doubt that it will be possible to calculate concept membership probabilities with equal accuracy, because our simulated universe is far more orderly than the real world, and we know how it is constructed. It may be possible, however, to construct useful semantic-laden estimates in realistic systems, given:

- Ontologies where relations are defined in association with rules for calculating their impact on membership probability; both independently, and when occurring with other relations.
- A model of the joint-probability function of the occurrence of terminal attribute values (for example, a Bayes net).

We define our first probabilistic compatibility measure, PSHR for “Probabilistic Shared-role Heuristic”, as:

$$\mathrm{PSHR}(C, D) = \hat{P}(C) / \hat{P}(A) \qquad (13)$$

where A is a concept defined as the conjunction of all concepts that are ancestors of both concepts C and D. Let $L_1,...L_n \in P'$ be the set of concepts that subsume C and D, with $A \equiv (L_1 \text{ AND } L_2 \ldots \text{ AND } L_i)$. Figure 14’s Venn diagram illustrates this calculation. It may not seem intuitive when compared to SEMR (Equation 2). Consider, however, that the more shared structure inherited by C and D, the more closely A^I will encircle C^I and D^I; and in the limiting case where C = D, A = C will also be true.

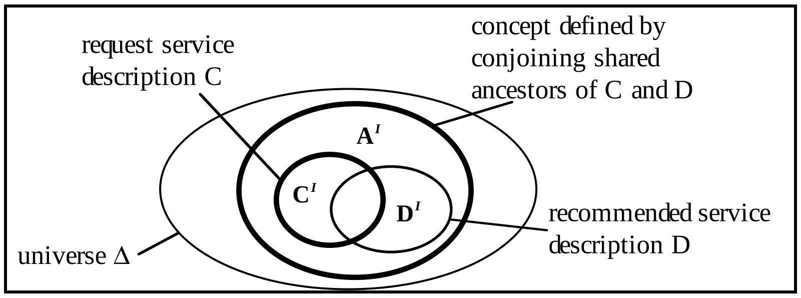

request service description C
concept defined by conjoining shared ancestors of C and D
A^I
C^I
D^I
recommended service description D
universe Δ

Figure 14: The semantic basis of PSHR

The next probabilistic compatibility measure is analogous to PSHR, but uses a matching to identify correspondences between the source and target concepts. We also include penalties for inconsistent numeric constraints (Q, as defined by Equation (11)), since these are easily derived from a matching and have a significant effect (see Section 7). Thus, PMHR, for “Probabilistic Matching Heuristic” is defined as:

$$\mathrm{PMHR}(C, D) = \hat{P}(C) * Q(M(C, D)) / \hat{P}(B) \qquad (14)$$

where B is a description graph constructed from a matching M(C, D) where each node match and relation match is converted to a concept-definition node or relation edge, respectively. To collapse a matching graph to a description graph, we:

1. Convert each relation match to the least-general relation that subsumes the matched relations. For numeric constraints, we construct a constraint whose range is the union of the matched constraints. For other relation matches, we rise in the taxonomy of relations until we reach a shared ancestor.
2. Convert each concept match to a node in the new graph. We do not need to generalize matched concept definitions, because all information in the concept definitions is present in the graph’s edges.

Compared to PSHR, PMHR includes both shared inherited structure and serendipitous correspondence. Therefore, with very few exceptions $\hat{P}(B) \le \hat{P}(A)$, and if there is no penalty (Q(M(C, D)) = 1), PMHR(C, D) ≥ PSHR(C, D). The few exceptions occur if a match that is not inherited is preferred instead of an inherited match, such that the preferred match must be generalized when the matching graph is collapsed to the description graph. In these unusual cases, the generalized edge may diminish the concept membership probability estimate less than the alternative inherited match that is incorporated into the PSHR estimate.

Our final probabilistic compatibility measure is a translation of the definition for SEMR (Equation 2). We define PCIR, “Probabilistic Conjoined-Intersection”, as:

$$\mathrm{PCIR}(C, D) = \hat{P}(C \wedge D) / \hat{P}(D) \quad . \tag{15}$$

where ($C \wedge D$) is a concept defined as the conjunction of C and D.

PCIR is an extremely accurate measure (see Section 7). Unfortunately, it is realistic only for source and target concepts in the same ontology, because it is highly unlikely that any model of the joint distribution of values, or rules for the interaction of relations, would be available for attributes and relations defined in differentiated ontologies.

7 Results

This section reports the performance of the syntactic correspondence measures in comparison to SEMR, the measure of semantic overlap appropriate for agent requests and recommendations. We first present results that support our fundamental premise that differentiated ontologies can support syntactic measures that are clearly associated with semantic overlap. Second, we focus on the matching-based measures to determine if, how, and when matchings contribute substantially to prediction accuracy. Finally, we analyze the results in terms of agent requests and recommendations to investigate the potential usefulness of description compatibility measures for guiding agent search for services.

Unless noted otherwise, these results are for Experiment 15 (see Section 5). This experiment includes over 20,000 concept comparisons, generated for source and target concepts from 42 pairs of differentiated ontologies. The set of ontology pairs is designed to have a broad range of variation in the number and ratio of communities that are shared and not shared. For each ontology pair, each concept in the source ontology is compared to concepts in the target ontology that share at least some inherited definitional structure, in accordance with Assumption (2) about differentiated ontologies (see Section 2). Matchings are built with our greedy matching algorithm (Section 6.2), using SHRR as the concept compatibility function, and with a maximum of five alternative matchings.

To reduce noise from small numbers, we accumulated membership and concept intersection counts for 500,000 instances. This was accomplished by repeatedly repopulating the simulation with new sets of instances and saving the counts to disk. To reduce noise further, only comparisons where both the source and target concept have at least 50 instances are included, thus excluding approximately five percent of the comparisons.

In studies of naturally occurring data, the statistical significance of estimates is of central interest, but this is not true here. All of the correlations in the following tables are highly significant, typically to less than one tenth of one percent. In general, the number of digits shown is limited for display purposes, rather than to indicate statistical precision. On the other hand, we cannot measure the fidelity of the simulation with respect to the real world. Thus, we disregard fine distinctions in the numbers, and concern ourselves only with the most salient results.

