AFRL-IF-RS-TR-2006-30
Final Technical Report
February 2006

AUTOMATED TOOLS FOR MAPPING AMONG ONTOLOGIES

Yale University Computer Science Department

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

STINFO FINAL REPORT

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. AGENCY USE ONLY (Leave blank)	**2. REPORT DATE** FEBRUARY 2006	**3. REPORT TYPE AND DATES COVERED** Final Jun 2000 – Jul 2005

4. TITLE AND SUBTITLE
AUTOMATED TOOLS FOR MAPPING AMONG ONTOLOGIES

5. FUNDING NUMBERS
C - F30602-00-2-0600
PE - 62301E
PR - DAML
TA - 00
WU - 09

6. AUTHOR(S)

Drew McDermott

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Yale University Computer Science Department
P. O. Box 208285
New Haven Connecticut 06520-8285

8. PERFORMING ORGANIZATION REPORT NUMBER

N/A

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Force Research Laboratory/IFSA
525 Brooks Road
Rome New York 13441-4505

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

AFRL-IF-RS-TR-2006-30

11. SUPPLEMENTARY NOTES

AFRL Project Engineer: John Lemmer/IFSA/(315) 330-3657/ John.Lemmer@rl.af.mil

12a. DISTRIBUTION / AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 Words)*
The Yale/BBN effort was focused on developing techniques for implementing and managing ontology translations. The simplest case is translation of datasets or queries expressed using one ontology into datasets or queries expressed in another. They devised and implemented the merging-deduction-projection model: Bridging axioms are stored in merged ontologies; a set of facts to translate are used to start forward chaining using the bridging axioms; conclusions in the target ontology are retained.

14. SUBJECT TERMS
Human factors uncertainty and analysis

15. NUMBER OF PAGES
23

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

Abstract

The Yale/BBN effort was focused on developing techniques for implementing and managing ontology translation. The simplest case is translation of datasets or queries expressed using one ontology into datasets or queries expressed in another. They devised and implemented the *merging-deduction-projection* model: *Bridging axioms* are stored in *merged ontologies*; a set of facts to translate are used to start forward chaining using the bridging axioms; conclusions in the target ontology are retained. This model was implemented as the Ontomerge translation system. It used a notation called Web-PDDL, which provides explicit representation of namespaces and component ontologies. Programs were devised to translate between RDF/Owl syntax and Web-PDDL. The system was tested against large datasets in the domains of genealogy and geography.

Table of Contents

1 Summary

2 Introduction

The purpose of the Yale/BBN project[1] was to devise tools for managing *ontology translation*. *Ontologies* are axiom systems that make explicit (some of) the intended meaning of a declarative vocabulary. These axioms can be used by software agents to draw inferences from statements expressed in that vocabulary. Because it is unlikely that all agents will use the same vocabulary, they must be able to use *translation agents* to communicate with each other. Such an agent takes a body of facts (a *dataset*) expressed in one vocabulary (backed by the *source* ontology), and derives a dataset expressed in the other (which uses the *target* ontology), where the derived dataset is intended to capture as much of the meaning of the source dataset as possible.

The goal of our work was *not* to find ways of matching ontologies, that is, finding which symbols in one had meanings related to which symbols in the other. Instead, we wanted to see what would happen if all the correspondence rules were given. We believed that many challenging problems would still exist, including:

- What should the format of the rules be?

[1]Kestrel Institute was involved in the first year, but we had insufficient funds to maintain meaningful collaboration with them.

- How do you cope with ontologies that analyze the world in fundamentally different ways? (Example: Ontology 1 assumes different propositions are true at different time points; ontology 2 assumes a static world.)
- Can translation be considered a web service?
- How you do manage changes in an ontology over time?
- In the long run there will be many ontologies, and many rule sets relating them; can they be composed, and can they be kept consistent?

3 Methods, Assumptions, and Procedures

We adopt the *deductive model* of ontology translation, in which translation is seen as a two-phase process:

1. Infer as many facts as are practicable from S, the source dataset.
2. Discard facts that are not in the target vocabulary.

Obviously, these two phases do not occur purely sequentially.

3.1 Deduction

Our model of inference for translation is first-order deduction, which raises a couple of issues. It is always somewhat suspicious when deduction is proposed as the core of an inference technique, because almost all inferences are nondeductive. Certainly if we construe “translation” in a broad sense, so that it includes, say, translating French poetry into Urdu, then of course our whole model is inadequate. Fortunately, we’re concerned with much more boring domains, such as business transactions or inventory records. Here it is a plausible working hypothesis that the translation of a dataset contains no more information than was present in the original dataset. (Keep in mind that an inference is *deductive* if what is inferred is true in all models of the premises.)

