Ontology Translation on the Semantic Web[*]

Dejing Dou, Drew McDermott, and Peishen Qi

Yale Computer Science Department
New Haven, CT 06520, USA
{dejing.dou,drew.mcdermott,peishen.qi}@yale.edu

Abstract. Ontologies are a crucial tool for formally specifying the vocabulary and relationship of concepts used on the Semantic Web. In order to share information, agents that use different vocabularies must be able to translate data from one ontological framework to another. Ontology translation is required when translating datasets, generating ontology extensions, and querying through different ontologies. OntoMerge, an online system for ontology merging and automated reasoning, can implement ontology translation with inputs and outputs in OWL or other web languages. Ontology translation can be thought of in terms of formal inference in a merged ontology. The merge of two related ontologies is obtained by taking the union of the concepts and the axioms defining them, and then adding *bridging axioms* that relate their concepts. The resulting *merged ontology* then serves as an inferential medium within which translation can occur. Our internal representation, *Web-PDDL*, is a strong typed first-order logic language for web application. Using a uniform notation for all problems allows us to factor out syntactic and semantic translation problems, and focus on the latter. Syntactic translation is done by an automatic translator between Web-PDDL and OWL or other web languages. Semantic translation is implemented using an inference engine (*OntoEngine*) which processes assertions and queries in Web-PDDL syntax, running in either a data-driven (forward chaining) or demand-driven (backward chaining) way.

1 Introduction

One major goal of the Semantic Web is that web-based agents should process and "understand" data rather than merely display them as at present [24]. Ontologies, which are defined as the formal specification of a vocabulary of concepts and axioms relating them, are seen playing a key role in describing the "semantics" of the data. Using ontologies, web-based agents can treat web documents as sets of assertions, and, in particular, draw inferences from them.[1]

---

[1] We use scare quotes for "semantic" and "understand" because many people use the former term without knowing what it means, and no one knows what the latter term means. We will avoid the word "understand" wherever possible, but "semantic" in the recent sense of "seeming to reveal a grasp of the meaning of the terms occurring in," as in "semantic translation," seems unavoidable.

More and more ontologies are being developed [4] as formal underpinnings for RDF-based data. An obvious desideratum of this movement is that two ontologies should not cover the same area; instead, those interested in formalizing descriptions in that area should agree on a standard set of concepts. However, this goal cannot always be met, for a variety of reasons. Some standard vocabularies arise in different parts of the world or among different linguistic communities, and attain popularity before their overlap is noticed. Even more likely is that two vocabularies will partially overlap, usually because what is central to one is peripheral to the other. For instance, a vocabulary devised by glove manufacturers will have something to say about the parts of the hand and how they mesh with the parts of gloves. A vocabulary devised by orthopedic surgeons will also talk about the parts of the hand, but in very different ways, including, obviously, going into much more detail.

These vocabulary differences make life difficult for agents on the Semantic Web. It is much simpler to program two agents to communicate if they use the same vocabulary and the terms have same meaning or interpretation. But in cases where their vocabularies differ, we must resort to *ontology translation* to allow them to communicate. This paper describes our contribution to the solution of this problem.

Our focus is on translation between languages that use the same syntax but have different vocabularies. We will have little to say about eliminating syntactic differences, and instead will generally assume that the facts or queries to be translated will be in the same logical notation after translation as before; only the vocabulary will change. Syntactic translation is not always trivial, but we will assume it is solved.

1.1   The Differences Between Ontologies on Similar Domains

The kinds of semantic differences between ontologies are innumerable, and rarely correspond to simple correspondences between symbols in one and symbols in the other, as we will discuss in section 1.2. For example, one genealogy ontology might use two properties — firstname and lastname — to represent a person's name, where another might use only one property, fullname.

Some more subtle examples arise in translation between two bibliographical ontologies developed at Yale [18] and CMU [3].[2] While they are both obviously derived from the Bibtex terminology, different decisions were made when ontology experts developed them.

EXAMPLE 1.1.1. Both ontologies have a class called Article. In the yale_bib ontology, Article is a class which is disjoint with other classes such as Inproceedings and Incollection. Therefore, in the yale_bib ontology, Article only includes those articles which were published in a journal. But in the cmu_bib ontology, Article

---

[2] These are "toy" ontologies developed as exercises to help populate the DARPA DAML ontology library [4]. They are much simpler than real-world ontologies would be. Their very simplicity helps reveal the phenomena we are after.

includes all articles which were published in a journal, proceedings or collection. There are no Inproceedings and Incollection classes in the cmu_bib ontology.

Complicated semantic differences can be caused by different understandings about similar concepts. Even if the concepts from two ontologies share the same class or property name, it is still possible that they have quite different meanings. The following example is about the booktitle property in the yale_bib ontology and cmu_bib ontology.

EXAMPLE 1.1.2. In the cmu_bib ontology, booktitle's domain is the Book class and its range is String. A booktitle relationship means that a Book has some string as its title. In the yale_bib ontology, booktitle's domain is Publication and its range is Literal, which can be taken to be the same class as String. However, yale_bib's booktitle domain is Publication. The assertion that publication P has booktitle S means that P was published in a conference proceedings or anthology, and that it is this proceedings or collection that has S as its title.

Another reason for complicated semantic differences is that they can be inherited from those between basic concepts, such as time, space etc.

EXAMPLE 1.1.3. There are several ontologies about time, such as DAML Time [6] and the time ontology in OpenCyc [11]. Those time ontologies have semantic differences among their concepts, such as events. Two genealogy ontologies, one based on DAML Time and the other on Cyc, might take identical positions on all design choices about specifically genealogical questions, but look different because of their differing assumptions about time, as expressed in the way they make assertions about genealogically relevant events such as birth and marriage.