7.1 Differentiated ontologies support estimation of semantic overlap

In this section, we describe the association between our measures of syntactic correspondence and the SEMR semantic measure.

Table 7 lists the first three moments of the distributions of the semantic and syntactic correspondence measures. Two observations are of interest. First, the mean SEMR is fairly small, and the skew is strongly positive. This shows that most pairs of concepts are not very similar. Typically, approximately six percent of the instances of a recommendation also satisfy the request. Secondly, the distributions of XDIFR and the probabilistic measures PSHR, PMHR and PCIR are quite similar to that of SEMR. This is appropriate, since SEMR and XDIFR are also probabilistic. The other measures would need to be transformed to approximate SEMR's distribution.

Table 7: Distributions of the semantic and syntactic correspondence measures

N=21,815 Method	Mean	Standard deviation	Skew
SEMR	.059	.155	4.45
LCSR	.473	.153	0.13
SHRR	.332	.152	2.00
XDIFR	.051	.161	4.94
RISOR	.128	.107	1.18
RIPDR	.105	.466	8.83
PSHR	.063	.141	4.89
PMHR	.066	.148	4.56
PCIR	.059	.155	4.42

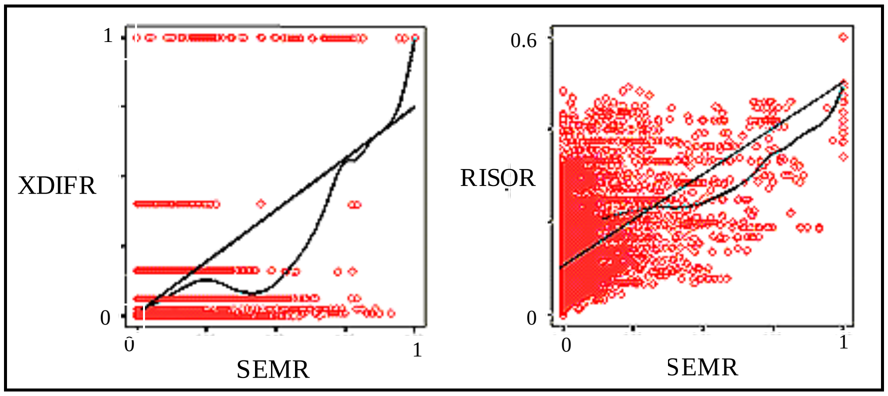
Table 8 describes the relative performance of the syntactic correspondence measures with two correlation statistics:

- Pearson's: the standard measure of correlation based on sums of squared distance from the means, and
- Spearman's: which applies Pearson's equation to the ranks of the observed data (Mendenhall and Scheaffer 1973).

Unlike Pearson's correlation, ranked correlation is not sensitive to extreme data points. Figure 15 demonstrates why ranked correlation is more appropriate for our data. Figure 15a show a scatterplot of comparisons located by SEMR and XDIFR. The slope of the straight linear regression line equals Pearson's correlation, but the data clearly are not linear, and the linear regression is often distant from the curved local regression line. Compared to RISOR, which is plotted in Figure 15b, XDIFR's Pearson correlation is higher, but its ranked correlation is much lower. Therefore, although the Pearson's correlations are usually larger, we will use ranked correlations for all subsequent analyses.

Table 8: Accuracy of the syntactic correspondence measures

N=21,815 Method	Correlation with SEMR	Ranked correlation
LCSR	0.413	0.213
SHRR	0.635	0.380
XDIFR	0.718	0.176
RISOR	0.573	0.485
RIPDR	0.726	0.426
PSHR	0.892	0.559
PMHR	0.952	0.706
PCIR	0.989	0.963



(a) **(b)**

Figure 15: Scatterplots of XDIFR vs. SEMR (a) and RISOR vs. SEMR (b)

Focusing on the ranked correlations in Table 8, we observe that LCSR is fairly weak, SHRR is stronger, the matching-based measures that identify definition isomorphism are yet stronger, and the probabilistic measures are quite strong.

The correlation between SHRR and SEMR is a succinct summarization of the relationship between differentiated ontologies and estimating semantic overlap. The greater the proportion of shared inherited structure compared to all definitional structure, the greater the expected semantic overlap. Figure 16 shows a second-order effect: if samples of comparisons are filtered by a test for shared inherited structure, there is a noticeable increase in the predictive accuracy of the compatibility measures. The filter test used for this figure, NSHRE (Number of SHared Rooted Edges) is similar to the SHRR measure, but includes inherited constraints as well as roles, and does not divide by the source's rooted roles as in Equation (5) (SHRR does not include constraints because these are not necessarily linked directly to the root). The increase in predictive accuracy occurs for NSHRE >= 3. The increase persists even with the removal of the significant minority of cases where the request subsumes the recommendation (SEMR = 1).

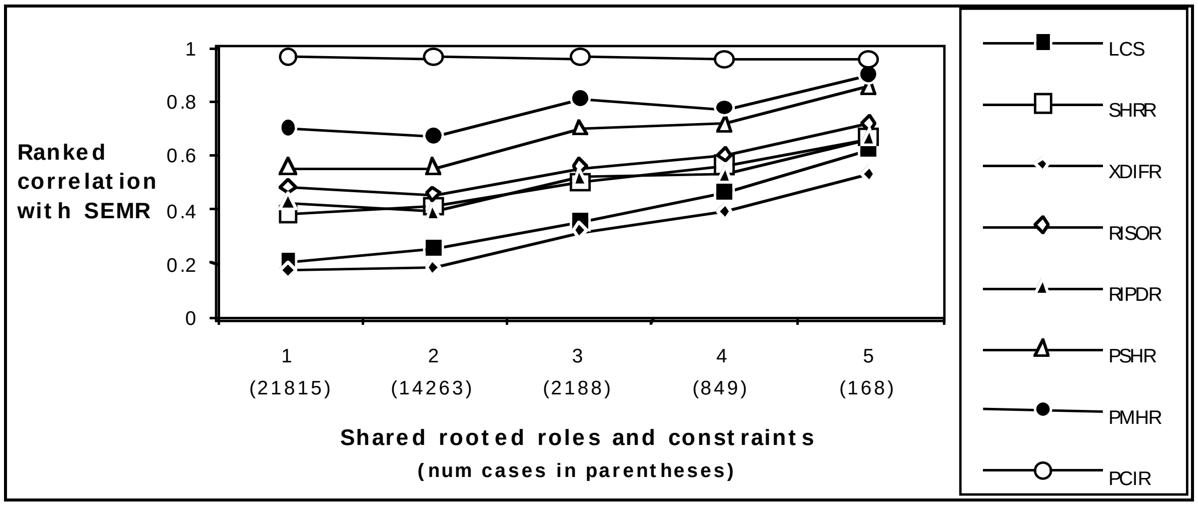


Figure 16: Moderate increase in accuracy with increasing shared inherited structure

