Runtime Classification of Agent Services

Peter Weinstein and William P. Birmingham

Department of Electrical Engineering and Computer Science
University of Michigan
Ann Arbor, MI 48109-2110
{peterw, wpb}@eecs.umich.edu

Abstract

The Service Classifier Agent maintains a *dynamic ontology* of agent capabilities. To advertise their services, agents define concepts at runtime. These concepts are automatically classified with description logic. Agents requesting services can select the best available to meet their needs, using queries that exploit rich knowledge about services and their relations to other services.

Runtime classification of agent services encourages the development of agents to provide new services. New agents may be utilized immediately upon joining a society, without requiring modification or even notification of existing agents.

1. Introduction

We are interested in societies of agents that are large and evolving. In these distributed systems, agents cooperate to solve problems. Sometimes, new problems demand cooperation in new combinations, or with previously unknown agents. When one agent chooses to work with another, we call that *selecting a service*. Services are the capabilities that agents make available to other agents.

In large, evolving societies accurate selection of services is a difficult problem. There are too many kinds of agents for any agent to know about all of them. New agents appear, and old agents change their behavior. Furthermore, although we do assume knowledge of common language and protocols, we consider terminological heterogeneity—when different terms reference equivalent services, or the same terms reference different services—to be inevitable. Despite adherence to "standards", language use will vary between agent societies, over time as societies change and grow, and even within societies, especially when agents are developed by many parties who may never coordinate their efforts. Without adequate common language, communication—and hence cooperation—is impossible. Thus, the problem of knowledge sharing is at the heart of the problem of building large and evolving societies of agents.

To support cooperation between agents and to facilitate change in agent societies, we are developing the Service Classifier Agent (SCA). This agent maintains an ontology that defines classes of agent services. To advertise, agents define concepts that describe their services at *runtime*. Description logic is used to automatically classify new concepts into a subsumption-based taxonomy. We call this taxonomy a *dynamic ontology*. In comparison, "static" ontologies are either fixed, or are changed slowly over time by committees of persons. Runtime service definitions can use terminology from both static and dynamic ontologies.

Previously, ontologies have been used in agent systems in an essentially static way. Human design teams divide a problem, assign an agent to each part, and negotiate ontology definitions used to describe agent interfaces (the PACT system [Cutkosky 96] is a good example). Henceforth, designers may change definitions, but with increasing difficulty as the system grows and language becomes entrenched in multiple interactions.

Runtime classification of services provides several important benefits. First, descriptions of services provided or required do not reference particular agents, adding flexibility and extensibility to the agent system. Second, the expressive power of ontologies supports selection of agent services with multiple dimensions, at any level of granularity, and from multiple perspectives. Third, we can reason about the relationships between services, to select the best available service or to manage sets of agents providing or seeking services. Finally, there is increased potential for developing ways to exploit ontological descriptions and inference to map between similar concepts, and thus to ameliorate terminological heterogeneity.

This paper describes an important use of ontology: for classifying and selecting services in agent systems designed to change and grow. Section 2 explains the high-level design of the SCA. Section 3 describes an initial implementation, and how the SCA is currently used. Section 4 discusses our plans for future research in the context of knowledge sharing and related fields. Section 5 concludes.

2. Technical approach

The SCA is part of the University of Michigan Digital Library (UMDL) [Birmingham et. al. 94]. Its design is motivated by the fundamental objectives of the UMDL. We envision a very large library, containing just about any kind of information imaginable – from journals to webpages to scientific data to live musical performances. We are implementing the UMDL as a society of agents, and consider this approach to be a research goal as well as a means. Finally, we plan to achieve the desired scale of the UMDL by making participation in the system attractive to third party developers. Third parties should be able to develop new agents to provide content and other services without great difficulty or expense, and should reap the benefits of doing so. Although the SCA is part of the UMDL, it is designed for any society of agents that is complex and evolving.

We use ontologies to represent agent services in the UMDL because the complexity of the digital library domain demands expressiveness well beyond that supported by relational databases. Services can be characterized with many dimensions, and may be viewed at many levels of granularity. Also, a service may be represented simultaneously from multiple perspectives. For example, an agent might seek an agent to recommend a collection of articles on volcanoes, for high-school audiences, to be contracted for via auction.[1] Or, the client might seek an auction that sells a service to recommend collections. Both perspectives are represented in the directed graph of Figure 1 (follow links either from recommend-dlcollection at the top, or auction at the bottom of the figure).

