Large-Scale Repositories of Highly Expressive Reusable Knowledge

Richard Fikes
Adam Farquhar

Knowledge Systems Laboratory
Computer Science Department
Stanford University

March 1997

Abstract

We describe an ongoing project to develop technology that will support collaborative construction and effective use of distributed large-scale repositories of highly expressive reusable ontologies. We are focusing on developing a distributed server architecture for ontology construction and use, representation formalisms that remove key barriers to expressing essential knowledge in and about ontologies, ontology construction tools, and tools for obtaining domain models for use in applications from large-scale ontology repositories. We are building on the results of the DARPA Knowledge Sharing Effort, specifically by using the Knowledge Interchange Format (KIF) as a core representation language and the Ontolingua system as a core ontology development environment.

In order to enable distributed ontology repositories and services, we are developing a distributed server architecture for ontology construction and use based on ontology servers which provide access via a network API to the contents of ontologies and to information derivable from the contents by a general purpose reasoner. Ontology servers will be analogous to data base servers and will provide services including configuration management, support for distributed ontologies with components resident on remote servers, and automatic caching of derived results.

We are developing new representation formalisms, integrating existing formalisms, and incorporating the results into the tools and servers developed in the project. The representation language resulting from this work will enable ontologies to contain richly textured descriptions that are structured into multiple views and abstractions, and are expressed in a generic representation formalism optimized for reuse. In addition, a computer interpretable ontology description language will enable annotation of ontologies with assumptions made, approximations made, topics covered, example uses, competency, relationships to other ontologies, etc.

We are addressing key difficulties in building large scale ontologies by developing ontology construction tools for specifying the overall structure of an ontology during the early stages of development, supporting teams of collaborating developers, testing and debugging ontologies, and merging ontologies.

We are also developing retrieval, extraction, composition, and translation tools that will enable users to effectively obtain domain models from large-scale ontology repositories that satisfy a set of application-specific requirements regarding content, level of abstraction, view, underlying assumptions, representation language, usability by problem solving methods, etc.

Overview

We are developing technology that will enable construction and effective usage of distributed large-scale repositories of highly expressive reusable ontologies. We consider ontologies to be domain theories that specify a domain-specific vocabulary of entities, classes, properties, predicates, and functions, and a set of relationships that necessarily hold among those vocabulary items (Fikes 1996). Ontologies provide a vocabulary for representing knowledge about a domain and for describing specific situations in a domain. They can be used as building block components of knowledge bases, object schema for object-oriented systems, conceptual schema for data bases, structured glossaries for human collaborations, vocabularies for communication between agents, class definitions for conventional software system, etc.

Ontology construction is a complex collaborative process that crosses individual, organizational, and geographic boundaries. It involves several types of groups with differing expertise, goals, and interactions. An ontology server must be carefully structured to support this complexity. Consider, for example, the task of building schema to support the command and control (C2) process. A common C2 schema would provide a substrate for numerous applications in planning, logistics, intelligence, etc.. With the proper underlying technology, it could support advanced knowledge-based applications as well as conventional data base and software systems. To construct this schema, small groups of experts in each of the key sub-areas collaborate to specify ontologies describing the essential concepts, their properties, and interrelationships. The products of these groups of authors must be merged and checked for consistency by a supervisory board of editors. The editors must then invite comments from a large group of reviewers and critics that include expert peers, end users, and application developers. As portions of the ontologies stabilize and the editors release them from the reviewing process, larger groups of application developers must become familiar with them and incorporate them into existing and new applications. Furthermore, the developers need support to convert the ontologies into a form that they can readily work with in a specific knowledge representation language, database schema language, interface language, or programming language, and they need support for extracting domain models from the ontologies that can be used by problem solving modules.

We are developing technology that will make this scenario a reality by addressing key barriers to representing essential knowledge in and about ontologies, constructing ontologies, accessing the content of ontologies, and obtaining domain models from large-scale ontology repositories. We are building on the results of the Defense Advanced Research Projects Agency (DARPA) Knowledge Sharing Effort (Patil et al. 1992), specifically by using the Knowledge Interchange Format (KIF) (Genesereth and Fikes 1991; Genesereth and Fikes 1992) as a core representation language and the Ontolingua system (Farquhar, Fikes, and Rice 1996; Gruber 1993) as a core ontology development environment. KIF is a highly expressive formally defined language specifically designed for representing knowledge in a reusable and translatable form. Ontolingua is a system in wide-spread use that adds object-oriented constructs to KIF, includes a library of reusable ontologies, supports collaborative ontology construction by assembling and extending building block ontologies from the library, provides an HTML-based user interface (Rice et al. 1996) accessible via a World Wide Web navigator (e.g., Netscape), and supports a network GFP API (Karp, Myers, and Gruber 1995) for object-oriented access and editing of ontologies.

Specifically, we are:

- Developing a distributed server architecture for ontology construction and use;
- Developing new representation formalisms, integrating existing formalisms, and incorporating the results into the tools and servers developed in the project;

- Developing tools that address key difficulties in building large scale ontologies; and
- Developing retrieval, extraction, composition, and translation tools for obtaining domain models from large-scale ontology repositories that satisfy a set of application-specific requirements.

Distributed Server Architecture – We are developing a distributed server architecture for ontology construction and use based on **ontology servers** which provide access to the contents of ontologies via a network API and to information derived from the contents by a general purpose reasoner. Ontology servers will be analogous to data base servers and will enable distributed ontology repositories and distributed servers for editing, browsing, etc. which access the repositories. Ontology servers will provide a suite of services, including configuration management for ontologies, support for ontologies that have components resident on remote servers, and support for an *Ontology-URL* that enables ontologies to be linked into World Wide Web pages so that they are accessible for browsing like any other Web "document".