1.2   The Relationship Between Ontology Translation and Ontology Mapping

It's important to distinguish ontology translation from *ontology mapping*, which is the process of finding correspondence (mappings) between the concepts of two ontologies. If two concepts correspond, they mean the same thing, or closely related things. Obviously, finding such mappings can be a valuable preprocessing step in solving the ontology-translation problem for the two ontologies. Automating the process of ontology mapping is an active area of research [40,42,27,46].

However, the emphasis on finding mappings has led, we believe, to a distorted view of the translation problem. Suppose one starts by assuming a particular notation for ontologies, such as OWL [12]. These notations are often represented visually as graph structures. Then it is natural to express a mapping between two ontologies as a network of "meta-links" that join nodes and links in one ontology graph with nodes and links in the other [36]. Each such link can be annotated with labels specifying whether the two constructs it joins mean exactly the same thing, or which of them covers a larger set of objects or relationships. It seems almost inevitable at that point to think of translation as a process of substitution of labels. One might identify a role R_1 in one ontology with a role R_2 in another. (In the terminology of relational databases, the columns R_1 and R_2 would be labelled as equivalent.) Translation then becomes a matter of relabelling data.

Complexities arise when the labels aren't exactly equivalent. Two classes might be connected by a subclass link instead of being exactly equivalent.

Meanwhile, there is a competing paradigm for translation, centered around the problem of answering queries in federated databases. Here the problem is that a query asked with respect to one schema (say, a central ontology) may require retrieval from a database using a different schema. This process is usually thought of as translating the *query* rather than translating the data used to answer it. The rules required to translate a query are essentially logical axioms (or logic-programming rules) of the form $A \leftarrow B$, where A is in the query vocabulary and B is in the vocabulary of the remote database [35,41].[3] The reason that this is not normally described as translating data from notation B is that the spotlight is not on arbitrary data expressed in that notation. Instead the main question is, Does the query result in finding all the bindings for its free variables that follow from the contents of the B database?

In this and our previous papers [28,29], we explore the idea of abolishing the distinction between "ontology translation" and "query translation," assimilating the former to the latter. Translation may be thought of abstractly as a two-stage process: *Inference* and *projection*. That is, to translate a set of facts S_1 expressed in ontology 1 to ontology 2, we draw all possible inferences from S_1, then *project* the results into the vocabulary of S_2. That is, we discard conclusions that are not in the *target vocabulary* used by S_2.

As stated, this proposal is highly schematic, and we will explain how it gets instantiated into a practical algorithm. In this section, however, the discussion will stay at a fairly abstract level. The first remark to make is that query translation already fits our paradigm, using backward chaining to keep the inference headed toward relevant results.

However, backward chaining is not always the best option, or even a feasible option. If an agent wants to do data mining on information supplied by another, then it won't know ahead of time what set of queries to use. Or an agent might be asked for a description of an object it is trying to find (e.g., "a recent book by Neal Stephenson"). It has in essence a small dataset that it wants to translate in its entirety to the language of the agent that asked for the description. In such cases we do *not* throw the infer-and-project framework overboard, we just use it in a forward direction.

From this point of view, translation rules are just axioms. The purpose of ontology mapping should be to find these axioms. Finding correspondences between symbols is just the first step. Unfortunately, in many papers, it often seems to be the last step. We seek to tilt in the other direction by focusing on the axioms and not worrying about automatic mapping at all. Unlike many other approaches (e.g., [27]), we do not manage axioms as links between otherwise disparate ontologies. Instead, we work in the *merge* of two related ontologies, obtained by taking the union of the concepts and the axioms defining them, and using XML

---

[3] We're oversimplifying somewhat here, by assuming that axioms define A predicates in terms of B predicates. In the competing, "local as view" paradigm, the axioms run the other way, and query translation is more difficult. See [44].

namespaces [17] to avoid name clashes. *Bridging axioms* are then added to relate the concepts in one ontology to the concepts in the other through the terms in the merge. This framework frees us from having to think in terms of just two ontologies joined by a slender "bridge." Instead, as many ontologies as are relevant can't be mixed together, and the bridging axioms can relate terms from these ontologies in whatever ways seem appropriate.

Devising and maintaining a merged ontology is a job for human experts, both domain experts and "knowledge engineers." Once the merged ontology is obtained, ontology translation can proceed without further human intervention. The inference mechanism we use is a theorem prover optimized for the ontology-translation task, called *OntoEngine*. We use it for deductive dataset translation (section 3), ontology-extension generation(section 4), and query handling through different ontologies (section 5). We will also discuss related work in section 6. In the last section, we will give some conclusions for our work so far and discuss our future plans for developing interactive tools for ontology merging based on our recent work on integrating different neuronal databases.

1.3 Three Kinds of Ontology Translation Problems

As we said above, we focus on three kinds of ontology translation problems: dataset translation, ontology-extension generation and querying through different ontologies.

Dataset translation can be defined as the translation of a "dataset" from one ontology to another. We use the term *dataset* to mean a set of facts expressed in a particular ontology [38]. The translation problem arises when web-based agents try to exchange their datasets but they use different ontologies to describe them.

EXAMPLE 1.2.1. Suppose there is a web-based agent which uses the cmu_bib ontology to collect and process the bibliography information of researchers in the area of computer science. A web-based agent at Yale can provide such information about publications by members of the Yale CS department. The CMU agent needs an ontology translation service to translate those datasets into the cmu_bib ontology before it can combine them with information it already has.

The problem of *ontology extension generation* is defined thus: given two related ontologies O_1 and O_2 and an extension (sub-ontology) O_{1s} of O_1, construct the "corresponding" extension O_{2s}.