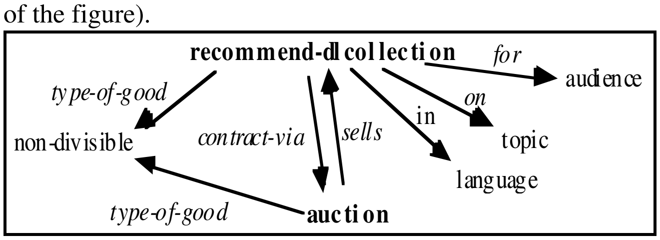

Figure 1: Multi-dimensional, multi-perspective complexity

Description logic is appropriate for maintaining dynamic ontologies because it can do automatic classification. New concepts are placed into their proper location in a concept taxonomy using a subsumption algorithm, based on the structure and content of their definitions. Concepts in the taxonomy are partially ordered from general to specific, and may also be connected by a variety of other relations.

The concept taxonomy can be used in sophisticated ways to reason about services and their relationships to other services. For example, an agent can pose a query to retrieve a list of increasingly general services, ranked according to how specifically they match a multi-dimensional set of constraints (a most-specific-subsuming search strategy). Each candidate service is evaluated, for each constraint, based on the path distance in the concept taxonomy between the request's ideal role filler and the actual filler in the service's concept definition. This type of search is useful when the service must be appropriate for all possible situations within the defined scope. On the other hand, search by increasing specificity is appropriate when the requesting agent doesn't care exactly which service is provided, as long as it is within the defined scope (most-general-subsumed search). For example, if you are searching for an image, you might not care if it was in GIF or JPEG format. In general, the classification taxonomy can be used to search for the best available service using any given quantified definition of "best" (i.e. given a multi-attribute utility function). Alternatively, clients might pose a series of queries—for example, to learn about available options within increasingly specific categories of services.

Classification taxonomies will be especially valuable to agents that need to manage sets of agents. For example, the UMDL includes an "auction manager agent" that is responsible for spawning auctions according to the current supply and demand for services. The current auction manager is very simple, but future versions will need extensive knowledge about services to do their job effectively. Different kinds of auctions are appropriate for different kinds of goods; this information is available in the ontology. Also, it is necessary to maintain desirable numbers and ratios of seller and buyer agents. These quantities will be controlled by adjusting the generality/specificity of the service to be auctioned. This can be accomplished by moving up or down in the description logic concept taxonomy, or by generalizing or further restricting role fillers.

The combination of declarative service descriptions that do not reference particular agents, plus the ability to search for the best available agent, adds agent-level extensibility to the UMDL society. If agents request services to meet their needs with the maximum precision afforded by the ontologies, and if they periodically repeat their search for the best available agent, then if a new agent joins the society that can meet their needs better than existing agents, they will automatically switch to using the new agent. For example, a user-interface agent for a middle school may routinely contract with a database query agent designed for K-12 education. A third party developer might recognize an opportunity, and introduce a new query agent designed specifically for middle schools. The middle-school user-interface agent will then switch to using the new agent. No modification of the user-interface agent is required. Nor is any user intervention or notification required. Thus, as agents evolve to fill increasingly specialized niches, the same client request may be satisfied by increasingly well-matched services. The society infrastructure should also ensure that third parties reap some form of benefit, in accordance with the contribution of their agents make to the society. The result is not only a society of agents that can easily evolve, but a system that actively fosters its own evolution.

To summarize, ontologies' expressive power, and description logic's ability to automatically classify and reason in other ways, make these tools appropriate for the implementation of service classification in large and evolving societies of agents.

[1] Auction agents provide efficient and flexible resource allocation, and are a step towards making third party development for the UMDL economically attractive [Mullen and Wellman 96].

3. An Initial Service Classifier Agent

This section presents an initial implementation of the SCA. We describe our approach to building the underlying static ontologies, the way the SCA is used in the UMDL, and the nature and capabilities of classification and query requests. We also discuss a limitation of currently available description logic systems—the rigid dichotomy between concepts and instances—whose significance for service classification was revealed during the SCA's implementation.

The current SCA provides the basic capabilities of runtime classification and query. It is implemented using the UMDL agent class [Durfee, Kiskis, and Birmingham 96], and the Loom description logic system [MacGregor 93][Brill 93][MacGregor and Brill 92]. An interactive demonstration of the SCA is available on the web at http://www.umich.edu/~peterw/Ontology/sca.html.