Of course, our model does *not* assume that the translation of a dataset is equivalent to that dataset. Instead, it makes explicit the model assumed by traditional approaches to query translation for databases. If Q_B is a query in the vocabulary (and schema) for database B, and we try to find answers to it using database A, we need to find a query Q_A such that any answer to Q_A is an answer to Q_B. But obviously there are other queries in the language of database A that are the translations of different B queries. So consider the set $\mathcal{Q}_B$ of all B queries that have A translations, and let $\mathcal{Q}_A$ be the set of all those translations. The set of all answers to members of $\mathcal{Q}_A$ is the maximal amount of information in A that can be expressed in the vocabulary of B. But nothing guarantees that this information is equivalent to A. Some A facts simply can't be expressed in the vocabulary of B. If this isn't a problem in the backward-chaining ("query translation") context, then it needn't be a problem in the forward-chaining ("dataset translation") context.

So translation means deducing *as much information as possible* expressed in the target ontology from a dataset expressed in the source ontology. There is no rigorous test for whether the deduction has achieved this goal; the only way to judge is by implementing our approach and trying it on realistic examples.

3.2 Merged Ontologies

The cleanest way to approach inter-ontology deduction is to reduce it to the intra-ontology case. That is, join all the concerned ontologies into one big ontology, and add axioms that express the relationships among symbols from the different components. We call the result a *merged ontology.* To avoid name collisions, we use namespace prefixes as they are used in XML. We prefix a symbol with the characters "`@`n`:`" to indicate that it is to be understood as relative to namespace n. Of course, if we prefix all the symbols and do nothing else then we will have the inert union of a set of ontologies. To express translation rules we provide further axioms that belong to the merged ontology itself. There is nothing special about these axioms except that they contain symbols from more than one ontology. They can be used for translation, but they can also be used to conduct ordinary business in the merged ontology. We call them *bridging axioms,* but, to say it again, they don't look any different from any other axioms.

An example of a bridging axiom comes from a use case involving translating genealogical facts between two ontologies. One, G1, has predicates `husband` and `wife`; the other, G2, has predicates `(gender` *person male-or-female*, and `(married` *person1 person2*`)`. The relationship between them is captured by the bridging axioms:

```
(forall (?x ?y - Person)
   (iff (@g1:husband ?x ?y)
        (and (@g2:gender ?x @g2:male) (@g2:married ?x ?y))))
```

```
(forall (?x ?y - Person)
   (iff (@g1:wife ?x ?y)
        (and (@g2:gender ?x @g2:female) (@g2:married ?x ?y))))
```

We express the axioms using first-order logic, rendered in Lisp-like syntax. (See section 3.3.) We decided against description logics because we didn't want to be constrained by their tight restrictions on what can be expressed. Of course, that makes our inference problem tougher in principle.

Now we can be a bit more specific about our dataset-translation algorithm: A dataset is a collection of facts in the merged ontology. The algorithm draw conclusions from them using forward chaining:
If θ unifies every p_i and p'_i, then

$$\frac{\textit{From: } p'_1, p'_2, \ldots, p'_k \textit{ and } p_1 \wedge p_2 \wedge \ \ldots \wedge p_k \supset q_1 \wedge \ldots q_m}{\textit{Infer: } \theta(q_1) \wedge \ldots \theta(q_m)}$$

(All rules can be put in the form this implication exemplifies, which we call *implicational normal form.*) An inference chain stops when it reaches a conclusions that uses only symbols from the target ontology. All such conclusions are collected to form "the" translation of the dataset; other conclusions are discarded. We call this second step *projection,* by analogy with projection into a vector subspace.

3.3 Web-PDDL

We built our notation for data and inferences on top of PDDL, the Planning Domain Definition Language (McDermott 1998). At the microlevel this notation looks a lot like KiF (Genesereth and Fikes 1994) and Common Logic (Hayes and others 2004): it's a Lisp-like syntax applied to logic in the obvious way. However, it differs in a few points:

1. It provides a built-in notation for hierarchies of ontologies – called *domains.*

2. It is strongly typed, and type checking of a domain's formulas is performed when the domain is defined.

3. It existed when we began our research; Common Logic didn't. (Common Logic is the resurrection of KiF, which had been dormant for years when our project began.)