A particularly difficult task for an ontology server is supporting efficient query answering from ontologies represented in a highly expressive language. To provide that support, we are developing an **idiom-based retrieval facility** that returns instances of a sentence containing schema variables from a given ontology. The retrieval facility will employ a general purpose reasoner that can be run as a background process to infer and cache sentences that match idioms used by the API and by translators. These derived facts will be removed by a truth maintenance facility when the statements on which they are based are removed.

Representing Essential Knowledge In And About Ontologies – There are significant gaps in the expressive power of current knowledge representation languages. These gaps prevent the inclusion in ontologies of knowledge about domains that is essential for many high-priority applications and knowledge about ontologies themselves that is essential for effective ontology use and reuse. We will close some of the more important of those gaps by developing new representation formalisms, integrating existing formalisms, and incorporating the results into the tools and servers developed in the project. The results will enable ontologies to contain richly textured descriptions that are structured into multiple views and abstractions, and are expressed in a generic representation formalism optimized for reuse. In addition, a computer interpretable ontology description language will enable annotation of ontologies with assumptions made, approximations made, topics covered, example uses, competency, relationships to other ontologies, etc.

We are developing a representation for **ontology competency** based on evaluable functions and relations analogous to methods in object-oriented programming. We are developing tools for composing and associating with an ontology methods that are procedural implementations of theorems provable in the ontology. The methods would then be directly callable by applications that use the ontology.

Ontology Construction – We are addressing key difficulties in building large scale ontologies by developing tools for specifying the overall structure of an ontology during the early stages of development, supporting teams of collaborating developers, testing and debugging ontologies, and merging ontologies.

We are developing tools for **testing ontologies** that enable a developer to use an ontology to describe familiar situations and to query those situations to determine if the situations as described have expected properties. The query answering facility will use the ontology server's general purpose reasoner to derive answers.

We are also developing tools for **merging ontologies** that describe a common sub-domain using differing vocabularies, assumptions, approximations, views, abstractions, etc. The tools will:

- Use the ontology server's reasoner to derive and add to the merged ontology equivalence, subsumption, and disjointness relationships among the classes, predicates, and functions of the ontologies being merged;
- Provide facilities for renaming classes, predicates, and functions in the merged ontology; and
- Provide facilities for combining classes, predicates, and functions that have differing definitions but are intended to be equivalent.

Obtaining Domain Models From Large-Scale Ontology Repositories – Sophisticated retrieval, extraction, composition, and translation tools will be needed in order to effectively obtain domain models from large-scale ontology repositories that satisfy a set of application-specific requirements regarding content, level of abstraction, view, underlying assumptions, representation language, usability by problem solving methods, etc.

Given that a set of classes, objects, relations, functions, views, and/or topics have been identified for retrieval, **relevance-based extraction and composition** techniques are needed for producing an ontology which contains all the sections of repository ontologies that are relevant to the identified elements. We are developing such techniques by extending our research on irrelevance reasoning in knowledge based systems and compositional modeling in engineering domains.

For ontologies from a repository to be incorporated into an application system, the knowledge must be translatable in some practical way into the receiving system's representation language. Currently, knowledge base translators are difficult to build, maintain, and extend because they must be hand coded by experts and translation rules are typically embedded procedurally in the program. We are addressing these problems by developing **vocabulary translation tools** that enable a knowledge base builder to specify and apply declarative translation rules, a **suite of translators** for the extended representation language we will develop, and a declarative translation rule language that enables customization of the translators for particular uses.

Representing Essential Knowledge In and About Ontologies

There are many gaps in the expressive power of current knowledge representation languages that prevent or make impractically difficult the inclusion *in* ontologies of knowledge about domains that is essential for many high priority applications and knowledge *about* ontologies themselves that is essential for their effective use and reuse. We are closing some of the more important of those gaps by developing new representation formalisms, integrating existing representation formalisms, and incorporating the results into the Ontolingua representation language and into all of the tools and servers developed in the project.

Multiple Models

Entities in a given domain can be usefully described from multiple perspectives (views), at multiple levels of abstraction, and under alternative sets of assumptions. We are adding a "multiple modeling capability" to the Ontolingua representation language to support:

- **Views**, where each view corresponds to a class with its own set of descriptive properties and attributes. Each view class could be linked to classes that can be used as other views of a common object and each instance of a view class could be linked to the other instance descriptions of the same object.
- **Abstractions**, where each abstraction corresponds to a class with its own set of descriptive properties and attributes. Classes could have class slots (e.g.,

"Abstracts" and "Abstraction-Of") whose values define an abstraction hierarchy. We would also develop a formalism for describing the relationships between levels of abstractions to support reasoning across abstraction levels and propagation of changes through abstraction levels.

- **Sets of incompatible alternative descriptions** (e.g., with different simplifying assumptions) to assure that at most one alternative from each group is included in an ontology.

Contexts

The notion of context is central for representing many aspects of knowledge. Contexts in knowledge representation languages provide a means of referring to a group of related assertions (closed under entailment) about which something can be said. For example, contexts can be used to represent the beliefs of an individual (e.g., a ship's captain) or group (i.e., a ship's crew), the information accessible to a sensor or to an agent, descriptions of situations, alternative descriptions of domain objects, or the ontology being used by an agent for a specific activity. Formal theories of context are sufficiently well developed so that constructs for representing contexts are a suitable candidate for inclusion in the Ontolingua representation language.