EXAMPLE 1.2.2. DAML-S [5] is a general ("upper") ontology describing web services at the application level (i.e., focusing on business processes), and WSDL Schema [16] is another general ontology describing web services at the communication level (i.e., focusing on messages and protocols). To use DAML-S to model a particular service, one must manually develop a sub-ontology that uses the DAML-S vocabulary to describe, say, a book seller's or airline's web service. But the description of a service is not complete or usable until it extends all the way down to the communication level. In other words, given the DAML-S sub-ontology for Congo.com, we would like to derive the "analogous" sub-ontology of WSDL Schema. To oversimplify, if property P_D maps to P_W at the upper level, and if P_{DC} is a sub-property of P_D belonging to the Congo ontology,

we should be able to infer the corresponding sub-property P_{WC} of Congo.com's counterpart at the communication level. This process is part of *grounding*, to use the terminology of [22].

Finally, we have the problem of *querying through ontologies*, in which a query made by one agent has the potential to be answered by another agent (perhaps a database manager) that uses a different ontology. answer a query

EXAMPLE 1.2.3. Suppose a web agent using the drc_ged [8] genealogy ontology wants to find the marriage date of King Henry_VI of England. It finds a knowledge base that it has reason to trust, which contains information about the individuals and families of European royalty, but it uses a different genealogy ontology, bbn_ged [2]. Ontology translation is required for the agent to get its query answered.

2   Our Approach: Ontology Merging and Automated Reasoning

In this section, we flesh out the inference-then-projection schema, yielding an approach we call *ontology translation by ontology merging and automated reasoning.* We have developed *Web-PDDL* as a strongly typed, first-order logic language to describe axioms, facts, and queries, which we use as our internal representation language for the formal inference. We have also designed and implemented a first-order theorem prover, *OntoEngine*, which is optimized for the ontology-translation task.

2.1   Separate Syntactic and Semantic Translation

Past work [32,25] on ontology translation has addressed both syntactic and semantic-issues, but tends to focus more on syntactic translation [25] because it is easier to automate. "Semantic" translation is more difficult because creating mapping rules often requires subtle judgments about the relationships between meanings of concepts in one ontology and their meanings in another. We assume that, at least for the foreseeable future, it can't be fully automated.[4]

We break ontology translation into three parts: syntactic translation from the source notation in a web language to an internal representation, semantic translation by inference using the internal notation, and syntactic translation from the internal representation to the target web language. All syntactic issues are dealt with in the first and third phases, using a translator, *PDDAML* [14] for translating between our internal representation and OWL. If a new web language becomes more popular for the Semantic Web, we only need extend PDDAML to

---

[4] The translation problem is certainly "AI-complete" in the sense that a program that solved it would have to be as intelligent as a person; but in fact it may be even harder than that, because agreeing on a translation in the most difficult cases might require bargaining between experts about what their notations really mean. This is not really the sort of problem a single program could solve, even in principle.

handle it (assuming it is no more expressive than first-order logic). This allows us to focus on semantic translation from one ontology to another.

Our internal representation language is *Web-PDDL* [39], a strongly typed first order logic language with Lisp-like syntax. It extends the Planning Domain Definition Language (PDDL) [37] with XML namespaces, multi-type inheritance and more flexible notations for axioms. Web-PDDL can be used to represent ontologies, datasets and queries. Here is an example, part of the yale_bib ontology written in Web-PDDL.

```
(define (domain yale_bib-ont)
    (:extends (uri "http://www.w3.org/2000/01/rdf-schema#" :prefix rdfs))
    (:types Publication - Obj
            Article Book Incollection Inproceedings - Publication
            Literal - @rdfs:Literal)
    (:predicates (author p - Publication a - Literal)
                    .....))
```

The `:extends` declaration expresses that this domain (i.e., ontology) is extended from one or more other ontologies identified by the URIs. To avoid symbol clashes, symbols imported from other ontologies are given prefixes, such as @rdfs:Literal. These correspond to XML namespaces, and when Web-PDDL is translated to RDF [39], that's exactly what they become. Types start with capital letters and are the same concept as classes in some other web languages, such as OWL. A type T_1 is declared to be of a subtype of a type T_0 by writing "T_1 - T_0" in the `:types` field of a domain definition. In other contexts, the hyphen notation is used to declare a constant or variable to be of a type T, by writing "x - T". Predicates correspond roughly to "properties" in OWL, but they can take any number of arguments. There are also functions, including Skolem functions and built-in functions such as `+` and `-` that can be evaluated when appropriate.

Assertions are written in the usual Lisp style: (author pub20 "Tom Jefferson"), for instance. We'll discuss quantifiers shortly.

Web-PDDL reflects a fundamentally different philosophy about knowledge-representation (KR) languages than that embodied in notations such as RDF and OWL. The latter reflect the strong opinions in the Description Logic community that a KR language should make it impossible to pose undecidable (or even intractable) problems. Our attitude is that languages should be as expressive as different reasoning applications will require. There are many interesting application areas where useful programs exist in spite of scary worst-case performance. As we hope to show below, ontology translation is a good example, where certain simple techniques from theorem proving solve a large portion of the problem, even though theorem proving is in principle undecidable.

2.2   Axiom-Based Ontology Merging

If all ontologies, datasets and queries can be expressed in terms of the same internal representation, such as Web-PDDL, semantic translation can be implemented as formal inference working with a *merged ontology* of the source and target ontologies. Ontology merging is the process of taking the union of the

concepts of source and target ontologies together and adding the *bridging axioms* to express the relationship (mappings) of the concepts in one ontology to the concepts in the other. Such axioms can express both simple and complicated semantic mappings between concepts of the source and target ontologies. The simple semantic mappings include "subClassOf," "subPropertyOf" or "equivalent" relationships. For example, if two types (class) are equivalent (sameClassAs), such as the Book type in the yale_bib ontology is equivalent to the Book type in the cmu_bib ontology. Because types are not objects, we cannot write an axiom such as $(= T_1\ T_2)$. So we have to use a pseudo-predicate (or, perhaps, "meta-predicate") `T->` and write bridging axioms about equivalent types. In the merged ontology of yale_bib and cmu_bib, the equivalent relationship about their Book types is written in Web-PDDL:

```
(:axioms
    (T-> @yale_bib:Book Book)
    (T-> @cmu_bib:Book Book)
    ...
```

Namespace prefixes distinguish yale_bib's Book and cmu_bib's Book. The symbols without a prefix are native to the merged ontology. Our axiom defines a new Book type in the merged ontology, and makes yale_bib's Book equivalent to cmu_bib's Book by making both of them be equivalent to the new defined Book in the merged ontology.