In the end, we added so many features to PDDL that we gave it a new name, "Web-PDDL." Here are the design changes we made:

1. We allowed domains to be named using URLs, so they could be loaded across the Web.

2. We introduced namespace prefixes (as described in the previous section).

3. During the course of our implementation, compatibility with the Owl ontology language (McGuinness and van Harmelen 2004) became more and more important. We initially hoped that PDDL types might map without change onto Owl classes, but the PDDL type hierarchy does not allow for multiple type membership. In Web-PDDL, this constraint is eliminated, so types are Owl classes for all practical purposes.

4. We added several convenience features to PDDL, including the capability for expressing arbitrary first-order axioms in the `:axioms` field of a domain definition. (For technical reasons, PDDL allows only a restricted syntax in its `:axiom` (and `:derived` predicate) declarations.)

It may seem that Web-PDDL is competing with Common Logic, and therefore shouldn't exist, it is actually the case that the two systems have evolved in parallel to a striking degree. One can think of Web-PDDL as a type-directed syntax-checking front end for Common Logic, plus an inference engine that uses types at unification time.

3.4 Ramifications

We now have a complete model for dataset translation[2]: deduce and project. We can use a dual approach for query translation: given a query in a source ontology, backward chain through bridging axioms until subgoal queries are reached that use only vocabulary items in the target ontology. These are then answered and the resulting bindings passed back as the answers to the original query.[3] This process is essentially the same as the query-translation problem studied by database researchers, and we haven't much to contribute

[2]We have often used the term *ontology translation* to mean translation of datasets from one vocabulary to another, but that choice leaves us with no good term for the process of translating an entire ontology. So in this report we will use the term *dataset translation* for the narrower-scoped problem.

[3]We have neglected the problem of how to translate terms occurring in the answer that aren't in the source language; see sect. 4.2.

to that discussion. We merely stop to point out a duality relating to the distinction between dataset and query translation. Suppose we want to translate $P \wedge Q$; we just assert both P and Q, deduce consequences, and project. Similarly, for $P \vee Q$, it is a plausible strategy to infer from P, infer from Q, and take the intersection of the results. (If all rules are Horn clauses, this will be a complete strategy.) But the relationship between the translations of $\not P$ and the translation of P is not so direct. We can't translate P and then slap a not-sign in front of the result. Instead, we must take notice of the fact that making inferences from $\not P$ is isomorphic to the series of backward chainings from P that occur when P is pursue as a query. (A backward-chaining goal is just a disguised negation as in resolution theorem proving.) In other words, when the translation relation crosses a negation boundary, it flips from data translation to query translation or vice versa.

Most databases accessible by web services do not contain negated statements, and so in the normal course of dataset translation this problem does not come up. However, it does present a technical obstacle for the translation of complex axiom systems. In principal we can draw inferences from a heavily quantified axiom in order to arrive at the corresponding axiom in the target ontology, but a state-of-the-art theorem prover will probably be unable to do this task. We have developed a more promising approach, which we will describe in section refsec:extensions

3.5 Implementation

Our implemented system is called *Ontomerge.* Deduction is performed by a first-order theorem prover called *Ontoengine,* is written in Java. It manipulates Web-PDDL statements, and can run in forward- or backward-chaining mode. In our experience, the sorts of inferences we need to make are focused on the following areas:

- Forward chaining from facts in source ontology to facts in target ontology.
- Backward chaining from queries in one ontology to get bindings from datasets in another.
- Introduction of skolem terms from existential quantified variables or skolem functions.
- Use of equalities to substitute existing constant terms for skolem terms.

OntoEngine is specialized for these sorts of inference. It uses generalized Modus Ponens chaining through bridging axioms with specified directions. To avoid infinite loops, we set a limit to the complexity of terms that OntoEngine generates; and, of course, OntoEngine stops when it reaches conclusions (or, in the case of backward chaining, goals) in the target ontology, which is called *target control.* Target control can avoid some redundant inference back from the target ontology to the source ontology. In addition, OntoEngine has a good type-checking system based on the strongly typed feature of Web-PDDL. The type-checking system can be used in both forward and backward chaining and can terminate blind alleys at the unification stage, without generating goals to prove that a term is of the correct type. If a variable is declared to be of class C, and a term has class C', then the two can unify if and only if C and C' are not disjoint. Putting this test in the unifier eliminates a whole class of deductive goal that would otherwise have to be dealt with.