We are extending the Ontolingua representation language to include contexts as formalized by McCarthy and Buvac (Buvac, Buvac, and Mason 1995; McCarthy and Buvac 1994). The extension will enable ontologies to include statements made with respect to a context, statements relating what is true in one context to what is true in other contexts, and statements that describe and relate contexts.

Ontology Competency Expressed as Procedural Methods

We are developing techniques and tools for characterizing the computations that are supported by an ontology. In particular, we will develop facilities that support a notion of ontology competency based on evaluable functions and relations analogous to methods in object-oriented programming. The facility would provide a capability of associating with an ontology a set of methods that are procedural implementations of theorems provable in the ontology. Such methods would be directly callable by applications that use the ontology.

If a constructive proof can be given for a sentence of the form $\Phi(x1,\ldots,xn) \Rightarrow R(x1,\ldots,xn)$, where R is a relation and $\Phi(x1,\ldots,xn)$ is a sentence containing only computable functions and relations, then R becomes a computable relation. We will develop a compiler that produces a method that computes $R(x1,\ldots,xn)$ from such a constructive proof. The resulting method would serve as a reference implementation of the method in some widely available implementation language (e.g., JAVA or C). Such methods associated with an ontology could be considered to be extensions of the API used to access the ontology.

Ontology Annotation

As with software in general, the reusability of ontologies depends critically on the availability of information describing each ontology. Useful information to an agent considering an ontology includes the assumptions made, approximations made, topics covered, example descriptions using the ontology, competency descriptions, relationships to other ontologies, etc. The availability of such statements is critical for many aspects of reuse. For example, when an ontology is being constructed using existing component ontologies, they can be used to ensure that the assumptions associated with the set of component ontologies do not conflict.

We are developing a computer interpretable annotation language for use in structuring ontology repositories that will include relations among ontologies such as "abstracts", "approximates", and "alternative". The annotation language may need to include a domain specific vocabulary so that provision will need to be made for an ontology to have associated with it an annotation ontology for that domain. Annotations may also provide examples of the competency of an ontology by describing example situations using the ontology and sets of queries about those situations whose answers are derivable from the ontology.

Ontology Construction

Key difficulties in building large scale ontologies include developing the overall structure of the ontology during the early stages of development, supporting teams of collaborating developers, testing and debugging, integrating alternative versions, and acquiring the numeric parameters of probabilistic models. We are developing a suite of ontology construction tools that address these difficulties as described in the following sections.

Browsers and Editors

We are extending the current Ontolingua system so that it supports browsing and editing large ontologies. We will employ client-side applets (e.g., in JAVA or VRML) to provide tightly coupled, fast response interactions. 2-D and 3-D visualization and structure editing metaphors will help users to get an overview of ontologies and manipulate them in novel ways.

We will also provide support for a broader range of the ontology development cycle, including the early phases which often require compound edits such as taxonomy restructuring as well as adding descriptions of new classes. An example of an editor for early stages of ontology development is one that mimics Microsoft Word's outline mode and is used to specify basic frame language constructs like class-subclass relationships, slots in classes, and value type restrictions. The editor could include a spelling corrector that uses the ontology's vocabulary as its dictionary. We would consider adapting existing outline editors (e.g., in the form of JAVA applets) to produce such an editing tool. A simple frame structure input language for Ontolingua would enable developers to use their own text editors to produce a document loadable by Ontolingua describing class-subclass relationships, slots in classes, and value type restrictions.

Collaborative Construction

As we have outlined, ontology construction is an inherently collaborative process. We are extending Ontolingua’s current facilities for collaborative construction to support large numbers of interacting groups and to provide more complete support for the editing and reviewing process. For example, reviewers will have commands available to annotate and suggest changes to an existing ontology, without being able to actually modify it. Using COTS tools, comments and suggestions can be integrated with threaded e-mail discussions so that reviewers can easily determine whether or not their suggestions have been followed up on by developers.

Ontology Testing

As with any software development, testing and debugging are major issues in ontology development. We are developing tools for testing ontologies, including:

- Tools that enable a developer to use an ontology to describe familiar situations and to query those situations to determine if the situations as described have expected properties. The ontology server's general purpose reasoner will be used to derive answers to queries, and the algorithm for extracting Bayesian network

models for specific situations from a knowledge base will be used for testing the uncertainty information in an ontology.

- A facility for specifying a test suite for an ontology, where each test consists of a situation specification, a set of queries about the situation, and the expected answers to the queries. Such test suites can be used as requirements specifications for ontologies, for regression testing of ontologies, and as examples of the computations supported by ontologies.

Ontology Integration

We are developing tools for integrating ontologies that describe a common sub-domain using differing vocabularies, assumptions, approximations, views, abstractions, etc. The result of the integration may be:

- A single ontology that is a merger of the component ontologies, or
- A set of rules for translating sentences in one ontology into sentences in another ontology.

Integrating the vocabularies of two ontologies involves determining and specifying relationships between the intended meanings of the non-logical symbols (i.e., symbols that name the classes, predicates, and functions) of the first ontology and the non-logical symbols of the second ontology. The ontology server's reasoner can assist with that task by deriving equivalence, subsumption, and disjointness relationships among the classes, predicates, and functions of the ontologies. When the integration is to produce a merger of the two ontologies, these derivations can be added to the resulting ontology. Facilities are then needed to enable the user to rename classes, predicates, and functions in the resulting ontology, and to combine classes, predicates, and functions that have differing definitions but are intended to be equivalent. When the integration is to produce a set of translation rules, then the derivations can be represented as rules, and facilities are needed to enable the user to add rules that rename and combine symbols.