7.2 Matchings can increase prediction accuracy

To evaluate the contribution of matchings to prediction accuracy, we need to control for information utilized by the compatibility measures apart from the matchings. Thus, we compare RISOR to SHRR, and PMHR to PSHR. Table 9 shows that the matching-based measures do add significant accuracy, by predicting a ranked SEMR with a two step linear regression, and testing the increase in explained variance.

Table 9: Significant improvement in prediction of ranked SEMR using matchings

Independent variables in a two-step regression	**Variance explained by the model (R square)**	**Change in F statistic (df 1, 21812)**	**Change level of significance**
SHRRK with RISORK	.144 .261	3442	0.000
PSHRK with PMHRK	.312 .498	8106	0.000

The source of the increase in predictive accuracy in this experiment, however, is almost entirely the identification of inconsistent numeric constraints (Equation 11). Table 10 includes measures that are versions of RISOR and PMHR without the penalty Q (NPRISOR and NPPMHR, respectively), and the correlations for these measures are less strong than for their non-matching-based counterparts. We need to make two points in this regard, however. Firstly, without Q, matches between nodes with inconsistent constraints actually increase NPRISOR and NPPMHR, thus reducing the accuracy of these measures. Secondly and most importantly, the *identification of positive correspondences in matchings enables subsequent identification of differences*. Thus, it is appropriate that the matching-based measures include Q, whereas the non-matching measures do not.

Table 10: Ranked correlations to SEMR with and without penalties for inconsistent constraints

Semantics-free	**SHRR** .380	**RISOR** .485	**NPRISOR** .166
Semantics-laden	**PSHR** .559	**PMHR** .706	**NPPMHR** .537

We anticipated that the high correlations between attribute values in our simulations would be an important source of order that could be exploited by matchings. This turned out not to be the case, primarily because less than one in five matchings includes a match between different attributes. For matchings that do match different attributes, the value correlations are important for measures that properly model these correspondences. Table 11 shows the association of RISOR to SEMR for four samples that vary by the cases included, and the covariance of attribute values in the universe of objects. When there is strong covariance of attribute values within clusters, RISOR retains predictive accuracy for cases that include matches across attributes, but when attribute values are independent it loses most of its predictive power. (There are fewer cases in the independent attributes column because we classified only 200,000 thousand instances for this experiment, causing more cases to be excluded for small numbers.)

Table 11: The effect of attribute value covariance on the predictive accuracy of RISOR

Case selection	Value covariance (number of cases in parentheses)	
	High (see Table 3)	None
All	.485 (21815)	.435 (18151)
With different attributes matched	.464 (4052)	.203 (3091)

Counter to our intuition, an equivalent table for PMHR would show that this measure is actually slightly more accurate when attribute values are independent. To understand this requires intimate knowledge of the way PMHR handles matches across different attributes: it generalizes these matches to attribute-independent HAS-Lx roles, with the effect of tending to underestimate semantic overlap in these cases. This results in better performance for the independent values scenario, which tends to have smaller semantic overlaps.

The intricacies of the PMHR algorithm are less important than the following general lesson about matchings. *The benefit of matching depends on the accuracy of the system's partly explicit, partly implicit model of the domain.* Elements of the model are declaratively encoded in the ontological structure. Other elements are procedurally encoded in compatibility functions for edges, concepts, and relations. For example, our top-level ontology accurately models attribute value covariance because HAS-LxAy inherit from HAS-Lx relations. When attribute values do not covary, this ontological model is inaccurate, since it indicates correspondence when none exists. Similarly, the inheritance of the HAS-Lx relations from HAS harms predictive accuracy because values do not covary across clusters. We correct this modeling inaccuracy by prohibiting cross-cluster relation matches in our relation compatibility function.

We illustrate this fundamental issue about modeling accuracy by comparing our results for Experiment 15 to another experiment with a substantially different ontology structure (see Table 12). In Experiment 16, there are proportionately far more type roles (in which the fillers are other instances) rather than value roles (in which fillers are numbers). This results in bigger description graphs, and a corresponding increase in the size of the average matching graph. Experiment 16 also has fewer opportunities for matches subject to contradictory constraints, only because such contradictions apply more frequently to value matches in our experiments. This characteristic is not inherent to value matches—many ontologies define concepts to be disjoint, which would yield the same effect for type matches.

Table 12: Predictive accuracy in two experiments. 16's definitions are bigger and matches are subject to fewer contradictions.

	Experiment 15 (N=21,815)	Experiment 16 (N=14,614)
Proportion of value roles / type roles	1.20	.57
Avg. edges in source concept	26.0	35.9
Avg. edges in matching	8.4	11.5
Avg. numeric constraint penalty	.810	.926
Ranked correlation with SEMR:		
SHRR	.380	.486
RISOR	.485	.478
RIPDR	.426	.489
PSHR	.559	.679
PMHR	.706	.762

Most of the findings that we have identified for Experiment 15 also hold for Experiment 16. In general, the compatibility measures have greater predictive accuracy in Experiment 16. In particular, SHRR improves most dramatically, to the point where it is more accurate than RISOR; we believe this is due mostly to the relative infrequency of contradictions in Experiment 16. In Experiment 15, the value added by matchings is due mostly to the identification of inconsistencies; in Experiment 16 inconsistencies are much less prevalent, thus the matching-based measures do not make the substantial gains that might otherwise results from the increased size of concept description graphs in Experiment 16.