OntoEngine can use equalities to substitute existing constant terms for skolem terms or other general function terms. Equality substitutions can decrease redundant inference results, such as redundant facts and queries. In OWL ontologies, equalities occur mainly in *cardinality axioms,* which state that there is exactly one or at most one object with a given property.[4] For example, in a genealogy ontology, there are two predicates `husband` and `wife`, whose cardinality axioms say that one family has only one husband and only one wife. The cardinality axiom about `husband` can be expressed in Web-PDDL:

```
(forall (f - Family h1 - Male h2 - Male)
      (if (and (husband f h1)
                (husband f h2))
          (= h1 h2)))
```

It is important to compare OntoEngine with other inference systems, such as Datalog systems, description logic systems and resolution theorem provers, which may be used to do reasoning with bridging axioms to implement semantic translations. The comparisons also can explain why we designed and built OntoEngine rather than use other existing inference systems.

A Datalog system can do backward chaining with Prolog-like rules to answer queries using view relations in databases (Pottinger and Halevy 2001). To avoid generating an infinite number of answers, Datalog rules are required to satisfy some *safety* conditions (Silberschatz et al. 2002). Hence, there are not any existentially quantified variable in the head (conclusion)

[4]Actually, you can specify other cardinalities, but it is pretty rare to do so.

side of a Datalog rule and Datalog systems don't have any mechanism to generate skolem terms or do equality substitution. However, relationships between concepts from different ontologies may require bridging axioms with existentially quantified variable in the conclusion side, such as the bridging axiom about `booktitle` in section 2.2. OntoEngine can generate skolem terms and do equality substitution to avoid redundant answers so that it can handle such kind of complicated axioms.

Description logics (Baader et al. 2003) are subsets of first order logic. Compared to the standard predicate calculus, the expressivity of description logic is limited, in order to guarantee the decidability of inference. There is a tradeoff between the expressivity of a representation language and the difficulty of reasoning over the representation built using that language. Although description logic (DL) reasoning systems are usually quite efficient, sometimes guaranteeably so, there is no model of forward inference in DLs, only various forms of query answering. It's not clear how dataset translation would work in a DL framework, and in particular how one would use Skolem terms as the incarnation of exisential quantifiers.

It is very common for bridging axioms to require existentials. For instance, suppose ontology H has a unary predicate `(is-married ?x)` and the other has a binary predicate `(married ?x ?y)`. The bridging axiom relating them is

```
(forall (x - Person)
   (iff (@h:is-married x)
        (exists (y - Person) (@g2:married x y))))
```

In translating from the unary to the binary predicate it is necessary to introduce a skolem term. If we know `(@h:is-married sally)`, that gets translated into `(exists (y) (@g2:married sally y))`, or, using Skolem terms, `(@g2:married sally (skolem y 34 sally))`. Sometimes it's convenient to specify an explicit name for such skolem terms. This rule (the "only if" part) would come out thus:

```
(forall (x - Person)
   (if (@h:is-married x)
       (@g2:married x (@skolem:spouse x))))
```

OntoEngine is not a complete first-order theorem prover, unlike resolution-based systems, such as Otter (Wos 1996). One reason (besides our obvious desire for efficiency) is that we have empirically observed that some deductive techniques are not necessary for ontology translation. Most important, so far we have had little need for *case analysis,* in which a proposition is proved by showing that it follows from A and from B, when $A \vee B$ is the strongest conclusion that can be drawn about A and B.

3.6 Web Services

Although it was not part of our original proposal, our group became heavily involved in the effort to develop an ontological framework for web services, under the label "DAML-S," which was changed to "OWL-S," and is now transitioning to the SWSF — Semantic Web Services Framework. What all these have in common is an attempt to characterize all facets of web services using declarative notations, in order to facilitate automated reasoning about them. Mark Burstein and Drew McDermott have been members of the OWL-S group, and Burstein now chairs the SWSA (SWS Architecture) subpanel of the SWSF group.

1. We developed a "presentation syntax" for OWL-S, which is much more pleasant for human consumption, including a parser that translates it into the RDF/XML format some people seem to insist on for some reason.

2. We applied our model of translation to several scenarios that arise during execution of web-service interaction plans.

3. We wrote a preliminary version of a planner for dealing with web services.

The presentation syntax is specified using a precedence-grammar formalism of our own devise, called Lexiparse (McDermott 2004b). The grammar takes the form of an executable parser. It transforms verbose XML into simple process specs such as

```
define composite process authorize(inputs: (cc - CreditCard,
                                             amt - Currency),
                                  outputs: (okay - Boolean))
   {
     b :: check_auth(ccnum <= cc);
     if (b.okay)
       then {
             a :: send_query(recip <= b.authorizer,
                             msg <= lim_check(amt))
```

```
            produce(okay <= a.reply = Yes)
          }
      else produce(okay <= false)
}
```

We were able to do some proof-of-concept experiments applying planning for interaction with web services. See (McDermott 2002a; Burstein and McDermott 2005). Translation occurs naturally in this context by backward chaining through rules in the merged ontology, as discussed earlier.