Accessing the Contents of Ontologies

Effective ontology construction and use are many faceted activities that require a wide variety of tools and services. Furthermore, organizations and individuals may have strong ownership interests in the ontologies or components of ontologies that they develop. In order to support ontology tools and services and to respect ownership interests, successful distribution of and access to ontologies requires a distributed server architecture.

We are designing and building a prototype *ontology server* which provides access to the contents of ontologies via a network API and to information derived from the contents by a general purpose reasoner. Ontology servers will be analogous to data base servers and will support distributed repositories and distributed servers for editing, browsing, etc. which access the repositories.

The ontology server will support distributed ontologies by enabling component (i.e., included) ontologies to be resident on remote servers. This capability will be achieved by implementing various forms of remote references within an ontology for classes, relations, functions, and axiom schema loaded on remote servers.

The ontology server will also provide the capabilities described in the following sections.

Configuration Management

Configuration management facilities for ontologies and their components are critical for supporting the development and usage life cycle of ontologies. We are developing configuration management facilities for ontology servers that are specific to ontologies

and that augment standard software system configuration management facilities, including the following:

- Version tracking of component ontologies so that a given version of an ontology always contains the same components;
- Author notification when there is a more recent version of a component ontology available;
- Commands for upgrading a component ontology to its later version;
- Facilities for specifying rules for translating descriptions in one version to descriptions in another version;
- Facilities for performing regression testing on new versions;
- Facilities for succinctly describing the differences between versions of an ontology; and
- Facilities to notify running applications and developers as releases are made.

Inference and Idiom-Based Retrieval

The ontology server must support queries against a rich representation language, answer them efficiently, and balance resources across the server and clients. To support efficient query answering and ontology translation (see below), we will employ an *idiom-based retrieval facility* that returns all instances of a given sentence containing schema variables (i.e., an idiom) that are in a given ontology.

The idiom-based retrieval facility will employ a general purpose reasoner (i.e., theorem prover) and classifier that can be run as a background process to infer sentences that match certain schematic patterns. These derived facts will be cached using a truth maintenance facility so that they can be removed in case the statements on which they are based are deleted.

We are developing a general purpose reasoner (i.e., theorem prover) for the Ontolingua representation language to provide basic reasoning support for ontology services, including:

- Classification;
- Deriving and caching instances of sentence schema to support idiom-based ontology access;
- Ontology testing in which an ontology is used to describe a familiar situation and the description is queried to determine if it has expected properties;
- Client-side execution. In order to make effective use of the server's resources, some inference will need to be done on client machines. A client-side inference tool that uses the network API to access an ontology's contents will be able to perform analysis, consistency checking, and inference without placing undo strain on the server's performance.

We will include a truth maintenance system in the ontology server that will enable derived sentences (i.e., theorems) to be cached in ontologies accompanied by the axioms that support their derivation, and will automatically delete derived sentences when their supporting axioms are deleted. This facility will enable servers to augment ontologies with derived sentences that support specific server capabilities, such as sentences that match axiom schema recognizable by translators or API's and sentences that define evaluable functions and relations.

Network API for Ontologies

Ontology servers will provide access to ontologies via a network API. We will specify a CORBA-based (Mowbray and Zahavi 1995) network API for ontology servers that includes the full Ontolingua representation language. We expect the Ontolingua API to

be an extension of the Generic Frame Protocol (GFP) (Karp, Myers, and Gruber 1995) , and we will work with the GFP Working Group in the DARPA Knowledge Sharing Effort to maximize the correspondence between these protocols and assure consistency in the areas where they overlap.

The API will use the ontology server's idiom-based retrieval mechanism as its basic mechanism for obtaining information from an ontology. For example, to obtain the subclasses of a given class C, the API will retrieve true instances of the sentence "(Subclass-Of ?x C)". The API will augment the server's idiom-based retrieval mechanism by tasking the server's background derivation facility to derive and add to each ontology true instances of the idioms that are used by the API's retrieval functions. For example, the derivation facility could derive the sentence "(Subclass-Of Csub C)" from the sentences:

(=> (Csub ?x) (C ?x))
(<=> (subclass-Of ?c1 ?c2) (=> (holds ?c1 ?x) (holds ?c2 ?x)))

The API's response to a retrieval request will then include the results derived during the background processing.

World Wide Knowledge Web

We are developing technology to enable ontologies to be linked into World Wide Web pages so that they are accessible for browsing via the Web like any other Web "document". Each ontology would be resident on a ontology server, but would be accessible for browsing from any Web browser. The basic enabler would be an "Ontology-URL" that specifies an ontology, a ontology server on which the ontology resides, and an ontology browser, so that clicking on the Ontology-URL would invoke the browser presenting a view of the ontology.

Obtaining Domain Models From Large-Scale Ontology Repositories

We anticipate that ontology repositories will contain richly textured descriptions that include uncertainty, are structured into multiple views and abstractions, and are expressed in a generic representation formalism optimized for reuse. Sophisticated retrieval, extraction, composition, and translation tools will be needed to effectively obtain domain models from large-scale ontology repositories that satisfy application-specific requirements regarding content, abstraction level, view, underlying assumptions, representation language, usability by problem solving methods, etc. We are developing a suite of such tools as described in the following sections.