Fortunately, the accuracy of matching-based measures is quite robust with respect to the concept compatibility function used to construct the matching. Table 13 shows correlations to SEMR for all of the matching-based measures for experiments where the matchings are constructed using alternative measures for judging the compatibility of matches between concepts in the source and target definitions. Note that LCSR yields larger matchings than does SHRR. This occurs because all pairs of concepts have a common parent (Service), but frequently do not share inherited structure. Indeed, the LCSR matching-based measures are slightly more accurate than the SHRR-based measures. We prefer to use SHRR nevertheless, because it is far more robust than LCSR (see Section 6.1). RISOR and SHRR-based measures are equally accurate, and since RISOR is more expensive SHRR is better for this purpose. Even using PCIR to judge concept compatibility does not yield much improvement. Although the matchings do depend on the concept compatibility function, it appears that we find the matches that matter with any of our measures.

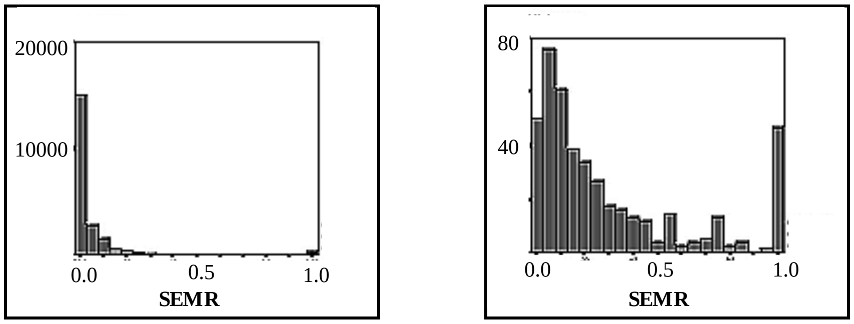
Table 13: Ranked correlations to SEMR for matching-based measures with alternative measures used for matching construction

Cases: 21,815	**Measure used for matching construction and submatch evaluation (mean matching edges in parentheses)**			
Evaluation measure	**LCSR (9.63)**	**SHRR (8.36)**	**RISOR (9.00)**	**PCIR (10.40)**
XDIFR	.204	.176	.187	.214
RISOR	.493	.485	.480	.487
RIPDR	.434	.426	.429	.435
PMHR	.707	.706	.707	.708

7.3 Description compatibility can help agents find services

In this section, we frame our analysis in terms of the model of agent interaction pictured in Figure 1. We call the set of comparisons that include a particular source concept a “search”. For each search, we rank comparisons by their SEMR value to obtain an ordered set of “candidate” services. We rank comparisons by each of the compatibility measures to obtain ordered sets of “recommended” services. We then characterize the usefulness of the compatibility measures in terms of the rankings of the recommended candidates. There are several ways to rank comparisons that have equal values, and unfortunately, the method used can have a substantial impact on the reported results. Therefore, we rank comparisons in several ways and combine the results as needed to answer questions of practical interest.

Experiment 15 includes 490 searches, averaging 53 comparisons per search. Of these, we drop searches with less than 10 comparisons, leaving 414 searches. Figure 17(a) shows that the large majority of comparisons are quite poor, with very small semantic overlaps. The distribution of semantic overlap for the best candidates is much more uniform, of course, but the average semantic overlap is still only 0.3, and the median is lower.



(a) (b)

Figure 17: Histograms of SEMR, for all cases (a) and for best candidates (b)

Table 14 shows the probability of recommending the best candidate on the first try. These probabilities are composed of two mutually exclusive parts: the probability that a single comparison has the highest score and that it identifies the best candidate, and the probability that a comparison randomly selected from those tied for first is the best candidate. From this perspective, measures that have fewer ties do better. Thus, the probability of recommending the best candidate on the first try is relatively low for SHRR and PSHR (see Equations (5) and (13) to understand why these measures do not distinguish well between comparisons within a search). The performance of RISOR and PMHR, on the other hand, is quite strong: the agent has close to a 50% chance of trying the best candidate first using RISOR, and substantially better than a 50% chance with PMHR. As usual, PCIR is the strongest, but we are interested in this measure only as a point of reference since it requires unrealistic assumptions.

Table 14: Probability of recommending the best candidate on the first try

N=414 searches	Prob. 1st try is best	Prob. one mapping is a single best	Prob. single best is best	Prob. tied for best is best	Average number tied for best
LCSR	0.172	0.167	0.797	0.047	26.00
SHRR	0.283	0.256	0.708	0.136	8.06
XDIFR	0.406	0.454	0.516	0.314	3.16
RISOR	0.467	0.495	0.600	0.337	2.79
RIPDR	0.453	0.514	0.577	0.321	2.52
PSHR	0.321	0.333	0.659	0.151	6.92
PMHR	0.595	0.510	0.739	0.446	2.54
PCIR	0.656	0.886	0.616	0.968	1.57

Figure 18 shows that the relative accuracy of the first recommendation is quite stable if we consider other candidates besides the best. The chance that the first PMHR recommendation will be in the top 10% of ranked candidates is nearly 80%, and so on. (The data series markers along the x axis identify the percent of comparisons that are the best for each search, those within the 10th percentile, the 20th, and so on—because of ties, the actual number of cases in each percentile interval varies).

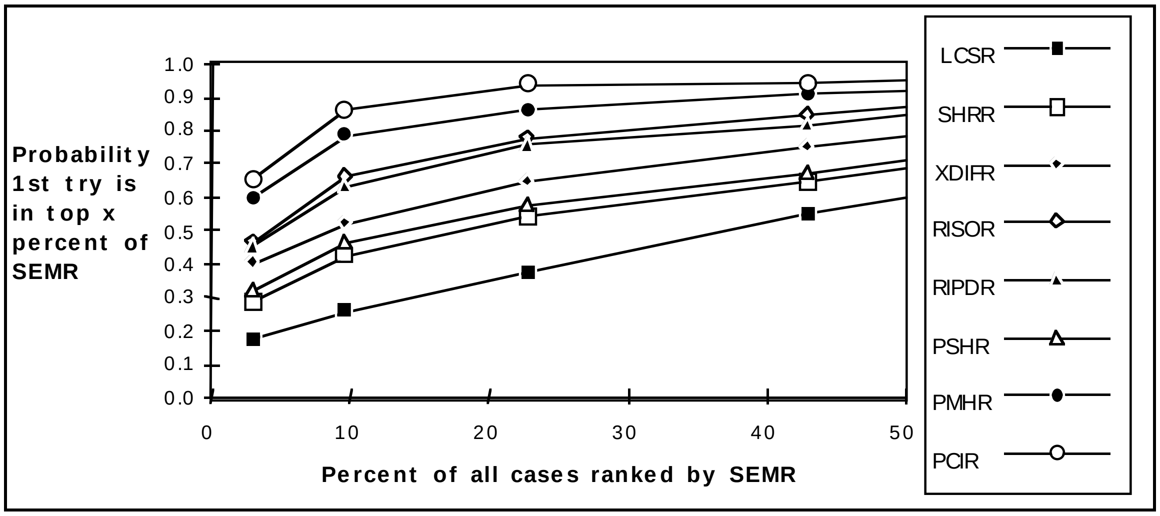


Figure 18: Cumulative probability that first recommendation is of certain accuracy