4 Results and Discussion

4.1 Performance Experiments

To test the performance of the Ontoengine component of our system, we tried translating some large datasets. For a complete analysis, see (Dou 2004).

Experiment 1: Ontoengine translates a dataset[5] with 7564 facts about the geography of Afghanistan using more than 10 ontologies into a dataset in the `map` ontology:

```
http://www.daml.org/2001/06/map/map-ont.daml.
```

4611 facts are related to the geographic features of Afghanistan described by the `geonames` ontology:

```
http://www.daml.org/2002/04/geonames/geonames-ont.daml.
```

and its airports described by the `airport` ontology:

```
http://www.daml.org/2001/10/html/airport-ont.daml.
```

The facts about one particular airport of Afghanistan are formalized thus:

```
(@rdfs:label @af:OAJL "JALALABAD")
(@airport:icaoCode @af:OAJL "OAJL")
(@airport:location @af:OAJL "Jalalabad, Afghanistan")
(@airport:latitude @af:OAJL 34.399166666666666)
(@airport:longitude @af:OAJL 70.49944444444445)
```

[5]http://www.daml.org/2001/06/map/af-full.daml

Actually either of these two ontologies just partly overlaps with the map ontology. The main semantic difference between their overlapping with the map ontology is that in the map ontology, any location in a map is a point whether it is an airport or other kind of geographic feature such as a bridge. But in the airport and geonames ontologies, airports are a special category disjoint from location which is different from bridges, and airports are not a point. We have merged the geonames ontology and the airport ontology with the map ontology. One of bridging axioms in the merge of the airport ontology and the map ontology is below:

```
(forall (x - Airport y z  - Object)
    (if (and (@airport:latitude x y) (@airport:longitude x z))
        (and (location (@skolem:aPoint x y z) - Point
                       (@skolem:aLocation x y z) - Location)
             (latitude (@skolem:aLocation x y z) y)
             (longitude (@skolem:aLocation x y z) z))))
```

After OntoEngine loads the two merged ontologies and all 7564 facts in, those 4611 facts in the airport and geonames ontologies are translated to 4014 facts in the map ontology by inference. The translated dataset for the above airport is:

```
(@map:label Point31 "JALALABAD")
(@map:label Point31 "OAJL")
(@map:label Point31 "Jalalabad, Afghanistan")
(@map:location Point31 Location32)
(@map:latitude Location32 34.399166666666666)
(@map:longitude Location32 70.49944444444445)
```

As part of DAML Experiment 2002, the result can be used by a map agent (BBN's OpenMap) to generate a map image about the airports and geographic features of Afghanistan. The semantic translation (inference) process by OntoEngine, which contains 21232 reasoning steps, only takes 18 seconds (including the time for loading the input dataset and merged ontologies) on our PC in PIII 800MHZ with 256M RAM.

Experiment 2: OntoEngine translates a bigger dataset[6] with 21164 facts (on 3010 individuals and 1422 families of European royalty) in the bbn_ged genealogy ontology:

```
http://www.daml.org/2001/01/gedcom/gedcom.daml.
```

to 26956 facts in the drc_ged genealogy ontology:

```
http://orlando.drc.com/daml/Ontology/Genealogy/3.1/Gentology-ont.daml.
```

[6]http://www.daml.org/2001/01/gedcom/royal92.daml

Here are some facts in the bbn_ged ontology about a King of France :

```
(@bbn_ged:name @royal92:@I1248@ "Francis_II")
(@bbn_ged:sex @royal92:@I1248@ "M")
(@bbn_ged:spouseIn @royal92:@I1248@ @royal92:@F456@)
(@bbn_ged:marriage @royal92:@F456 @royal92:event3138)
(@bbn_ged:date @royal92:event3138 "24 APR 1558")
(@bbn_ged:place @royal92:event3138 "Paris,France")
```

Although these two genealogy ontologies are very similar and overlap a lot, there are still some differences. For example, in the drc_ged ontology, there are two properties wife and husband, but the most related concept in the bbn_ged ontology is the spouseIn property. If a person is a male (his sex is "M") and he is spouseIn some family which is related to some marriage event, he will be the husband of that family. We have written the bridging axioms for the bbn_ged and drc_ged ontologies to express such semantic differences. The one for the above example is given below.

```
(forall (f - Family h - Individual m - Marriage)
    (if (and (@bbn_ged:sex h "M") (@bbn_ged:spouseIn h f)
             (@bbn_ged:marriage f m))
        (husband f h)))
```