Topic-Based and Class-Based Retrieval

We are developing facilities for accessing ontology repositories by browsing and querying a topic index to identify ontologies of interest in a repository (i.e., *topic-based* retrieval). In addition, we are developing facilities for accessing ontology repositories by browsing and querying the class-subclass taxonomy of a reference ontology to identify classes of interest in a repository (i.e., *class-based* retrieval). The class-based retrieval facility will automatically determine additional classes that are needed from a repository to support the retrieved classes.

A consensus standard "upper level" reference taxonomy could serve as an index into an ontology repository for class-based retrieval by asking ontology developers to specify for each class in their ontologies which if any immediate superclasses that class has in the reference ontology. (This is a generalization of the current notion of specifying "Thing" as the superclass of top level classes in an ontology.) The result would be that one could browse the reference ontology looking for classes of interest in the repository. The

repository would look like one big ontology with all the repository classes reachable from the reference ontology. One could also search the reference repository to find a class of interest and then browse from the retrieved class as before.

We are working as members of the ANSI committee on ontology standards to develop and make generally available a consensus standard reference ontology. The goal of that committee is to build a large, reusable, widely-held, representation-neutral ontology to serve as a central terminology base and inter-model fulcrum. We intend to focus our work in that committee on extending the reference ontology to include a reference set of slot names and augmenting the reference classes with slots, slot constraints, and disjointness constraints so that each class has a distinct semantics from other classes in the reference set.

In addition to the top down index provided by a reference ontology, a retrieval facility could provide suggestions for alternatives to a given class based on abstraction and view links associated with the class.

Relevance-Based Ontology Composition

Given that a set of classes, objects, relations, functions, views, and/or topics have been identified for retrieval, techniques are needed for extracting and composing an ontology that contains all the sections of repository ontologies that are relevant to the identified elements and is logically consistent. For example, such techniques would support selecting the "relevant" portion of the reference ontology to include in an ontology that is being developed. We are developing a set of relevance-based extraction and composition techniques as extensions to the research we have done on irrelevance reasoning in knowledge based systems (Levy 1993) and compositional modeling in engineering domains (Falkenhainer et al. 1994; Low and Iwasaki 1992).

Translation

For ontologies from an ontology repository to be incorporated into an application system, the knowledge must either be represented in the receiving system's representation language or be translatable in some practical way into that language (Baalen and Fikes 1993; Buvac and Fikes 1995; Gruber 1993). We cannot expect a standard knowledge representation language to emerge that would be used generally in application systems, and we cannot expect all application systems to use the same domain-specific vocabulary in their knowledge bases. Thus, a critical enabler for widespread use of ontology repositories is an ability to effectively translate knowledge from the language(s) in which it is represented in ontology repositories into specialized representation languages. Currently, knowledge base translators must be hand coded by experts, and translation rules are typically embedded procedurally in the program. Translators are therefore difficult to build, maintain, and extend. More effective tools are needed for specifying and performing translation of repository ontologies both among knowledge representation formalisms (e.g., between KIF and IDL) and among domain-specific vocabularies.

We are developing vocabulary translation tools that:

- Enable a knowledge base builder to declaratively specify translation rules between the vocabulary used in a source ontology and the vocabulary used in a target ontology, and
- Apply declarative translation rules to translate a source ontology into an ontology that uses the vocabulary of a target ontology.

We are developing a suite of Ontolingua translators for standard schema languages (e.g., IDL), data base languages (e.g., SQL), object-oriented programming languages (e.g., C++), and knowledge representation languages (e.g., LOOM). These translators will:

- Support the additional expressive power being added to Ontolingua, and

- Include a declarative translation rule language that enables knowledge base builders to extend and customize the translators for particular uses.

Translating from Ontolingua *into* a given representation language is in general a difficult problem, because in most cases the expressive power of the target language is less than that of Ontolingua. Hence, there will not be a translation in the target language of every possible Ontolingua sentence. Instead, some subset of Ontolingua will be translatable into the target language. For example, the only form of quantified statement representable in an object schema language might be:

(forall <variable>$_1$) (=> (member-Of <variable>$_1$ <class>) <quantifier_free_sentence>)

All quantified Ontolingua statements that are not in the above form would not be translatable into the object schema language.

In order to translate Ontolingua into a given target language, one needs to specify the subset of Ontolingua sentences that are translatable into the language and the translation of each such sentence. Given such a specification, the task of a translator, then, is to determine for each sentence in the ontology to be translated whether it is in the translatable subset or *is equivalent to a sentence in the translatable subset.* The test of whether a given sentence is in the translatable subset can be performed by describing the subset as a grammar and applying standard pattern matching techniques for "recognizing" the grammar (Baalen and Fikes 1993; Buvac and Fikes 1995). Thus, for example, we might specify that the following form of quantified statement is representable in a simple target frame language:

(forall <variable>$_1$) (=> (member-Of <variable>$_1$ <class>) {TRANS <sentence>})

where "{TRANS <sentence>}" matches against any sentence that is itself translatable.