Figure 19 shows that the degree of semantic overlap available among the candidates is associated with the probability that the top recommendation will be accurate. The dip in the middle of the figure, which is especially pronounced for RISOR and RIPDR, is intriguing, but should not be considered definitive: the number of searches in each percentile interval is only about 40, and we do not see a similar dip for Experiment 16. We interpret Figure 19 as indicating that the accuracy of all of the measures is somewhat lower for searches with only very poor candidates, higher for searches with a very strong (subsumed) candidate, and more or less equally accurate searches with candidates in the range from fair to good. (Note, the comparisons included in this figure include only those with a single best SEMR, so the data is not entirely consistent with Figure 18).

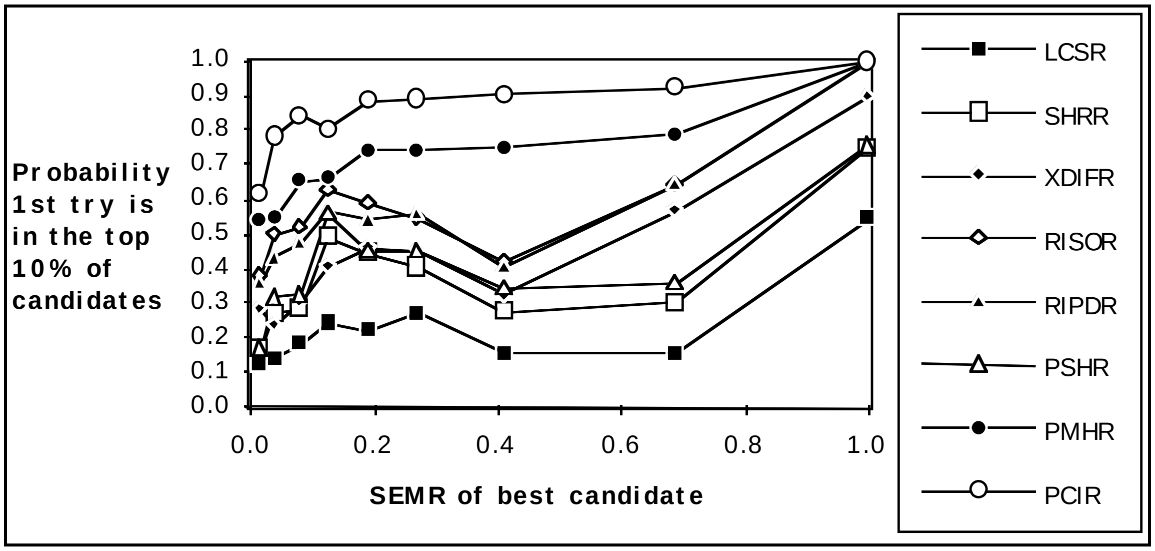


Figure 19: Relation of expected quality of first recommendation to the semantic overlap of the best candidate

Figure 20 describes the number of recommendations that an agent would expect to try until finding the service that is actually the best candidate. The position of the first data-series markers along the x axis identifies the percentage of comparisons tied for first according to each measure; the next marker identifies the 10th percentile cutoff, the next the 20th, and so on up to the 50th percentile (the actual number of cases in percentile intervals varies widely for the different measures). Again, PMHR and RISOR are the best, excluding PCIR. Trying approximately 15% of the recommendations gives the agent better than an 80% chance of finding the best candidate with PMHR, and a 68% chance with RISOR.

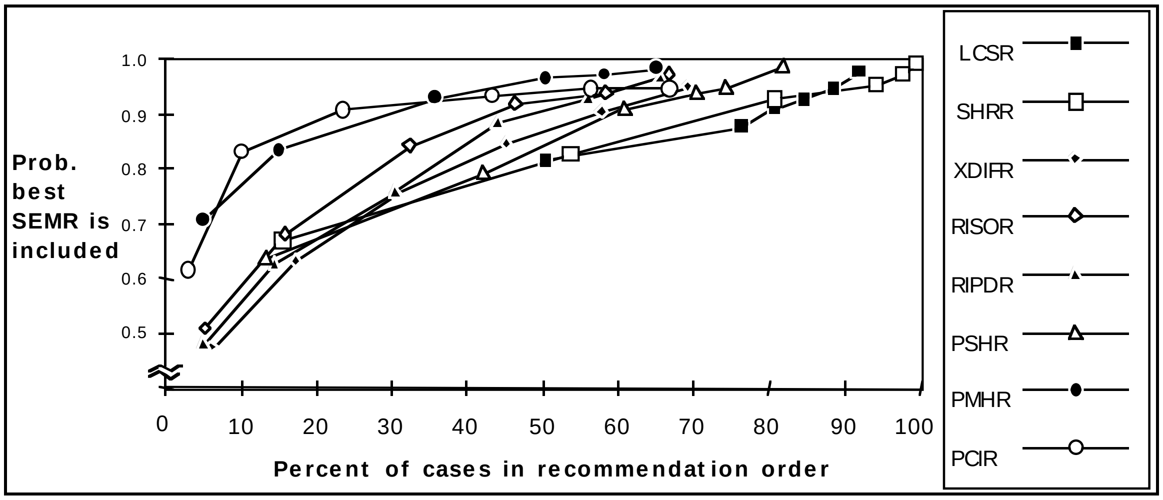


Figure 20: Cumulative probability of finding the best candidate

In Figure 16, we saw that filtering comparisons to require increased shared inherited structure results in higher associations between the description compatibility measures and semantic overlap. Such filtering would not be desirable for agents searching for services, however, unless there are many satisfactory candidates, because of the undesirable exclusion of candidates that are strong because of serendipitous correspondence. Figure 21 shows the distribution of SHRR for the best candidates; the mean is 0.5, and the median is lower.