This merged genealogy ontology works well for semantic translation. After loading the input dataset and merged ontology, OntoEngine runs 85555 reasoning steps to generate all the 26956 facts. The whole process takes 59 seconds. The translated dataset for King Francis_II in the drc_ged ontology is:

```
(@drc_ged:name @royal92:@I1248@ "Francis_II")
(@drc_ged:sex @royal92:@I1248@ "M")
(@drc_ged:husband @royal92:@F456 @royal92:@I1248@)
(@drc_ged:marriage @royal92:@F456 @royal92:event3138)
(@drc_ged:date @royal92:event3138 "24 APR 1558")
(@drc_ged:location @royal92:event3138 "Paris,France")
```

The Ontomerge system has been used to run a translation servlet at Yale University. We are currently bringing up a new incarnation at the University of Oregon (where Dejing Dou is now located).

4.2 Limitations

Some of the research directions we intended to pursue had to be dropped for lack of time or other resources. In particular, we only scratched the surface of the issue of "representation clashes," in which two ontologies rest on contradictory assumptions about the world. A simple case of this kind of clash is where one ontology requires extra objects to define a situation that the other ontology can describe using a single predicate. As discussed in section 3.5, our system handles this using existential quantifiers and skolem terms. But now let ontology A be atemporal, while ontology B refers to changes over time. To be concrete, suppose A might represent a fact such as `(location` *object coordinates*`)` to represent, say, the locations of buildings; while B represents similar facts thus: `(holds (place` *object coordinates*`)` *timepoint*`)`. The following bridging axiom, which seems superficially plausible, is not right:

```
(forall (x - Physobj lat long - Float)
   (if (@A:location x lat long)
       (exists (t - Timepoint)
          (@B:holds (@B:place x lat long) t))))
```

In some ways this axioms confirms the strength of our approach. We have no trouble at all with the mapping of a predicate (`location`) to a function (`place`). The problem is that all the facts in a dataset using ontology A are true with respect to the *same* time point. So in translating a dataset using A to one using B, we will produce an overly weak result. The translation back is impossible: We can't change the main connective from an `if` to an `iff`, the resulting axiom is false, because from

```
(@B:holds (@B:place ob23 40.3 120.5) t1)
(@B:holds (@B:place ob23 50.9 121.5) t2)
```

we could infer

```
(@A:location ob23 40.3 120.5)
(@A:location ob23 50.9 121.5)
```

which puts `ob23` in two places at once.

Solving this problem requires being able to associate *dataset parameters* with datasets. In this case, any dataset using ontology A cannot be translated without specifying what time is assumed; this time is a *parameter* for that dataset. Its value would either be a particular time or a specification that the facts should be assumed to hold for all times. (If ontology B is normally used to create datasets talking about rapidly evolving situations, and ontology A is used to describe a dataset of geographical facts, then as far as B is concerned the A facts are eternal and unchanging.)

Another issue we weren't able to deal with was the problem of *term translation.* Suppose that an inference produces a formula all of whose predicates are in the target ontology, but which has functions still in the source ontology. For instance, suppose the source has a function `(top-of` b`)` that denotes the top of physical object b. If Ontoengine infers a formula `(`P `(@A:top-of` a`))` where P uses only target symbols, further chaining (forward or backward) is unlikely to help. What might be better is directed equality substitution (perhaps like that suggested by (McDermott et al. 2001)) to replace `(@A:top-of ...)` with a term drawn from the target ontology.

Finally, we had hoped to have enough merged ontologies on line that the ontology-management issues sketched in section 2 would become important. That didn't happen, primarily because we didn't have the manpower to address the opportunities that came our way. For instance, we developed a partnership with the Bioinformatics Group at the Yale Medical School to study merging of neuroscience databases. That effort is continuing, but slowly.

4.3 Extensions

Although some of our goals were not attained, we made up for that partially by discovering several novel directions our techniques could take us, including application to "semantic web services," and, the topic of this section, the translation of axioms. As we admitted above, forward chaining is not likely to produce a useful translation of a formula more complex than an atomic formula. In some cases we need to translate axioms, which are likely to be of the form `(forall (...) (if` A C`))`. Cases like this arise in *theory translation*, in which an entire ontology is translated, not just a dataset. One application of theory translation is in *ontology composition*, defined thus: Suppose that O_{1_2} is the merge of ontologies O_1 and O_2; O_{2_3}, of O_2 and O_3. Derive the merged ontology O_{1_3} of O_1 and O_3 from O_{1_2} and O_{2_3}?