The reason we need a Book type in the merge is: the merge will be a totally new ontology which can be merged further with other ontologies. Suppose we have got the cyb ontology as the merge of the cmu_bib and yale_bib ontologies. There is another foo_bib ontology needs to be merged with the cyb ontology and the foo_bib ontology also has a Book type. Since we already have cyb's Book, we don't need to specify the relationship between foo_bib's Book and cmu_bib's Book, or the relationship between foo_bib's Book and yale_bib's Book. What we need to do is specify the relationship between foo_bib's Book and cyb's Book. Therefore, we need to define types and predicates in the merge, even through they are the same as one of two related types or predicates in the component ontologies.

The more complicated semantic mappings, such as the one about yale_bib's booktitle and cmu_bib's booktitle in Example 1.1.2, can be expressed as bridging axioms in the merged ontology. But we must be careful to distinguish the two senses of (booktitle *a s*), which in yale_bib means "Inproceedings or Incollection *a* appeared in a book with title *s*" and in cmu_bib means "The title of book *a* is *s*". Namespace prefixes suffice for the distinguishing two booktitles. The more interesting task is to relate the two senses, which we accomplish with the bridging axioms

```
(forall (a - Article tl - String)
        (iff (@yale_bib:booktitle a tl) (booktitle a tl)))

(forall (a - @yale_bib:Inproceedings tl - String)
        (iff (booktitle a tl)
                (exists (p - Proceedings)
```

```
            (and (contain p a)
                 (@cmu_bib:inProceedings a p)
                 (@cmu_bib:booktitle p tl)))))
```

Note that the bridging axioms can be used to go from either ontology to the other. The second axiom uses an existential quantifier and `p` is a existential quantified variable. It also can be written in the form of skolem functions after skolemization [45]:

```
(forall (a - @yale_bib:Inproceedings tl - String)
        (if (booktitle a tl)
            (and (contains (@skolem:aProc a tl) - Proceedings a)
                 (@cmu_bib:inProceedings a (@skolem:aProc a tl))
                 (@cmu_bib:booktitle (@skolem:aProc a tl) tl)))))
```

We use the prefix @skolem: as a convention for the skolem functions.

Some bridging axioms may need "callable" functions. For example, yale_bib's year predicate uses Number to represent the year when a publication was published. However, cmu_bib's year predicate uses String to represent the year. When we try to express the mapping between these two predicates, we select yale_bib's year predicate as the one in the merge. We have to use two functions, one for converting a number to a string and the other one for converting a string to a number, to express these mapping axioms:

```
(forall (p - Publication yn - @cmu_bib:Year)
        (if (@cmu_bib:year p yn)
            (year p (@built_in:NumbertoString yn))))

(forall (p - Publication y - String)
        (if (year p y)
            (@cmu_bib:year p (@built_in:StringtoNumber y))))
```

We use the prefix built_in to indicate that these two functions are built-in functions.

For the foreseeable future the construction of merged ontologies has to involve the efforts of human experts. If necessary, when the source and target ontologies are very large, automatic mapping tools can give some suggestions to human experts, but, in our view, before we know what bridging axioms look like, there's no point in spending a lot of effort on building automated tools.

2.3 OntoEngine: An Optimized Theorem Prover for Semantic Translation

Our decision to use a theorem prover for semantic translation may cause some concern, given that in general a theorem prover can run for a long time and conclude nothing useful. However, 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.

Our theorem prover, called *OntoEngine*, is specialized for these sorts of inference. OntoEngine 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 target ontology to 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.

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.[5] 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 [44]. To avoid generating an infinite number of answers, Datalog rules are required to satisfy some *safety* conditions [47]. Hence, there are not any existentially quantified variable in the head (conclusion) 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

---

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

equality substitution to avoid redundant answers so that it can handle such kind of complicated axioms.

Description logics [23] 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, they cannot generate new objects — only select subsets of existing objects. For example, the DL systems cannot generate skolem terms although description logics have existential quantifier. The DL rules (axioms) do not allow (built-in) functions which is necessary in some bridging axioms, such as year example in section 2.2. OntoEngine can generate skolem terms and process the axioms with (built-in) functions.

OntoEngine is not a complete first-order theorem prover, unlike resolution-based systems, such as Otter [48]. 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 Deductive Ontology Translation Between Datasets

In this section we describe how to apply our new approach to implement dataset translation. We set up an online ontology translation service, OntoMerge, to do deductive dataset translation on the Semantic Web. A more detailed account on the forward chaining algorithm for our generalized modus ponens reasoner appears in [28].

The problem for translating datasets can be expressed abstractly thus: given a set of facts in one vocabulary (the *source*), infer the largest possible set of consequences in another (the *target*). We break this process into two phases:

1. *Inference:* working in a *merged ontology* that combines all the symbols and axioms from both the source and target, draw inferences from source facts.
2. *Projection:* Retain conclusions that are expressed purely in the target vocabulary.