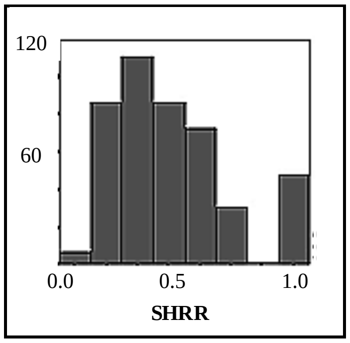


Figure 21: Histogram of SHRR for best candidates

8 Discussion

Developing general techniques to achieve semantic interoperability is an extremely difficult problem, profoundly rooted in the nature of the world and communication using symbols. Therefore, there has been little research that tackles semantic heterogeneity straight on. Some papers characterize the problem and explain why it is hard (Heiler 1995), or analyze the ways in which semantic heterogeneity manifests in heterogeneous schemas (Kim and Seo 1991) or ontologies (Visser, Jones et al. 1997). Many papers, especially in the area of data integration, dance around the problem. A few authors make bold claims about the power of their technique to overcome semantic heterogeneity, but these inevitably conceal some requirement that the problem essentially be solved outside rather than within the processing.

We divide this section into two parts. First, we review research that is related to ours in the areas of agent communication and integration of heterogeneous data. Second, we offer a critique of our own work.

8.1 Related work

Several multi-agent systems have been developed where agents advertise by describing their capabilities in ontological structures, enabling broker agents to recommend services to match requests. In the InfoSleuth project at MCC (Bayardo, Bohrer et al. 1996; Nodine and Unruh 1997) the ontologies are highly self-descriptive: the descriptions talk about themselves as frames. This enables translation between alternative syntaxes that support different types of reasoning. Term semantics, however, are assumed to be shared. In the Service Markets Society of the University of Michigan Digital Library (Durfee, Mullen et al. 1998), description logic is used to organize service descriptions in a subsumption taxonomy. This helps brokers reason in sophisticated ways to match requests, and leads to interesting system behavior. For example, agents automatically switch to services provided by new agents if the new agents' service descriptions match their requests more closely than those of existing agents. Again, however, semantic heterogeneity is avoided; in this case by assuming that all agents subscribe to the same set of ontologies. The Kraft project (Pazzaglia and Embury 1998) uses ontologies that focus on constraints. Current work has focused on characterizing the problem of semantic heterogeneity (Visser, Jones et al. 1997) rather than solving it.

Most research on the integration of heterogeneous data skirts the issue of semantic heterogeneity, although the titles of some papers suggest a more direct confrontation. For example, Milo and Zohar (Milo and Zohar 1998) describe a system to semi-automate the generation of wrappers that translate queries from one schema to another. Knowledge of term semantics, however, must be embedded in the rules that drive matching between the schemas. In the SIMS project (Ambite and Knoblock 1995; Knoblock and Ambite 1997), every database is represented by an agent that models its data with a description logic ontology, that can be mapped using "query reformulation operators" to ontologies that model databases represented by other agents. Again, knowledge of term semantics is embedded in the initial ontological models and in the reformulation operators.

Some research does address semantic heterogeneity directly. Li (Li 1995) identifies similarities between attributes from two schemas using neural networks. He runs a clustering algorithm on the attributes of records from database A, trains a neural net to learn the clusters, then runs records from database B through the network and sees how well it is able to cluster the data. Li does not describe his "similarity determination" algorithm.

Lehmann and Cohn (Lehmann and Cohn 1994) develop an "EGG/YOLK reliability hierarchy" that characterizes compatibility between ontological concept definitions. They assume that concept definitions (eggs) include secondary, more specialized definitions for typical instances (yolks). The eggs and yolks of two concepts can overlap in 42 different ways, which the authors order by their reliability in an exceedingly elaborate and elegant way. The authors do not address how a system can identify the egg/yolk configuration of a pair of concepts that use different vocabulary.

Campbell and Shapiro describe an agent that mediates between agents that subscribe to different ontologies (Campbell and Shapiro 1995). They assume that the ontologies share some concepts. This form of overlap is not an adequate basis for mapping on its own, so their mediating agent engages in sustained dialog with the agent using the destination ontology, in which the mediator questions the user agent about words in the source description. It is not clear, however, whether or how much such dialog can help: “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” (ibid. p. 8).

Bright, Hurson, and Paksad (Bright, Hurson et al. 1994) use a thesaurus to automate generation of a global schema from local database schemas. Then, they estimate similarity between elements of user queries expressed in the global schema to attributes in local schemas, thus supporting a form of imprecise query service. Today, they could use Wordnet (Miller 1990) instead of the common thesaurus. Or, if a formal ontology was available, they could use the measures of description compatibility described in this paper, rather than their “semantic-distance metric” based on path distance, which is an additive version of our LCSR measure.

Kashyap and Sheth (Kashyap and Sheth 1996), like us, seek to describe the similarities and differences between intensional definitions of two concepts (which they call “objects”). Their “semantic proximity” is defined by a tuple, including:

- the context of comparison—which they define most closely as an attribute/value space,
- the objects’ domains—lists of their potential values,
- a correspondences between the domains (called an “abstraction/mapping”), and
- the objects’ states—their current extensions in the respective databases.

This structure then enables a review of the many forms of semantic heterogeneity, and development of a set of rules for applying operations that transform one object to the other. This work, however, is theoretical in nature; it is hard to see how one might quantify their semantic proximity to enable a system to make decisions.

Our matching algorithm is inspired most directly by Artificial-Intelligence research in analogy. In most of this research the things being compared are represented as graphs, and similarities are identified as mappings between elements of the graphs (see Owen (Owen 1990) for a lucid overview). Historically, greedy matching was adapted from Gentner’s “structure mapping” (Gentner 1990). Structure mapping builds internally consistent one-to-one correspondences between expressions in two domains, such that they have the same relations between them. An initial set of alternative mappings is generated by matching relation predicates across two input expressions. The relation predicates must be identical. Then, correspondences between entities are hypothesized by matching the arguments of the relations. The first argument of the relation in the source expression corresponds to the first argument in the target expression, and so on. Next, correspondences are gathered into maximal internally-consistent sets, and evaluated for “systematicity”, a notion involving the size and compactness of the matching. In comparison, greedy matching is more tolerant of heterogeneity than is structure mapping, which depends on identical syntax for relations in the source and target expressions.