The difficulty in the translation arises in determining whether a given sentence is *equivalent* to a translatable sentence. As simple examples, consider the following sentence forms that are equivalent to the example above:

(forall <variable>$_1$) (=> (class-Of <class> <variable>$_1$)) {TRANS <sentence>})
(forall <variable>$_1$) (=> (<unary relation> <variable>$_1$) {TRANS <sentence>})

In the first form, the "Member-Of" relation has been replaced by its inverse "Class-Of", and in the second, the class is considered to be a unary relation rather than a set. In general, a translator can be considered to have available a set of axioms that can be used to *reformulate* sentences that are not recognizable by the translation grammar by deriving sentences that are recognizable and therefore translatable by the grammar. For example, sentences of the forms given above could be transformed into translatable sentences by using the following axioms:

(<=> (member-Of ?x ?c) (class-Of ?c ?x))
(<=> (member-Of ?x ?c) (holds ?c ?x))

Translation servers can prepare for requests to translate a given ontology by tasking the ontology server on which the ontology resides with deriving in the background and adding to the ontology translatable reformulations of sentences in the ontology that would otherwise not be translatable.

Comparison With Other Research

Current and recent work on the Ontolingua ontology development environment at the Knowledge Systems Laboratory (KSL) is unique in its translation approach to knowledge sharing, its implemented architecture for distributed collaborative ontology construction, its large user community, and its substantial library of reusable ontologies. Nonetheless, there are a number of efforts and research projects that are relevant to the proposed work. These fall into four categories:

- Large knowledge base efforts such as CYCorp's CYC project (Guha et al. 1990; Lenat 1995), the Botany Knowledge Base (BKB) project at the University of Texas (Clark and Porter 1996), and the SRI ECOCYC project (Karp et al. 1996);
- Taxonomic ontology efforts, including those aimed at supporting machine translation such as Pangloss (Hovy and Nirenburg 1992; Knight and Luk 1994) and WordNet (Miller 1995), and those aimed at providing uniform terminology in the medical domain such as UMLS (Lindberg, Humphreys, and McCray 1993);
- Interoperable knowledge base tools and protocols such as those developed in the DARPA Knowledge Sharing Effort (Patil et al. 1992), the Generic Frame Protocol (GFP) (Karp, Myers, and Gruber 1995), and SRI's Generic Knowledge Base (GKB) editor; and
- Knowledge representation systems such as Classic (Borgida et al. 1989), Loom (MacGregor 1991), and Theo (Mitchell et al. 1989).

Large Knowledge Base Efforts

There have unfortunately been few efforts to construct large knowledge bases with both broad coverage and substantial depth in specific domains.

Perhaps the best known of these efforts is the CYC project (Guha et al. 1990; Lenat 1995). CYC's ambitious goal was to encode a substantial portion of human knowledge over a ten year period. During this time it has represented a large number of concepts and relations among them. Since its contents have been kept proprietary[1], the quality and coverage of the representations are largely unknown. Also, the CYC knowledge base is basically monolithic, making use in application systems a daunting proposition. Although CYC has developed some sophisticated editing and browsing techniques, it has not provided distributed access for editing or support for collaboration, relying on frequent person-to-person meetings and tight managerial control to avoid conflicts.

The Botany Knowledge Base (BKB) project at the University of Texas aims at encoding the knowledge present in a college level botany textbook (Clark and Porter 1996). It has made substantial progress in this direction and has been used to support applications in tutoring, natural language generation, and qualitative simulation. Work coming out of the BKB project has taken important steps towards supporting multiple views (Ackers and Porter 1994) and extracting relevant portions of a knowledge base for solving specific problems (Rickel and Porter 1994).

The ECOCYC knowledge base project at SRI (Karp et al. 1996) aims at encoding a substantial amount of knowledge about e coli bacteria, including there genetic structure and metabolic pathways. In order to be useful for domain experts, it provides sophisticated display methods that present the knowledge in schematic drawings that experts are familiar with.

Taxonomic Ontology Efforts

There have been an increasing number of efforts to build large taxonomic ontologies. These taxonomic ontologies are being used in several different areas including natural language processing and medical informatics.

Hovy at ISI has employed a substantial ontology for the purpose of machine translation (Hovy and Nirenburg 1992; Knight and Luk 1994). This application focus provides important constraints on the work and yields crisp criteria for making ontological decisions; i.e., does it make a difference for translation? Hovy has integrated the

[1] A small number of top-level nodes have been released for inspection by a broader community in recent months.

ontological portion with the over 50,000 term WordNet (Miller 1995) lexicon providing a very valuable resource. The ontology, however, is of limited use for purposes where the need for axiomatic information is critical such as planning or simulation because the classes contain no descriptive information other than their location in the taxonomy.

There are several substantial efforts underway in the medical informatics community to construct a unified ontology of medical terminology (Lindberg, Humphreys, and McCray 1993). The need to share medical records makes a unified terminology critical. These ontologies are basically large structured glossaries augmented with an encoding scheme. They form an important building block for future ontological work. For instance, they could be augmented with additional axiomatic information to support information integration. The existing projects might well benefit from the technology being proposed in this project.

Interoperable Knowledge Base Efforts

In addition to the Ontolingua work at KSL, there have been several efforts aimed at supporting interoperation between knowledge bases.

The DARPA Knowledge Sharing Effort (KSE) (Patil et al. 1992)is a collaboration of several leading knowledge representation research groups who have been looking for ways to increase our ability to share represented knowledge across systems. One key result of that effort has been the Knowledge Interchange Format (KIF) (Genesereth and Fikes 1991; Genesereth and Fikes 1992), a highly expressive formally defined declarative interchange language. KIF is currently being standardized by ANSI. KSL has been an active participant in the knowledge sharing effort and co-developers of the KIF language. The Ontolingua system uses an extended version of KIF as its internal representation language and includes in its library the KIF ontologies for sets, numbers, tuples, etc.