In *Example* 1.2.1, suppose the source ontology is yale_bib and the target ontology is cmu_bib. Considering the semantic difference mentioned in Example 1.1.2, the fact "The publication BretonZucker96 appeared in the Proceedings of IEEE Conf. on Computer Vision and Pattern Recognition" is expressed in the yale_bib ontology thus:

```
(:objects ... BretonZucker96 - InProceedings)
(:facts ... (booktitle BretonZucker96 "Proceedings of  CVPR'96"))
```

In the cmu_bib ontology, the same fact should be expressed thus:

```
(:objects ... BretonZucker96 - Article proc38 - Proceedings)
(facts ... (inProceedings BretonZucker96 proc38)
           (booktitle proc38 "Proceedings of  CVPR'96") ...)
```

Recall the bridging axioms related to this booktitle example:

```
(forall (a - Article tl - String)
        (iff (@yale_bib:booktitle a tl) (booktitle a tl)))

(forall (a - @yale_bib:Inproceedings tl - String)
        (iff (booktitle a tl)
             (exists (p - Proceedings)
                (and (contain p a)
                     (@cmu_bib:inProceedings a p)
                     (@cmu_bib:booktitle p tl)))))
```

When used from left to right, the bridging axioms causes the inference engine to introduce a new constant (proc38) to designate the proceedings that the article (BretonZucker96) appears in. Such *skolem terms* are necessary whenever the translation requires talking about an object that can't be identified with any existing object.

On the Semantic Web model, the knowledge is mostly represented in XML-based web languages. We have set up an online ontology-translation system called OntoMerge. OntoMerge serves as a semi-automated nexus for agents and humans to find ways of coping with notational differences, both syntactic and semantic, between ontologies. OntoMerge wraps OntoEngine with PDDAML, which implement the syntactic translation for the input and output DAML or OWL files. The architecture of OntoMerge for translating datasets is shown in Figure 1.
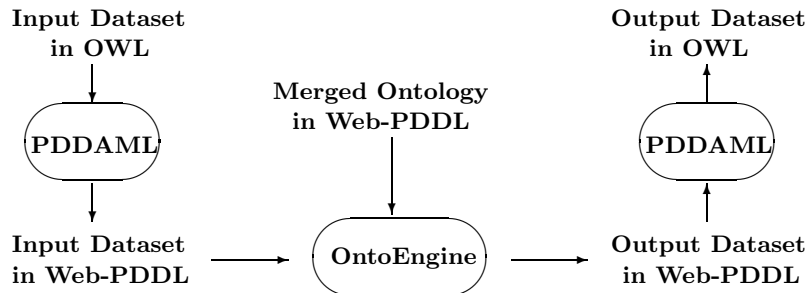

Fig. 1. The OntoMerge Architecture for Translating Datasets

When receiving an input dataset to translate, OntoEngine needs a merged ontology that covers the source and target ontologies. If no such merged ontology is available, all OntoEngine can do is to record the need for a new merger. (If

enough such requests come in, the ontology experts may wake up and get to work.) Assuming that a merged ontology exists, located typically at some URL, OntoEngine tries to load it in. Then it loads the dataset (facts) in and does forward chaining with the bridging axioms, until no new facts in the target ontology are generated.

OntoMerge has worked well so far, although our experience is inevitably limited by the demand for our services. In addition to the small example from the dataset[6] using the yale_bib ontology to the equivalent dataset using the cmu_bib ontology, we have also run it on some big ones.

Experiment 1: OntoMerge translates a dataset[7] with 7564 facts about the geography of Afghanistan using more than 10 ontologies into a dataset in the map ontology [10]. 4611 facts are related to the geographic features of Afghanistan described by the geonames ontology [9] and its airports described by the airport ontology [1]. Some facts about an airport of Afghanistan are:

```
(@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)
```

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: 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, an airport is a special location which is different from a bridge, and it's 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 like:

```
(@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)
```

---

[6] http://cs-www.cs.yale.edu/homes/dvm/daml/datasets/yale_bib_dataset.daml
[7] http://www.daml.org/2001/06/map/af-full.daml

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 Onto-Engine, 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[8] with 21164 facts (on 3010 individuals and 1422 families of European royalty) in the bbn_ged genealogy ontology [2] to 26956 facts in the drc_ged genealogy ontology [8]. 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 ontology 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. As our general understanding, 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")
```

Prospective users should check out the OntoMerge website[9]. We have put all URLs of existing merged ontologies there. OntoMerge is designed to solicit descriptions of ontology-translation problems, even when OntoMerge can't solve them. However, according to our experience, we believe that in most cases we

---

[8] http://www.daml.org/2001/01/gedcom/royal92.daml
[9] http://cs-www.cs.yale.edu/homes/dvm/daml/ontology-translation.html

can develop and debug a merged ontology within days that will translate any dataset from one of the ontologies in the merged set to another. It's not difficult for a researcher who knows first-order logic to write bridging axioms in Web-PDDL. We encourage other people to develop their own merged ontology to solve ontology translation problems they encounter.

4   Ontology Extension Generation

As we have said, manually developing sub-ontologies extended from existing ontology(s) is tedious at the Web scale. Tools are needed to make it easier because the number of sub-ontologies is usually much larger. In this section, we will introduce our approach to generate ontology extensions automatically by ontology translation.

One scenario is that ontology experts have some sub-ontologies of the existing ontology(s), and they want to generate the corresponding sub-ontologies of other related existing ontology(s). If they know the relationships between those existing ontologies, ontology-translation tools can automate this process. Another scenario is that ontology experts often need to update some existing ontologies when new knowledge or new requirement comes up. This work has to be done manually, but how about updating their sub-ontologies? Since they know the relationships between the old and updated ontologies, new sub-ontologies can be generated automatically.

In *Example* 1.2.2, if ontology experts can merge DAML-S and WSDL Schema first, they can translate Congo.com into its "grounding." The advantage is they only need to get one merged ontology for DAML-S and WSDL Schema. Further translation from the sub web service ontologies of DAML-S to their groundings on WSDL Schema can be implemented automatically.

The structure for OntoMerge to generate ontology extensions is similar to that shown in Figure 1. The difference is the input and output are not datasets but sub-ontologies. Instead of a set of facts, we input a set of sub-property definitions. In *Example* 1.2.2, the following sub-property occurs in the Congo.com ontology:

```
(deliveryAddress sp1 - SpecifyDeliveryDetails st2 - @xsd:string)
```

where SpecifyDeliveryDetails is a subtype of @DAML-S:Process. To find the corresponding sub-property of a WSDL property, we create an instance of deliveryAddress, with new skolem constants for the variables:

```
(deliveryAddress SDD-1 str-2)
;;SDD-1 and str-2 are skolem constants of types SpecifyDeliveryDetails
;;and @xsd:string respectively
```

Hypothetically assume that this is a true fact, and draw conclusions using forward chaining. This inference process uses the axioms in the Congo ontology, and the bridging axioms in the merged ontology for DAML-S and WSDL Schema such as:

```
(forall (ob1 ob2)
   (if (deliveryAddress ob1 ob2) (@process:input ob1 ob2)))
;;the above axiom is from the Congo ontology to express that
;;deliveryAddress is a sub property of @process:input in DAML-S.

(forall (x - @DAML-S:Process)
        (exists (sg - ServiceGrounding) (ground sg x)))

(forall (p - Process sg - ServiceGrounding ob1 - String)
    (if (and (ground sg p) (@process:input p ob1))
        (exists (ms - Message pa - Part pm - Param)
          (and (@wsdl:input p pm) (paramMessage pm ms)
               (part ms pa) (partElement pa ob1)))))
;;these two axioms are from merged ontology for DAML-S and WSDL Schema.
```

OntoEngine can generate the translated facts in Web-PDDL:

```
(@wsdl:input SDD-1 Param374)
(@wsdl:operation PortType367 SDD-1)
(@wsdl:partElement Part376 str-2)
(@wsdl:part Message375 Part376)
(@wsdl:paramMessage Param374 Message375)
```

where Param374 and such are further skolem terms produced by instantiating existential quantifiers during inference.

All of the conclusions are expressed in the WSDL Schema ontology. The first three mention the two skolem constants in the original assumption. These are plausible candidates for capturing the entire meaning of the deliveryAddress predicate as far as WSDL Schema is concerned. So to generate the new extension WSDL_congo, simply create new predicates for each of these conclusions and make them sub-properties of the predicates in the conclusions:

```
(define (domain WSDL_congo)
   (:extends (uri "http://schemas.xmlsoap.org/wsdl/"))
   (:types SpecifyDeliveryDetails - Operation ....)
   (:predicates
      (deliveryAddress_input arg1 - SpecifyDeliveryDetails arg2 - Param)
      (deliveryAddress_operation arg1 - PortType
                                 arg2 - SpecifyDeliveryDetails)
      (deliveryAddress_partElement arg1 - Part arg2 - @xsd:string)
   ...
```

The corresponding axioms for sub-property relationships are:

```
(forall (ob1 ob2) (if (deliveryAddress_input ob1 ob2)
                      (@wsdl:input ob1 ob2)))
(forall (ob1 ob2) (if (deliveryAddress_operation ob1 ob2)
                      (@wsdl:operation ob1 ob2)))
(forall (ob1 ob2) (if (deliveryAddress_partElement ob1 ob2)
                      (@wsdl:partElement ob1 ob2)))
```

The output sub-ontology is a grounding of Congo in WSDL Schema and it can
be represented in WSDL after feeding it into a translator between Web-PDDL
and WSDL. That translator has been embedded in PDDAML and the output
for the grounding of Congo in WSDL looks like:

```
<wsdl:message name="SpecifyDeliveryDetailsInputMsg">
   <wsdl:part name="deliveryAddressPart"
              element="xsd:string"/>
   ...
</wsdl:message>

<wsdl:portType name="SpecifyDeliveryDetails_PortType">
   <wsdl:operation name="SpecifyDeliveryDetails">
     <wsdl:input name="SpecifyDeliveryDetailsInput"
                 message="SpecifyDeliveryDetailsInputMsg"
     </wsdl:input>
   </wsdl:operation>
</wsdl:portType>
```

Our automatically generated WSDL_congo is very similar to the manually
produced grounding by the DAML-S group[10].

This result is encouraging, but obviously much remains to be done. The tech-
nique of treating skolemized definitions as pseudo-axioms can translate only ax-
ioms expressing sub-property relationships in the source sub-ontology. Although
this technique works fine for grounding Congo to WSDL, other ontology exten-
sion generation problems may need translating more general axioms from the
source sub-ontology to the target sub-ontology. We have developed an algorithm
called *automatic axiom derivation* to deal with it[11].

5   Querying Through Different Ontologies

Forward-chaining deduction is a data-driven inference technique that works well
for translating datasets and ontology-extension generation. We have also em-
bedded a more traditional backward-chaining reasoner into OntoEngine. This
module becomes the central component of an end-to-end workflow (similar to
that in figure 1) to translate queries expressed in the standard query language
DQL [7] to Web-PDDL, answer the queries using backward chaining, and trans-
late the results back as a DQL response. As usual, we will focus on the semantic
internals of this process, not the syntactic translations between Web-PDDL and
DQL.

To extend OntoMerge to handle querying problem through different ontolo-
gies, we embedded some tools for query selection and query reformulation. One
input query can be the conjunction of some sub-queries and each of them may

---

[10] http://www.daml.org/services/daml-s/0.7/CongoGrounding.wsdl

[11] The full solution about general axioms translation can be found in Dejing Dou's
Ph.D. Dissertation *Ontology Translation by Ontology Merging and Automated Rea-
soning* (Technical Report 1300, Yale Computer Science, August, 2004).

be answered by different knowledge bases. We might not be able to "translate" the whole input query in one ontology to the query in another. For example, suppose we add to the query of *Example* 1.2.3 a conjunct asking for the name of the woman Henry VI married (the @xsd prefix is for "XML Schema Datatype"):