3.1 Ontologies

The UMDL ontology currently has three modules. One includes library content and services that we consider to be part of a "generic" digital library. The second module adds concepts specific to the UMDL implementation, such as auctions. To avoid reinventing common concepts such as duration or periodical, we embed the UMDL ontologies in the Frame Ontology and others in the library of ontologies maintained by the Knowledge Systems Lab at Stanford [Farquhar 96]. The third module includes agent services defined at runtime. In Figure 2, the thin line around the SCA represents a "conceptualization boundary", defined by the set of terminology available in its ontologies for use by agents wishing to communicate with the SCA. This line is jagged to show that new concepts are added to the dynamic ontology, as illustrated by the Query Planner Agent's classification request in Figure 2.

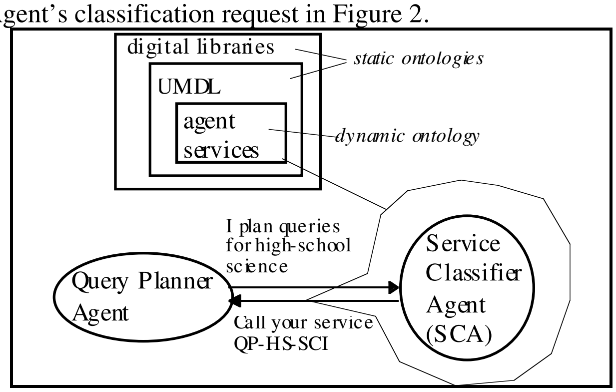

Figure 2: Dynamic ontology nested in static ontologies

To facilitate development of ontology definitions, up-to-date versions are maintained in stylized natural language [Uschold and King 95].[2] Eventually we hope to automate, as much as possible, translations from natural language to KIF, and from KIF to Loom and other representation systems using Ontolingua [Gruber 93]. We now translate manually, directly from English to Loom. We do not attempt to keep the Loom ontologies up-to-date with the natural language versions, which are currently in flux.

3.2 Protocol

The procedure for advertising capabilities in the UMDL involves two agents in the UMDL infrastructure, the SCA and the Registry. The SCA maintains a knowledge base of agent services. The Registry maintains a relational database, which includes the network addresses of individual agent instances. If agent addresses were stored in the SCA knowledge base as instances of the agent service concepts, this would greatly simplify reasoning about those instances (as required, for example, by the auction manager described in Section 2). The Registry, however, is used to maintain instances of agents because of its capabilities for large-scale database management (features related to concurrency, recoverability, query optimization, data distributability, and so on). Figure 3 summarizes the protocol for classification and query. Agent interactions are labeled by their relative order of occurrence.

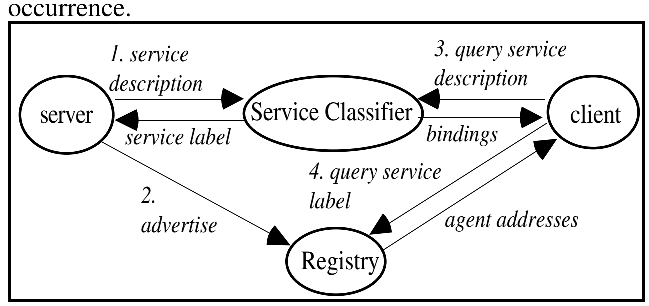

Figure 3: Service classification and query protocol

To advertise, an agent submits a service description to the SCA, and in return receives a service label. The service description is in the syntax of Loom concept expressions, and uses terminology from the static UMDL ontologies, the dynamic agent-services ontology, or both. Agents then use their service labels to advertise in the Registry.

To query for available services a client agent follows a procedure similar to that for classification. The client submits a service description to the SCA using Loom's query expression syntax. It receives a list which includes zero or more labels for services that match the description. The client then uses a label to query the Registry about the availability of agent instances.

If an agent provides multiple services, it classifies each one separately. For example, the same database query agent may have behaviors that make it suitable either for

[2] Our current natural language definitions, along with explanatory diagrams and text, may be viewed at http://www.umich.edu/~peterw/Ontology/ontology.html.

schools, or for businesses. This agent would advertise in the Registry with two relatively specific labels, rather than one general label that encompasses both school and business users. Although Loom does accept disjunctive concept definitions, overuse causes performance to seriously degrade.