KIF embodies one approach to sharing knowledge – a declarative interchange language. Another approach is to provide an application program interface (API) that can be used to transfer knowledge between programs. KSL has worked with SRI to develop an API for object-oriented knowledge bases called the Generic Frame Protocol (GFP) (Karp, Myers, and Gruber 1995) and has subsequently extended this protocol to support network interaction. We have recently demonstrated networked interoperation of Ontolingua with SRI's knowledge base browsing and editing tools using the network GFP. It is worth noting that the expressiveness of GFP is very restricted when compared with full KIF. By using the network extensions to GFP, GKB can be used to interact with remote knowledge bases.

The generic knowledge base (GKB) editing and browsing tools developed at SRI nicely complement the approach that we have taken to providing distributed access to ontologies. GKB requires a powerful client machine and provides a direct manipulation graphical interface to browse and edit a knowledge base. This interface is restricted to the level of expressiveness that GFP provides., but is excellent for some classes of users and applications.

Knowledge Representation Tool Efforts

There have been a large number of efforts to develop knowledge representation tools including frame systems such as KEE, Clips, and Theo, description logic systems such LOOM (MacGregor 1991) and Classic (Borgida et al. 1989), and systems supporting full first order logic such as PTTP (Stickel 1992) and Epikit (Genesereth 1990). There has also been some work (e.g., Parka (Stoffel, Taylor, and Hendler 1996)) on integrating the expressiveness of a knowledge representation system with database technology to support larger knowledge bases. These systems are potential delivery vehicles for knowledge bases constructed from common ontologies. Existing knowledge bases developed for these systems can provide the starting point for sharable reusable ontologies.

We hope to exploit some of the underlying technologies that support large scale knowledge bases. These systems do not, however, address the issues of knowledge sharing or collaborative construction, although recent work on Loom has begun to address issues such as distributed access and translations into C++ and KIF.

Bibliography

Ackers, L. and B. Porter. (1994). Extracting Viewpoints from Knowledge Bases. In Proceedings of the Twelfth National Conference Artificial Intelligence (AAAI-94). Seattle, WA: AAAI Press/MIT Press.

Baalen, J. V. and R. E. Fikes. (1993). The Role of Reversible Grammars in Translating Between Representation Languages. KSL-93-67. Knowledge Systems Laboratory.

Borgida, A., R. J. Brachman, D. L. McGuinness, and L. A. Resnick. (1989). CLASSIC: A Structural Data Model for Objects. In Proceedings of the1989 ACM SIGMOD International Conference on Management of Data. Portland, Oregon.

Buvac, S., V. Buvac, and I. A. Mason. (1995). Metamathematics of contexts. Fundamenta Informaticae 23 (3).

Buvac, S. and R. Fikes. (1995). A Declarative Formalization of Knowledge Translation. In Proceedings of the ACM CIKM: The 4th International Conference on Information and Knowledge Management; Baltimore, Maryland; November 1995.

Clark, P. and B. Porter. (1996). Building Domain Representations from Components. AI96-241. University of Texas at Austin.

Erol, K., J. Hendler, and D. Nau. (1994). HTN Planning: Complexity and Expressivity. In Proceedings of the Twelfth National Conference on Artificial Intelligence. Seattle, WA: AAAI/MIT Press.

Falkenhainer, B., A. Farquhar, D. Bobrow, R. Fikes, K. Forbus, T. Gruber, Y. Iwasaki, and B. Kuipers. (1994). CML: A compositional modeling language. KSL-94-16. Stanford Knowledge Systems Laboratory.

Farquhar, A., R. Fikes, and J. Rice. (1996). The Ontolingua Server: a Tool for Collaborative Ontology Construction. In Knowledge Acquisition Workshop, ed. Brian Gaines. Banff,. Canada.

Fikes, R. Ontologies: What Are They, and Where's The Research?; Fifth International Conference on Principles of Knowledge Representation and Reasoning; Cambridge, Massachusetts; November 5-8, 1996.

Fikes, R., M. Cutkosky, T. Gruber, and J. van Baalen. (1991). Knowledge Sharing Technology Project Overview. KSL 91-71. Stanford University, Knowledge Systems Laboratory.

Fikes, R., A. Farquhar, and W. Pratt. (1996). Information Brokers: Gathering Information from Heterogeneous Information Sources. In Florida Artificial Intelligence Symposium (FLAIRS-96).

Finin, T., J. Weber, G. Wiederhold, M. Genesereth, R. Fritzson, D. McKay, J. McGuire, P. Pelavin, S. Shapiro, and C. Beck. (1993). Specification of the KQML Agent-Communication Language. EIT TR 92-04. Enterprise Integration Technologies, Palo Alto, CA.

Genesereth, M. R. (1990). The Epikit Manual. Epistemics, Inc. Palo Alto, CA.

Genesereth, M. R. and R. Fikes. (1991). Knowledge Interchange Format, Version 2.2. Logic-90-4. Computer Science Department, Stanford University.

Genesereth, M. R. and R. E. Fikes. (1992). Knowledge Interchange Format, Version 3.0 Reference Manual. Logic-92-1. Computer Science Department, Stanford University.

Ginsberg, M. L., ed. (1987). Readings in Nonmonotonic Reasoning. San Francisco: Morgan Kaufmann.

Glesner, S. and D. Koller. (1995). Constructing Flexible Dynamic Belief Networks from First-order Probabilistic Knowledge Bases. In Proceedings of the European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty (ECSQARU '95), ed. Ch. Froidevaux and J. Kohlas:217-226: Springer Verlag.