Ontology composition can be thought of as an optimization for ontology translation tasks, especially for large sets of data. For example, if we have O_{1_2} and O_{2_3} and want to do dataset translation from O_1 to O_3. Without ontology composition, OntoEngine only can translate dataset from O_1 to O_2 first, then from O_2 to O_3. If we can get O_{1_3} before OntoEngine does dataset translation, the dataset in O_1 can be directly translated to O_3. This requires translating axioms using vocabulary O_2 to an axiom using only symbols from O_1 and O_3.

To translate an axiom, we replace all the universally quantified variables with Skolem constants. Then we take all the negated atomic formulas and do query translation on them, and do forward translation on the unnegated atomic formulas. An axiom of the form `(if` P Q`)` yields P' by translating P as a query, Q' by translating Q as an assertion. Then Skolem constants in P' and Q' are then turned back into variables, yielding an axiom of the form `(forall` *vars* `(if` P' Q'`))`. (For more details, see (Dou and McDermott 2005).)

5 Conclusions

We started with the working hypothesis that translation can be thought of as deduction in a merged ontology, followed by projection to the target ontologies. Based on our experiments with the Ontomerge system, we conclude that this is eminently practical. It's straightforward to express ontologies in a logical notation, and the deductive techniques required to make forward or backward chaining work can be implemented quite efficiently.

In order to interface to our Ontomerge system, it is necessary to use predicate calculus, as embodied in the Web-PDDL system. We developed a technique for embedding that notation — or any logical notation — in RDF (McDermott and Dou 2002). We wrote a program to translate Web-PDDL to RDF and back. This enables us to make use of existing ontologies, many of which consist entirely of Owl expressions.

However, although Owl has become a standard notation for ontologies, there seems to be a wide recognition that to gain further expressivity one must leave the world of RDF behind. That accounts for the momentum behind the development of Common Logic (Hayes and others 2004). Web-PDDL is consistent with CL, and can easily be considered to be a front end for it. Developing a translator to CL would be quite straightforward.

In parallel with the development of Web-PDDL, we have also developed a different successor of PDDL, called Opt (McDermott 2004). Opt involves a more sophisticated type system, which, in particular, allows for polymorphism. The relationship between Opt and CL is not quite so straightforward as that between Web-PDDL and CL. This is a matter for further investigation.

5.1 Future Work

There are many promising directions to pursue. The most pressing is to enlarge the complexity of ontologies that our deductive engine can handle. It has no trouble with the typical Owl ontology, perhaps augmented with a few small axioms. But a serious merged ontology will have axioms of considerably greater depth. This raises several issues, including:

- As mentioned in section 4.2, in general ontologies have "parameters," such as the implied time when merging a static ontology with a dynamic one. So far we have not devised a mechanism for supplying values for these parameters.
- When two ontologies have different "grain sizes," it is often the case that the axioms combining them are asymmetrical. Suppose ontology A expresses routes at the scale of a walking person, and B at the scale of tank maneuvers. We can translate a statement about a possible motion in B into a statement about a sequence of A motions. But how do we translate from a set of arbitrary statements in A to a set of statements in the vocabulary of B? One might propose that the translation from A to B can be done only "in reverse," that is, by translating B queries to vocabulary A. If so, our model requires revision.

We anticipated at the beginning of this research effort that a key result would be tools for managing a large collection of merged ontologies, whose components evolved over time. So far, our tools are quite rudimentary, but we believe the need for a more sophisticated toolset will inevitably arise.

References

Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter Patel-Schneider 2003 *The Description Logic Handbook; Theory, Implementation, and Applications.* Cambridge University Press

Mark H. Burstein and Drew V. McDermott 2005. Ontology Translation for Interoperability Among Semantic Web Services *AI Magazine* , **26**(1), 71–82

Dejing Dou 2004 *Ontology Translation by Ontology Merging and Automated Reasoning.* Ph.D. dissertation Unpublished Yale University

Dejing Dou and Drew McDermott 2005 Deriving axioms across ontologies. Submitted to Conf. on Info. and Knowledge Mgt

Michael R. Genesereth and Richard E. Fikes 1994 *Knowledge Interchange Format Version 3.0 Reference Manual*1. Available at `http://logic.stanford.edu/kif/Hypertext/kif-manual.html`

Patrick Hayes and others 2004 *SCL: Simple Common Logic*. Web document, available at `http://www.ihmc.us/users/phayes/SCLJune2004.html`

Drew McDermott 1998 The Planning Domain Definition Language Manual. Yale Computer Science Report 1165. (CVC Report 98-003) Available at `ftp://ftp.cs.yale.edu/pub/mcdermott/software/pddl.tar.gz`