It is also possible for several different kinds of agents to provide the same service. For example, a digital-format-translation agent may be able to translate a document from Microsoft Word to ClarisWorks, and a deliver-magazines agent may do the same. If the delivery agent advertises its translation capabilities as a separate service, and the translation agent does not distinguish itself with any attributes available in the ontologies, then the SCA will recommend the same label to both agents. In this case the service-requesting agent receives multiple agent addresses from the Registry, and typically chooses between them in an agent-dependent way.

3.3 Classification

An agent requesting classification submits a service description to the SCA, and also a preferred label. The SCA responds in one of three ways:

- If the service description is classified as a new concept, the preferred label is returned as the recommended service label.
- If the service description is equivalent to a service that has already been classified, then the previously existing label will be returned.
- If the concept is new but the label has already been used for another service description, the SCA generates a new label automatically.

Figure 4 is an example of a description for a service to recommend a collection on the topic science, for middle school audiences. The first line indicates inheritance from the recommend-dlcollection concept, and the second and third lines further restrict the values of its slots. An initial colon (:) identifies a Loom keyword. Other symbols are from the ontologies.

```
(:and  recommend-dlcollection
       (:filled-by recommend-dlcollection.has.audience middle-school)
       (:filled-by recommend-dlcollection.has.topic science))
```

Figure 4: A classification service description

3.4 Queries

Loom's query expression language has full first order expressiveness. Variables may be chained to traverse ontology relationships, either to restrict matches to services with specified role fillers, or to reveal fillers for services that are otherwise selected. If the query is successful, the SCA returns a list of sets of bindings, where each set includes a value for each variable in the query expression. If multiple sets of bindings are returned, the client may analyze the values of secondary variables to determine which service label to use.

The example in Figure 5 asks for a service to recommend a collection, where the service is suitable for an audience that is a kind of school. Symbols that start with a question mark (?) are variables.

```
(:and  (recommend-dlcollection ?service)
       (recommend-dlcollection.has.audience ?service ?audience)
       (school ?audience))
```

Figure 5: A query service description

Loom's query language can be extended with custom predicate functions that accept or reject every combination of bindings considered by the query. In Figure 6 the concept ?service is accepted if the recommend-dlcollection.has.topic slot is filled with the value science, or is not filled at all.

```
(:and  (recommend-dlcollection ?service)
       (:predcall #'same-filler-or-none ?service
           'recommend-dlcollection.has.topic 'science))
```

Figure 6: A query with a custom predicate

Wrappers are special SCA functions that are invoked once for the entire query. Whereas predicate functions execute within Loom queries, wrapper functions typically include a Loom query. Figure 7 shows an example of the most-specific-subsuming wrapper function. The constraints identify a list of candidate sets of bindings. They are implemented as a Loom query. Here, the query selects only recommend-dlcollection services for audiences that are schools, and whose topic is a kind of science. The preferences identify an ideal candidate. Here, candidate services are ranked by inheritance distance between their type of audience and a concept that subsumes middle-school, and their topic and a subsumer of earth-science. The audience preference is prioritized to be twice as important as the topic preference. Also, because (hypothetically) the ontology differentiates topics more densely than audiences, candidate services are penalized less for each inheritance link required for the topic preference, than for the audience preference.

(:wrapper #'most-specific-subsuming	
con-straints	'(:and (recommend-dlcollection ?service) (recommend-dlcollection.has.audience ?service ?audience) (school ?audience) (recommend-dlcollection.has.topic ?service ?topic) (science ?topic))
prefe-rences	'((middle-school ?audience) (earth-science ?topic)) :priorities '(2 1) :attenuation-factors '(0.6 0.8))

Figure 7: A most-specific-subsuming query

3.5 Concepts and instances

In Loom, and other currently available description logic systems, there is a deeply rooted bifurcation between the handling of concepts, which denote sets of objects, and instances, which denote individual objects. Historically, this dichotomy developed partially as part of the philosophy of clarifying links, and partially to enable highly expressive languages for assertions while maintaining computational tractability for classification of concept definitions. Unfortunately, the split between concepts and instances has undesirable consequences for the SCA. See Brachman et. al. [91] for a description of consequences for other systems, and Woods [91] for a design for a more flexible description logic.

For example, classification operates on concepts, but retrieval is over instances. To be able to query service concepts, therefore, the SCA declares each service concept to be an instance of itself. Thus, the concept classified by the example in Figure 4 is A-KIND-OF recommend-dlcollection, and also IS-A recommend-dlcollection.