```
(:query (freevars (?k ?q - Individual ?f - Family ?m - Marriage
                   ?n - @xsd:string ?d - @xsd:date)
        (and (@drc_ged:name ?k "Henry_VI") (@drc_ged:husband ?f ?k)
             (@drc_ged:wife ?f ?q) (@drc_ged:name ?q ?n)
             (@drc_ged:marriage ?f ?m) (@drc_ged:date ?m ?d))))
```

The required answer must give the bindings for variables ?d and ?n.

This query is expressed using the drc_ged ontology. Suppose an agent asks OntoMerge for help in answering it, and OntoMerge's library of merged ontologies includes some with drc_ged ontology as a component. This means that OntoMerge might be able to help answer the query with those web resources described by the other component ontologies of the merged one. In particular, suppose OntoMerge has a merged ontology for the drc_ged and bbn_ged ontologies. It can would ask some broker agent to find some web knowledge bases using the bbn_ged ontology. In this experiment, we just assume one such web knowledge base exists (and is trustworthy!).

The whole process is described as follows. OntoMerge calls the query selection tool to select one sub-query. Here, the tool will first select (@drc_ged:name ?k "Henry_VI") because it has only one variable. OntoEngine then does backward chaining for this sub-query and translates it into a query in the bbn_ged ontology, (@bbn_ged:name ?k "Henry_VI"). The new one is sent to the web knowledge base described by the bbn_ged ontology, which returns the binding {?k/@royal92:@I1217@}. (@royal92:@I1217@ is an Individual in the web knowledge base.) With this binding, OntoMerge call the query-reformulation tool to reform the rest of the sub-queries and get another selection: (@drc_ged:husband ?f @royal92:@I1217@). After backward chaining and querying, the next binding we get is {?f/ @royal92:@F448@}, which leads to a new sub-query

```
(and (@drc_ged:wife @royal92:@F448@ ?q)
     (@drc_ged:marriage @royal92:@F448@ ?m))
```

and its corresponding one in the bbn_ged ontology:

```
(and (@bbn_ged:sex ?q "F") (@bbn_ged:spouseIn ?q @royal92:@F448@)
     (@bbn_ged:marriage @royal92:@F448@ ?m))
```

The bindings this time are {?q/@royal92:@I1218@}, and {?m/@royal92:event3732}. Repeat the similar process and the final query in the bbn_ged ontology is