8.2 Critique

Ultimately, proof of semantic interoperability must be provided by working systems. Description compatibility measures will predict the probability of satisfactory agent service, or of a useful query response. The accuracy of these measures can then be evaluated by comparison to pragmatic measures: feedback following actual use, including analysis of the frequency of false negatives and false positives (recall and precision, respectively, in Information Retrieval idiom).

The disadvantage of pragmatic evaluation, however, is that it requires commitment to a specific and unavoidably idiosyncratic task domain. Furthermore, the current stage of development of automatic semantic interoperability techniques does not support implementation of systems to solve problems that are important in and of themselves. In comparison, by generating artificial ontologies we can calculate a model-theoretic measure of semantic compatibility that is independent of a pragmatic context.

We consider the artificial ontologies to be the greatest strength of this work—and also the greatest weakness. We believe that we have captured the essential characteristics of differentiated ontologies as they will actually develop, but without actual differentiated ontologies to test, we have no way of demonstrating this. In particular, we have:

- investigated a very small part of the simulation parameter space (specified in Tables 3 and 4).
- restricted the expressiveness of our concepts to a small subset of description logic.

  Recently, tableaux-based description logics (Horrocks 1998) have demonstrated an ability to speedily classify concepts whose definitions include disjunction and negation. While it would be possible to represent these concepts as graphs (MacGregor 1994), it may be difficult to find good matchings for them. Furthermore, although description logic is far more expressive than, say, relational databases, it lacks a satisfactory means of representing uncertainty, which is ubiquitous in real-world applications.
- glossed over additional complexity in the real world.

  One aspect of our simulations that is quite unrealistic is the structure of described-by relations between objects. Our objects are related with a certain probability to objects that are related to each other; but beyond this “one step” level of structure, these relationships are almost random (see Section 5.1). We believe a much greater degree of order will exist in real differentiated ontologies. This order will constitute both an opportunity and a problem for compatibility measures (as we discussed in terms of model accuracy in Section 7.2).

  We have also avoided the issue of primitives. In our simulations there are few primitives and they are all shared, thus fully satisfying our third assumption about differentiated ontologies (see Section 2).

The implications of the primitives issue are not obvious at this time. Ontology specification is not an exercise in philosophy, where the objective is to describe the true nature of the world. Rather, developing ontologies is an engineering process where an organization or community defines consensus knowledge that can be reused to address multiple problems. Philosophically, almost everything is a "natural kind" and requires definition as a primitive. In practice, it is always possible to develop non-primitive definitions for those concepts that are the focus of effort. Weinstein and Alloway (Weinstein and Alloway 1997) includes discussion of the issue of the difficulty of developing formal ontologies.

9 Conclusions

To support communication despite semantic heterogeneity, we use differentiated ontologies. In these structures, concepts are defined in relation to other concepts using logic. Local concepts inherit from shared concepts, and primitives are shared. In our artificial ontologies, concept definitions include roles with numeric and instance fillers, subject to unary and binary constraints. We then explore the nature of these structures: the degree to which we can predict overlap of concept denotations, and the potential usefulness of these predictions to support agent communication.

The results are encouraging. The eight proposed measures of description compatibility have varying levels of association with semantic overlap, consistent with their requirements in terms of input and computation. SHRR measures the proportion of concept roles that are inherited from shared concepts. As such, its correlation to the semantic measure SEMR is a succinct summary of the leverage provided by differentiated ontologies; and that correlation is moderately strong (a ranked correlation of 0.38 in Experiment 15; the relatively unstable unranked correlation is much higher). Matchings are one-to-one correspondences among elements of concept definitions. Matchings enable measures that are somewhat stronger, with ranked correlations approaching 0.5. Most of the improvement for matching-based measures derives from identification of inconsistencies between the source and target definitions. The probabilistic measures require knowledge of the domain and the semantics of relations in concept definitions. With this additional input it is possible to achieve very accurate predictions of semantic overlap: the matching-based probabilistic measure PMHR, for example, enjoys ranked correlations of 0.7 or better. Furthermore, if comparisons are filtered to include only those with three or more roles inherited from shared concepts, there is a moderate increase in the association of all of the measures with SEMR (except for PCIR which is nearly perfect to start with). This confirms our basic intuition that shared inherited definition structure supports estimation of description compatibility.

The degree of discrimination required for specific tasks will determine whether particular measures can be used for that purpose. All of the measures, except perhaps XDIFR, are fully satisfactory for purposes that require less discrimination: for example, as the basis for constructing matchings. Similarly, if agents can try multiple candidates before selecting a particular service, then even the less discriminating foundational measures might be satisfactory. For example, the best available candidate service will be included in the top 15 percent of recommendations by SHRR two thirds of the time. Matching-based measures such as RISOR, however, do a better job of discriminating. Thus, the first candidate recommended by RISOR will be in the top 10 percent of all candidates two thirds of the time, but only 40 percent of the time for SHRR. Again, PMHR is the most accurate of the measures that may be practical to calculate in realistic systems.

We believe this research will be useful to system engineers. Architects will consider differentiated ontologies as an approach to achieving semantic interoperability. Developers will use our results to guide their selection of language to advertise their agents' capabilities. Finally, differentiated ontologies will facilitate ontology reuse by relaxing the commitment required to use an ontology.

Acknowledgments

Ed Durfee, Mike Wellman, and John Lawler provided valuable suggestions for conducting this research. This work was supported in part by the NSF/ARPA/NASA Digital Library Initiative under grant CERA IRI-9411287, and partially by the Congregating Agents project, NSF grant IIS-9872057.

References

Ambite, J. L. and C. A. Knoblock (1995). Reconciling Distributed Information Sources. AAAI Spring Symposium on Information Gathering from Distributed, Heterogenous Environments, Palo Alto, CA.