More seriously, Loom refuses to classify a concept description based on role fillers that are instances. For example, a service description exactly like that in Figure 4, except for biology instead of science, would not be subsumed by the concept in Figure 4, because the :filled-by keyword indicates an instance role filler rather than a concept value restriction.

Currently available description logics view concepts as Platonic ideals. Concepts never change. All dynamic flux in the system pertains to instances only. From this perspective, classification according to instantiated attributes is undesirable—because instances can change, and that would require changing the classification of the concept as well [Brachman et. al. 91 p. 424].

Because the SCA needs to retrieve service concepts based on role fillers, we need to fill our roles with instances. Several approaches can be used to work around the problem. None of them, however, are fully satisfactory.

Our current approach is to establish subsumption outside of Loom: the query wrapper functions described in Section 3.4, for example, operate directly on the subsumption relationships of the individual role values, rather than on the subsumption relationships of the full service concept.

Alternatively, trickier concept definitions could be required. The example in Figure 8 automatically tells the knowledge base about instances (with the :filled-by keyword) that correspond to concept roles (defined with :exactly). The role number restriction is overly constraining, but this is required, in combination with :default inference, to avoid deducing multiple instance fillers (one for each of the concept's subsuming services).

```
(define-concept 'QP-School-Science
  :is '(:and recommend-dlcollection
        (:exactly 1 recommend-dlcollection.has.audience middle-school)
        (:exactly 1 recommend-dlcollection.has.topic biology))
  :default  '(:and
        (:filled-by recommend-dlcollection.has.audience middle-school)
        (:filled-by recommend-dlcollection.has.topic biology)))
```

Figure 8: An integrated concept/instance definition

A third work-around would be to parse the Loom concept definitions, and tell the knowledge base that an instance exists for each concept value restriction.

4. Discussion

This section discusses our plans for future development in the context of research in knowledge sharing and related fields. Our goal is knowledge sharing that supports cooperation in societies of agents that are large, dynamic, evolving, and terminologically heterogenous. The current SCA, however, is not really doing knowledge sharing, because we represent services with a particular representation system rather than an interchange format. Nor does it support large societies, in which, by definition, no agent can know about all of the agents. Nor do we address terminological heterogeneity, since all agents that wish to communicate with the SCA must subscribe to the SCA's ontologies. We intend to rectify these deficiencies by extending service classification to include capabilities for language sharing, translation, and management of multiple SCAs.

4.1 Related research

The methods developed for selecting agent services in societies of agents have corresponded roughly to the problem addressed by the system. For example, in Cooperative Distributed Problem Solving [Gasser and Huhns 89], agents usually know *a priori* about all of the types of agents in the system. Agents contact each other using a name server, or somewhat more generally, by advertising using a compile-time label with some sort of broker agent (such as the UMDL Registry Agent).

Many societies of agents have been developed to find information on the Internet. Sometimes agents are used to represent individual pieces of information, such as "for-sale" and "to-buy" ads. Then, instead of brokers, decentralized clustering of agents based on semantic distance measures can be used to organize inter-agent contacts [Foner 95]. The problem of matching client requests to service descriptions is identical to that faced by agent service brokers, however. Solutions have applied various techniques from information retrieval and artificial intelligence. These may be characterized on a spectrum in which there is a tradeoff between the precision and reliability of the match, and the amount of domain-specific knowledge required. For example, keyword-based matching is universally available, but is neither precise or dependable. Techniques based on word co-occurrence can improve results, but require a scoped corpus of representative documents to provide the statistics. Pattern matching and unification algorithms use local context around keywords to improve matching, and these can be augmented with inference rules that know about the content of expected messages [Kuokka and Harada 95]. On the knowledge-intensive end of the spectrum, context logic [Guha 92] can be used to translate rigorously between requests and service descriptions [Fikes et. al. 95].

One agent that uses extensive knowledge structures to match a client request to available services is the UMDL's Query Planning Agent [Vidal and Durfee 95]. This agent recommends a library collection agent to match the multi-dimensional requirements of a user agent's query. A hierarchy of topics borrowed from the library world provides the ability to broaden or narrow queries, and a thesaurus is used to bridge between the words in the query and the topic hierarchy. The Query Planning Agent is implemented as a rule-driven plan-execution agent. The content language, however, and the method of matching requests to services is essentially *ad hoc* and domain specific.