Gruber, T. R. (1993). A Translation Approach to Portable Ontology Specifications. Knowledge Acquisition 5 (2): 199-220.

Guha, R., D. Lenat, K. Pittman, D. Pratt, and M. Shepherd. (1990). Cyc: A Midterm Report. Communications of the ACP 33 (8).

Haddawy, P. (1994). Generating Bayesian Networks from Probability Logic Knowledge Bases. In Proceedings of the Tenth Annual Conference on Uncertainty in Artificial Intelligence (UAI '94):262-269.

Halpern, J. Y. (1990). An analysis of first-order logics of probability. Artificial Intelligence 46: 311-350.

Heckerman, D. (1986). Probabilistic interpretations for MYCIN's certainty factors. In Uncertainty in Artificial Intelligence, 2, ed. N. L. Kanal and J. F. Lemmer:11-22: Amsterdam: North Holland.

Heckerman, D. (1995). A Tutorial on Learning with Bayesian Networks. MSR-TR-95-06. Microsoft Research.

Hovy, E. H. and S. Nirenburg. (1992). Approximating an Interlingua in a Principled Way. In Proceedings of the DARPA Speech and Natural Language Workshop. Arden House, NY.

Karp, P., M. Riley, S. Paley, and A. Pellegrini-Toole. (1996). EcoCyc: Electronic encyclopedia of E. coli genes and metabolism. Nuc. Acids Res. 24 (1): 32-40.

Karp, P. D., K. Myers, and T. Gruber. (1995). The Generic Frame Protocol. In 14th International Joint Conference on Artificial Intelligence. Montreal, Canada.

Knight, K. and S. Luk. (1994). Building a Large Knowledge Base for Machine Translation. In Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94). Seattle, WA.

Lauritzen, S. L. and D. J. Spiegelhalter. (1988). Local computations with probabilities on graphical structures and their application to expert systems. Journal of the Royal Statistical Society B~50 (2): 157--224.

Lenat, D. (1995). Cyc: A Large-Scale Investment in Knowledge Infrastructure. Communications of the ACM 38 (11).

Levy, A. Y. (1993). Irrelevance Reasoning in Knowledge Based Systems. Ph.D., Stanford University.

Lindberg, D. A. B., B. L. Humphreys, and A. T. McCray. (1993). The Unified Medical Language System. Methods of Information Medicine 32 (4): 281-291.

Low, C. M. and Y. Iwasaki. (1992). Device Modeling Environment: An integrated model formulation and simulation environment for continuous and discrete phenomenon. In Proceedings of the First International Conference on Intelligent Systems Engineering: The Institute of Electrical Engineers, London.

MacGregor, R. (1991). The Evolving Technology of Classification-based Knowledge Representation Systems. In Principles of Semantic Networks: Explorations in the Representation of Knowledge, ed. John Sowa:385-400. San Mateo, CA: Morgan Kaufmann.

McCarthy, J. and S. Buvac. (1994). Formalizing Context (Expanded Notes). Technical Note STAN-CS-TN-94-13. Stanford University.

McGuire, J. G., D. R. Kuokka, J. C. Weber, J. M. Tenenbaum, T. R. Gruber, and G. R. Olsen. (1993). SHADE: Technology for knowledge-based collaborative engineering. Journal of Concurrent Engineering: Applications and Research (CERA) 1 (2).

Miller, G. A. (1995). WordNet: A Lexical database for English. Communications of the ACM, November, 39-41.

Mitchell, T. M., J. Allen, P. Chalasani, J. Cheng, O. Etzioni, M. Ringuette, and J. C. Schlimmer. (1989). Theo: A Framework for Self-Improving Systems: National Science Foundation, Digital Equipment Corporation.

Mowbray, T. J. and R. Zahavi. (1995). The ESSENTIAL CORBA: System Integration Using Distributed Objects.: John Wiley and Object Management Group.

Patil, R. S., R. E. Fikes, P. F. Patel-Schneider, D. McKay, T. Finin, T. R. Gruber, and R. Neches. (1992). The DARPA Knowledge Sharing Effort: Progress report. In Principles of Knowledge Representation and Reasoning: Proceedings of the Third International Conference. Cambridge, MA: Morgan Kaufmann.

Pearl, J. (1988). Probabilistic Reasoning in Intelligent Systems: Morgan Kaufmann.

Poole, D. (1993). Probabilistic Horn Abduction and Bayesian Networks. Artificial Intelligence 64 (1): 81-129.

Rice, J., A. Farquhar, P. Piernot, and T. Gruber. (1996). Using the Web Instead of a Window System. In Conference on Human Factors in Computing Systems (CHI96):103-110. Vancouver, CA: Addison Wesley.

Rickel, J. and B. Porter. (1994). Automated Modeling for Answering Prediction Questions: Selecting the Time Scale and System Boundary. In Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94):1191-1198: AAIT/MIT Press.

Shortliffe, E. H. (1976). Computer-Based Medical Consultations: MYCIN: Elsevier/North-Holland.

Stickel, M. (1992). A Prolog Technology Theorem Prover: a New Exposition and Implementation in Prolog. Theoretical Computer Science 104: 109-128.

Stoffel, K., M. Taylor, and J. Hendler. (1996). Parka-DB: Integrating Knowledge and Data-based Technologies. In Knowledge Representation and Reasoning.

Wellman, M. P., J. S. Breese, and R. P. Goldman. (1992). From Knowledge Bases to Decision Models. The Knowledge Engineering Review 7 (1): 35-53.