Bayardo, R. J. J., W. Bohrer, R. Brice, A. Cichocki, J. Fowler, A. Helal, V. Kashyap, T. Ksiezyk, G. Martin, M. Nodine, M. Rashid, M. Rusinkiewicz, R. Shea, C. Unnikrishnan, A. Unruh and D. Woelk (1996). InfoSleuth: Agent-Based Semantic Integration of Information in Open and Dynamic Environments. Austin, TX, MCC.

Borgida, A. and P. F. Patel-Schneider (1994). A Semantics and Complete Algorithm for Subsumption in the CLASSIC Description Language. *Journal of Artificial Intelligence Research* 1: 277-308.

Bright, M. W., A. R. Hurson and S. Pakzad (1994). Automated Resolution of Semantic Heterogeneity in Multidatabases. *ACM Transactions on Database Systems* 19(2): 212-253.

Campbell, A. E. and S. C. Shapiro (1995). Ontologic Mediation: An Overview. IJCAI95 Workshop on Basic Ontological Issues in Knowledge Sharing, Montreal.

Chang, C.-C. K. and H. Garcia-Molina (1998). Conjunctive Constraint Mapping for Data Translation. Third ACM Conference on Digital Libraries, Pittsburgh.

Chawathe, S., H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. Ullman and J. Widom (1994). The TSIMMIS Project: Integration of Heterogenous Information Sources. IPSJ Conference, Tokyo, Japan.

Cohen, W. W. and H. Hirsh (1994). The Learnability of Description Logics with Equality Constraints. *Machine Learning* 17(2/3): 169-199.

Durfee, E. H., T. Mullen, S. Park, J. Vidal and P. Weinstein (1998). The Dynamics of the UMDL Service Market Society. Cooperative Information Agents II, LNAI. M. Klusch and G. WeiB, Springer: 55-78.

Farquhar, A., R. Fikes and J. Rice (1996). The Ontolingua Server: a Tool for Collaborative Ontology Construction. Palo Alto, California, Computer Science Department, Stanford University.

Garey, M. R. and D. S. Johnson (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York, W. H. Freeman and Company.

Gentner, D. (1990). The mechanisms of analogical learning. *Readings in Machine Learning*. J. W. Shavlik and T. G. Dietterich, Morgan Kauffman: 601-622.

Gold, S. and A. Rangarajan (1996). A Graduated Assignment Algorithm for Graph Matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 18(4): 377-388.

Guha, R. V. (1992). Contexts: A formalization and some applications. Ph.D. thesis, Computer Science, Stanford.

Heiler, S. (1995). Semantic Interoperability. *ACM Computing Surveys* 27(2): 271-273.

Horrocks, I. R. (1998). Using an Expressive Description Logic: FaCT or Fiction? Sixth International Conference on the Principles of Knowledge Representation and Reasoning.

Kashyap, V. and A. Sheth (1996). Semantic and Schematic Similarities between Database Objects: A Context-based approach. *International Journal on Very Large Data Bases* 5(4): 276-304.

Kim, W. and J. Seo (1991). Classifying Schematic and Data Heterogeneity in Multidatabase Systems. *IEEE Computer* 24(12).

Knoblock, C. A. and J. L. Ambite (1997). Agents for Information Gathering. *Software Agents*. J. Bradshaw. Menlo Park, CA, AAAI/MIT Press: 1-27.

Kuokka, D. and L. Harada (1995). Supporting Information Retrieval via Matchmaking. Spring Symposium on Information Gathering from Distributed, Heterogenous Environments.

Lawler, E. L. (1976). *Combinatorial Optimization: Networks and Matroids*. New York, Holt, Rinehart and Winston.

Lehmann, F. and A. G. Cohn (1994). The EGG/YOLK Reliability Hierarchy: Semantic Data Integration Using Sorts with Prototypes. Third International ACM Conference on Information and Knowledge Management (CIKM-94), New York, ACM Press.

Li, W.-S. (1995). Knowledge Gathering and Matching in Heterogeneous Databases. AAAI Spring Symposium on Information Gathering from Distributed, Heterogenous Environments.

MacGregor, R. M. (1994). A Description Classifier for the Predicate Calculus. Twelfth National Conference on Artificial Intelligence (AAAI-94).

Mendenhall, W. and R. L. Scheaffer (1973). *Mathematical Statistics with Applications*. North Scituate, MA, Duxbury Press.

Miller, G. A. (1990). WORDNET: An On-Line Lexical Database. *International Journal of Lexicography* 3(4): 235-312.

Milo, T. and S. Zohar (1998). Using Schema Matching to Simplify Heterogeneous Data Translation. Very Large Databases (VLDB).

Nodine, M. H. and A. Unruh (1997). Facilitating Open Communication in Agent Systems: the InfoSleuth Infrastructure. http://www.mcc.com/projects/infosleuth/papers/open_comm.ps

Owen, S. (1990). *Analogy for Authomated Reasoning*. San Diego, California, Academic Press.

Pazzaglia, J.-C. R. and S. M. Embury (1998). Bottom-up Integration of Ontologies in a Database Context. 5th KRDB Workshop, Seattle, WA.

Quillian, M. R. (1968). Semantic Memory. *Semantic Information Processing*. M. L. Minsky. Cambridge, MA, MIT Press: 216-270.

Rangarajan, A. (1996). A Lagrangian Relaxation Network for Graph Matching. *Transactions on Neural Networks* 7(6): 1365-1381.

Visser, P. R. S., D. M. Jones, T. J. M. Bench-Capon and M. J. R. Shave (1997). An Analysis of Ontology Mismatches; Heterogeneity versus Interoperability. AAAI-97 Spring Symposium on Ontological Engineering, Palo Alto, California.

Weinstein, P. and G. Alloway (1997). Seed Ontologies: growing digital libraries as distributed, intelligent systems. Second ACM International Conference on Digital Libraries, Philadelphia, PA, USA.

Weinstein, P. C. and W. P. Birmingham (1998). Creating Ontological Metadata for Digital Library Content and Services. *International Journal on Digital Libraries* 2(1): 19-36.

Woods, W. A. (1991). Understanding Subsumption and Taxonomy: A Framework for Progress. *Principles of Semantic Networks*. J. F. Sowa. San Mateo, California, Morgan Kaufmann Publishers: 45-94.