Progress towards reusable ontologies now makes communication between agents in large and terminologically heterogenous societies a plausible goal. It is not clear, however, if societies of agents where ontologies are used in a static way to define agent interfaces (see Figure 2) are scalable. Consider the PACT system [Cutkosky 96][Olson et. al. 95]. Every agent is associated with a set of KIF ontologies that defines the terminology that may be used to communicate with that agent. The sets of ontologies subscribed to by various agents necessarily overlap, but need not be identical. The ontologies may potentially be reused in different problem contexts, as long as the vocabulary remains appropriate. The problem is that usually vocabulary does not remain appropriate from one context to another. This is partly because of the richness of the interconnected relationships that give words meaning; i.e., frames for important concepts tend to have lots of slots.

The problem is that meaning derives from context, rather than entirely from the word itself. For example, user agents may select among query planning agents on the basis of the characteristics of library content, in library collections, that the planner agents know about. If a book is about volcanoes and written for high-school students, and the book is in a collection, and a query planner knows about that collection, then the query planner can help the user to find that book. Although the attributes of <u>audience</u> and <u>topic</u> are extrinsic to the query planner itself, nevertheless they are needed as slots (necessary conditions) of that concept's frame. It is not difficult, in a knowledge base, to propagate extrinsic attributes using transitive inference rules, starting from the concepts for which they are intrinsic (this capability is built into the representation language used in CYC [Lenat and Guha 90]). What is needed, however, is a way to separate intrinsic and extrinsic attributes such that adequate definitions of concepts can be automatically recreated in different contexts.

For societies of agents to scale, we need to be able to translate between ontologies, so that local systems can develop their own terminology as needed for local problems. Context logic [Guha 92] can be used to translate precisely between terms in different ontologies, but the required mapping axioms are very difficult to develop. We believe that approaches that do "rough mapping" have more promise. All of these methods start with some sort of overlap between the source and target ontologies. This overlap can be in the form of shared "typical" instances [Lehmann and Cohn 94], shared concepts [Campbell and Shapiro 95], or shared parents from which terms in the source and target ontologies inherit [Weinstein 95]. Many of the rough mapping algorithms work by identifying correspondences between subgraphs in the source and target ontologies, similar to certain techniques developed for research in analogy [Gentner 89].

4.2 Future development

In the future, service classifiers will be used to select agent services in increasingly large and terminologically heterogenous societies. This will require the development of capabilities for language sharing and translation. It will also require multiple service classifiers, and a way to manage their organization and interaction.

The SCA should accept descriptions in a knowledge-interchange language such as KIF, not a syntax required for a particular representation system such as Loom. This transition will present some obstacles, but nothing insurmountable. It will be necessary to execute a translation tool (such as Ontolingua) at runtime, without significantly slowing performance. More seriously, it will be necessary to either remove, or avoid, the gap in expressiveness between the interchange language and the representation language. Ontolingua cannot translate all KIF content into any given target language, because no target language can express everything that KIF can.

Progress, however, is being made towards increasing the expressiveness of description logic systems without losing adequate computational tractability. For example, the PC Description Classifier [MacGregor 94] has full first-order expressive power for concepts, extended with sets, cardinality, equality, scalar inequalities, and predicate variables. Hopefully, new description logics will be developed that avoid the unnecessarily strong dichotomy between concepts and instances. A system that learns how to translate concepts between ontologies, for example, would need to be able to modify concepts without destroying individuals defined as instances of those concepts.

We are currently working on two complementary strategies for translating terminology between ontologies. One idea, that we call "reclassification", is appropriate when most terms are shared. Consider a client agent whose ontologies include a concept that expresses a need, but the concept is not part of the classifier's ontologies. If the concept is defined using terms that are in the UMDL ontology, then the client can ask the SCA to classify it. Figure 9 illustrates this situation. The unshared concept in the other agent's ontologies that requires translation is shown with an oval. Related, shared concepts are shown with small rectangles. In case 1, the classifier already has an equivalent definition, whose label it returns, thus translating the unshared concept. In case 2, the classifier does not have an equivalent definition. Instead, it learns a new concept. A third case, not shown in Figure 6, occurs if the classifier does not have an equivalent concept, but the label is already used. In this case the classifier generates a new label. In situations where the concept to be translated has slots with concepts that are not shared, it might be possible to achieve translation by recursive application of reclassification on the slot concepts. Reclassification, however, does not work when there are cycles of unshared concepts.