Drew McDermott 2004 Draft opt manual. Available at `http://www.cs.yale.edu/~dvm/papers/opt-manual.ps`

Drew McDermott 2002. Estimated-regression planning for interactions with web services*Proc. AI Planning Systems Conference 2002*

Drew McDermott 2004 *Lexiparse: A Lexicon-based Parser for Lisp Applications*. Draft, available at `http://www.cs.yale.edu/homes/dvm/papers/parser-manual.pdf`

Drew McDermott and Dejing Dou 2002. Representing Disjunction and Quantifiers in Rdf *Proc. Int'l Semantic Web Conference*, pp. 250–263. LNCS 2342 Springer-Verlag

Drew McDermott, Mark Burstein, and Douglas Smith 2001. Overcoming ontology mismatches in transactions with self-describing service agents*The Emerging Semantic Web: Selected Papers from the First Semantic Web Working Symposium*, pp. 228–244. IOS Press

Deborah L. McGuinness and Frank van Harmelen (eds) 2004 *OWL Web Ontology Language Overview*. Available at `http://www.w3.org/TR/owl-features/`. W3C Recommendation

Rachel Pottinger and Alon Halevy 2001. A scalable algorithm for answering queries using views *VLDB Journal* , **10**, 182–198

Abraham Silberschatz, Henry F. Korth, and S. Sudarshan 2002 *Database System Concepts, 4th edition.* "Boston": McGraw-Hill

Larry Wos 1996 The Automation of Reasoning: An Experimenter's Notebook with Otter Tutorial. Academic Press

All Publications

Anupriya Ankolenkar, Mark Burstein, Jerry R. Hobbs, Ora Lassila, David L. Martin, Drew McDermott, Sheila A. McIlraith, Srini Narayanan, Massimo

Paolucci, Terry R. Payne, and Katia Sycara 2002. “Daml-s: web service description for the semantic web*Proc. Int’l Semantic Web Conf.* ”

Mark H. Burstein 2004. Ontology mapping for dynamic service invocation on the Semantic Web*Proc. 2004 AAAI Spring Symposium on Semantic Web Services*. Palo Alto: AAAI Press

Mark H. Burstein and Drew V. McDermott 2005. Ontology Translation for Interoperability Among Semantic Web Services *AI Magazine* , **26**(1), 71–82

Dejing Dou and Drew McDermott 2005 Deriving axioms across ontologies. Submitted to Conf. on Info. and Knowledge Mgt

Dejing Dou, Drew McDermott, and Peishen Qi 2003. Ontology translation on the semantic web*Proc. Int’l Conf. on Ontologies, Databases and Applications of SEmantics (ODBASE) 2003*, pp. 952–969. LNCS 2888 Springer-Verlag

Dejing Dou, Drew McDermott, and Peishen Qi 2005. “Ontology translation on the Semantic Web *LNCS Journal on Data Semantics* , **2**, 35–56. ”

Jerry R. Hobbs, George Ferguson, James Allen, Richard Fikes, Pat Hayes, Drew McDermott, Ian Niles, Adam Pease, Austin Tate, Mabry Tyson, and Richard Waldinger 2002 *DAML Ontology of Time*. `http://www.cs.rochester.edu/~ferguson/daml/`

David Martin, Massimo Paolucci, Sheila McIlraith, Mark Burstein, Drew McDermott, Deborah McGuinness, Bijan Parsia, Terry Payne, Marta Sabou, Monika Solanki, Naveen Srinivasan, and Katia Sycara 2004. Bringing semantics to web services: the Owl-s approach*Proc. First Int’l Workshop on Semantic Web Services and Web Process Composition (SWSWPC 2004)*

Drew McDermott 2002. Estimated-regression planning for interactions with web services*Proc. AI Planning Systems Conference 2002*

Drew McDermott 2004 Draft opt manual. Available at `http://www.cs.yale.edu/~dvm/papers/opt-manual.ps`

Drew McDermott and Dejing Dou 2002. Representing Disjunction and Quantifiers in Rdf *Proc. Int’l Semantic Web Conference*, pp. 250–263. LNCS 2342 Springer-Verlag

Drew McDermott, Mark Burstein, and Douglas Smith 2001. Overcoming ontology mismatches in transactions with self-describing service agents*The Emerging Semantic Web: Selected Papers from the First Semantic Web Working Symposium*, pp. 228–244. IOS Press

Drew McDermott, Mark Burstein, and Douglas Smith 2001. Overcoming ontology mismatches in transactions with self-describing agents*Proc. Semantic Web Working Symposium*, pp. 285–302