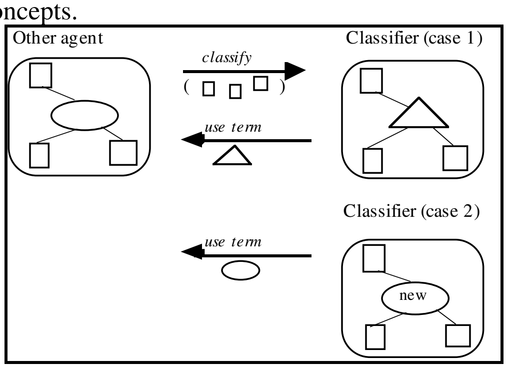

Figure 9: Reclassification to translate a term with shared slot concepts

We also intend to answer requests for services in which most of the terms are not in the SCA's ontologies. We will assume, however, that some of the structure of all concept definitions is shared, by inheritance from terms in ontologies shared by both agents. We will also assume a normalizable syntax for requests, by expecting all service concepts to inherit from a generic concept for service.

To match a request for services, we will build and evaluate alternative *rough mappings* between the source and target concepts [Weinstein 95]. Mappings are sets of one-to-one correspondences between nodes in subgraphs in the source and target concepts. Our mappings are "rough" because correspondences are not exact. The strength of each correspondence is a function of the link proximity between its concepts and their least general subsuming concept in the union of concept taxonomies. The largest and most coherent (densely linked) mappings are evaluated as the best. Currently, the mapping evaluation function is valid for comparing alternative mappings to match two concepts, but not for comparing mappings for matches between different pairs of concepts. To decide whether to recommend a candidate service to the requesting agent requires a technique for comparing rough mappings between different pairs of concepts. We are considering using abduction for this purpose.

Finally, service classification in large societies of agents will require many SCAs. Each SCA will subscribe to some set of static ontologies, and maintain one or more dynamic ontologies. In cases where multiple SCAs are needed to support a single community (defined by a single dynamic ontology of agent services), a single SCA will do all classification, but new concepts will be replicated to other SCAs that handle queries. The quality of translations between agents in different communities will depend on the degree of shared concept structure (the proximity of the concepts' parents in the shared ontologies).

To support society-wide searches for the best available service, ontologies of ontologies will prove useful. Meta-service classifier agents will help agents select service-classification service. The structure and behavior of meta-service classifiers will differ from that of regular service classifiers only in the content of their ontologies.

5. Summary

The SCA maintains a dynamic ontology of agent services. Agents define concepts to advertise their services. Description logic classifies these concepts into a subsumption-based taxonomy. Agents seeking services query the knowledge base to learn what is available.

Runtime classification of agent services has the following advantages. First, service descriptions do not reference particular agents. When new agents are introduced they are immediately utilized on the basis of their service descriptions, without requiring any modification of existing agents. Second, the expressive power of ontologies can represent complexity in domains that are inherently multi-dimensional and multi-perspective. Third, the high degree of organization which description logic imposes on the space of services supports sophisticated queries that utilize rich knowledge about services and the relationships between services. Finally, by providing public access to dynamic repositories of ontological knowledge, service classifiers will facilitate the development of ways to translate between ontologies, either precisely (as in reclassification), or loosely (as with rough mapping). The effectiveness of translation will depend on the requirements of the situation, and the extent of shared structure and terminology in the concept definitions of the communicating agents.

Runtime classification of services gives agents much greater flexibility than does hard-coded selection of services using static ontologies, or dynamic selection of services using less expressive representations. Agent interactions in large and evolving societies of agents may become quite complex. Task planners, auction managers, and other intelligent agents will use knowledge embodied in dynamic ontologies to manage this complexity in a variety of ways.

Acknowledgments

This work was supported by the NSF/ARPA/NASA Digital Library Initiative under grant CERA IRI-9411287.

References

Brachman, R. J.; McGuinness, D. L.; Patel-Scheider, P. F.; Resnick, L. A.; and Borgida, A. 1991. Living with CLASSIC: When and How to Use a KL-ONE-Like Language, pp. 401-56 in Sowa, J. F. (editor), *Principles of Semantic Networks*, Morgan Kaufmann Publishers, San Mateo, California.

Brill, D. 1993. Loom Reference Manual, Version 2.0. University of Southern California. http://www.isi.edu/isd/LOOM/documentation/LOOM-DOCS.html.

Birmingham, W.P.; Drabenstott, K.M.; Frost, C.O.; Warner, A.J.; andWillis, K. 1994. The University of Michigan Digital Library: This is Not Your Father's Library. In Digital Library '94 Proceedings, June 1994, 53-60.

Campbell, A. E., and Shapiro, S. C. 1995. Ontologic Mediation: An Overview, the IJCAI95 Workshop on Basic Ontological Issues in Knowledge Sharing, August 19-20, 1995, Montreal, http://www.csi.uottawa.ca/~doug/

ontolWshop/ontolworkshop.html.

Cutkosky, M. R. 1996. Agent Based Concurrent Design. *Concurrent Engineering* V4 N? p. 439 - 437.

Durfee, E. H.; Kiskis, D. L. ; and Birmingham, W. P. 1996. The Agent Architecture of the University of Michigan Digital Library. To appear in a Special Issue of the *IEE British Computer Society Journal of Software Engineering on Developing Agent Based Systems*.

Farquhar, A.; Fikes, R.; and Rice, J. 1996. The Ontolingua Server: a Tool for Collaborative Ontology Construction. *KAW96*. Also available as KSL-TR-96-26.

Fikes, R.; Engelmore, R.; Farquhar, A.; and Pratt, W. 1995. Network-Based Information Brokers. Presented at the Spring Symposium on Information Gathering from Distributed, Heterogenous Environments. http://www.isi.edu/sims/knoblock/sss95/fikes.ps.

Foner, L. N. 1995. Clustering and Information Sharing in an Ecology of Cooperating Agents, presented at the Spring Symposium on Information Gathering from Distributed, Heterogenous Environments. http://www.isi.edu/sims/knoblock/sss95/foner.ps.

Gasser, L., and Huhns, M. N. (eds). 1989. *Distributed Artificial Intelligence*. Volume II. Pitman, London, Morgan Kauffman, San Mateo, California.

Gentner, D. 1989. The mechanisms of analogical learning, reprinted p. 601-622 in (editors Jude W. Shavlik, Thomas G. Dietterich) *Readings in Machine Learning* (1990), Morgan Kauffman.

Gruber, T. R. 1993. A Translation Approach to Portable Ontology Specifications. http://www-ksl.stanford.edu/yearindex.html.

Guha, R. V. 1992. Contexts: A formalization and some applications. Ph.D. dissertation, Stanford University.

Kuokka, D., and Harada, L. 1995. Supporting Information Retrieval via Matchmaking, presented at the Spring Symposium on Information Gathering from Distributed, Heterogenous Environments. http://www.isi.edu/sims/knoblock/sss95/kuokka.ps.

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

Lenat, D. B., and Guha, R. V. 1990. *Building Large Knowledge-Based Systems*. Reading, MA: Addison-Wesley.

MacGregor, R. M. 1993. Representing Reified Relations in Loom," In *Journal of Experimental and Theoretical Artificial Intelligence*, 5:179-183.

MacGregor, R. M. 1994. A Description Classifier for the Predicate Calculus, in Proceedings of the Twelfth National Conference on Artificial Intelligence, (AAAI 94), pp. 213-220, 1994.

MacGregor, R. M., and Brill, D. 1992. Recognition Algorithms for the Loom Classifier, In Proceedings of the Tenth National Conference on Artificial Intelligence, (AAAI 92), pp. 774-779.

Mullen, T., and Wellman, M. P. 1996. Market-Based Negotiation for Digital Library Services. Second USENIX Workshop on Electronic Commerce, November 1996. http://ai.eecs.umich.edu/people/mullen/papers/usenix/.

Olsen, G. R.; Cutkosky, M. R.; Tenenbaum, J. M.; and Gruber, T. R. 1995. Collaborative Engineering Based on Knowledge Sharing Agreements. *Concurrent Engineering* V3 N2 p. 145 - 159.

Uschold, M., and King, M. 1995. Towards a Methodology for Building Ontologies, Artificial Intelligence Applications Institute technical report AIAI-TR-183, July 1995.

Vidal, J. M., and Durfee, E. H. 1995. Task Planning Agents in the UMDL. Proceedings of the 1995 Intelligent Information Agents Workshop. http://ai.eecs.umich.edu/people/jmvidal/papers/tpa/tpa.html

Weinstein, P. 1995. Matching Expressions with Differentiated Vocabulary: Technology for Knowledge Sharing in Partially Focused Communities. http://www.umich.edu/~peterw/abstract.vocab.html.

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

Woods, W. A., Schmolze, J. G. 1992. The KL-ONE family, *Computers and Mathematics with Applications*, 23:133-177. Reprinted in Lehmann, F. (ed.), *Semantic Networks in Artificial Intelligence*, Pergamon Press.