```
(and (@bbn_ged:name @royal92:@I1218@ ?n)
     (@bbn_ged:date @royal92:event3732 ?d))
```

The ultimate result is {?n/"Margaret of_Anjou"} and {?d/"22 APR 1445"}.

In addition, answering query by backward chaining may be necessary in the middle of forward chaining. For example, when OntoEngine is unifying the fact (P c1) with (P ?x) in the axiom:

$$(\text{P } ?x) \wedge (\text{member } ?x \ [c1, c2, c3]) \Rightarrow (\text{Q } ?x)$$

it can't conclude (Q c1) unless it can verify that c1 is a member of the list [c1,c2,c3], and the only way to implement this deduction is by answering that query by backward chaining.

6   Related Work

As we said at the beginning of this paper, one can consider our work to be the application of insights from query translation to the general ontology-translation problem. We have been able to draw on a long tradition of work in query translation for databases [42,44,20], and for web searching [21,41]. Much of this work (e.g., [30,20]) has tackled the *query optimization* problem, which we have not focused much on yet, although there are some query selection and reformulation tools in OntoMerge.

There has been some previous work on developing deductive-rule systems for the semantic web [33]. The emerging standard is OWL Rules [13], which can be characterized as an XML serialization of logic-programming rules. While we use heuristics similar to those embodied in logic programming, we believe that ontology translation requires equality substitution and a more systematic treatment of existential quantifiers than logic programming can provide. The dominant paradigm for the Semantic Web is description logics (DLs), and there has been work on reconciling the rule-based approach with DLs [31]. We would rather "layer" logic on top of RDF in a way that leaves it completely independent of the constraints of description logics [39].

Our ontology merging is rather different from what some other people have emphasized in talking about ontology combination, because traditionally merging two ontologies has meant finding mappings between them and using those to weed out redundancies. The PROMPT [43] and Chimaera [40] are two prominent examples. As we said in section 1, our interest is not in mapping, but in the content of the rules that mapping might help discover. The large literature on ontology mapping(e.g., [27]) is therefore only tangentially relevant to our work.

One approach to ontology translation is to create a global, centralized ontology that all other notations can be translated into [32,34]. The problem with this strategy is that it is difficult to keep the central ontology updated and in harmony with all sub-ontologies, especially since new sub-ontologies will appear every day. If someone creates a simple, lightweight ontology for a particular domain, he may be interested in translating it to neighboring domains, but can't be bothered to think about how it fits into a grand unified theory of knowledge representation.

The idea of building up merged ontologies incrementally, starting with local mergers, has been explored in a recent paper [19], in which bridging rules are

assumed to map database relations by permuting and projecting columns. These rules are simpler than ours, but in return the authors get some very interesting algorithms for combining local ontology mappings into more global views.

People working in specific domains have tried to use ontology techniques to help their own data integration tasks. For example, some eCommerce and Tourism researchers have proposed a way to do data translation between real-world (e.g. travel and tourism) ontologies [26].

7   Conclusions and Future Work

The distributed nature of the Web makes ontology translation one of the most difficult problems web-based must cope with. We described our new approach to implement ontology translation on the Semantic Web. Here are the main points we tried to make:

1. *Ontology translation* is required when translating datasets, generating ontology extensions, or querying through different ontologies. It must be distinguished from ontology mapping, which is the process of finding likely correspondences between symbols in two different ontologies. This sort of mapping can be a prelude to translation, but it is likely to be necessary for the foreseeable future for a human expert to produce useful translation rules from proposed correspondences.
2. Ontology translation can be thought of in terms of formal inference in a merged ontology. The merge of two related ontologies is obtained by taking the union of the terms and the axioms defining them, then adding bridging axioms that relate the terms in one ontology to those in the other through the terms in the merge.
3. If all ontologies, datasets and queries can be expressed in terms of the same internal representation, semantic translation can be implemented by automatic reasoning. We believe the reasoning required can be thought of as typed, first-order inference with equality substitution, easily implemented using a language such as Web-PDDL for expressing type relationships and axioms. The syntactic translation can be done by an automatic syntax translator between Web-PDDL and other Web agent languages.

We set up an online ontology translation server, OntoMerge, to apply and validate our method. We have evaluated our approach by the experiments for large web knowledge resources and its performance is good so far. We also discuss the efficiency and completeness of our inference system. We hope the existence of OntoMerge will get more people interested in the hard problem of generating useful translation rules.

Our results so far open up all sorts of avenues of further research, especially in the area of automating the production of bridging axioms. Although these can be quite complicated, many of them fall into standard classes. We are working on tools that allow domain experts to build most such axioms themselves, through a set of dialogues about the form of the relation between concepts in

one ontology and concepts in the other. We also will develop tools to check the consistency of the generated bridging axioms. The long-range goal is to allow domain experts to generate their own merged ontologies without being familiar with the technicalities of Web-PDDL.

Recently, we began cooperating with the medical informatics researchers of Yale to apply our approach to integrate different Web-based neuronal databases: Yale's SenseLab database and Cornell's CNDB database. Although both of their data and database schemas have been marked up by using some XML specifications, there are still some major differences between what the data of each database concerns. The differences exist because the database designers had different views and purposes: SenseLab's data is about model and structure information of a particular class of neurons but CNDB's is about experimental data for individual neurons measured at a particular day. These kind of differences make data integration very difficult. Based on OntoMerge structure, we are designing some initial tools to support construction and testing of axioms for merging two different database schemas. Our future work will focus on designing human computer interactive tools to help domain experts, such as neuroscientists, to find and build bridging axioms between the concepts from different ontologies or database schemas. The biggest obstacle is that domain experts may not be familiar with any formal logic languages but only know the knowledge of their domains. Therefore, this future work will involve automatic ontology mapping, bridging axiom production from machine learning and natural language processing, pattern reuse and consistency testing for merged ontologies.

Acknowledgements

References

1. http://www.daml.org/2001/10/html/airport-ont.daml.
2. http://www.daml.org/2001/01/gedcom/gedcom.daml.
3. http://www.daml.ri.cmu.edu/ont/homework/atlas-publications.daml.
4. http://www.daml.org/ontologies/
5. http://www.daml.org/services/.
6. http://www.ai.sri.com/daml/ontologies/time/Time.daml.
7. http://www.daml.org/2003/04/dql/.
8. http://orlando.drc.com/daml/Ontology/Genealogy/3.1/Gentology-ont.daml.
9. http://www.daml.org/2002/04/geonames/geonames-ont.daml.
10. http://www.daml.org/2001/06/map/map-ont.daml.
11. http://opencyc.sourceforge.net/daml/cyc.daml.
12. http://www.w3.org/TR/webont-req/.
13. http://www.daml.org/2003/11/swrl/.

14. http://www.cs.yale.edu/homes/dvm/daml/pddl_daml_translator.html.
15. http://www.w3c.org/TR/wsdl.
16. http://schemas.xmlsoap.org/wsdl/.
17. http://www.w3.org/TR/REC-xml-names/
18. http://www.cs.yale.edu/homes/dvm/daml/ontologies/daml/yale_bib.daml.
19. K. Aberer, P. Cudré-Mauroux, and M. Hauswirth. The chatty web: emergent semantics through gossiping. In *Proc. International World Wide Web Conference*, 2003.
20. S.Adali, K.Candan, Y.Papakonstantinou, and V. Subrahmanian. Query Caching and Optimization in Distributed Mediator Systems. In *Proc. ACM SIGMOD Conf. on Management of Data*, pages 137–148, 1996.
21. Yigal Arens, Craig A. Knoblock and Wei-Min Shen. Query reformulation for dynamic information integration. *J. Intelligent Information Systems — Special Issue on Intelligent Information Integration* **6**(2/3), pp. 99–130
22. D.-S. C. A. Ankolekar, M. Burstein, J. R. Hobbs, O. Lassila, D. Martin, D. McDermott, S. A. McIlraith, S. Narayanan, M. Paolucci, T. Payne, and K. Sycara. Daml-s: Web service description for the semantic web. In *Proceedings of International Semantic Web Conference 2002*, pages 348–363, 2002.
23. F. Baader, D. McGuinness, D. Nardi, and P. P. Schneider. *The Description Logic Handbook*. Cambridge University Press, 2002.
24. T.Berners-Lee, J.Hendler, and O.Lassila. The Semantic Web. *Scientific American*, 284(5):34–43, 2001.
25. H. Chalupsky. OntoMorph: A translation system for symbolic logic. In *Proc. Int'l. Con. on Principles of Knowledge Representation and Reasoning*, pages 471–482, San Francisco, 2000. Morgan Kaufmann.
26. M. Dell'Erba, O. Fodor, F. Ricci, and H. Werthner. Harmonise: A Solution for Data Interoperability. In *I3E 2002*, 2002.
27. A. Doan, J. Madhavan, P. Domingos, and A. Halevy. Learning to map between ontologies on the semantic web. In *Proceedings of the World-Wide Web Conference (WWW-2002)*, 2002.
28. D. Dou, D. McDermott, and P. Qi. Ontology Transaltion by Ontology Merging and Automated Reasoning. In *Proceedings of EKAW02 Workshop on Ontologies for Multi-Agent Systems*, 2002. Available at http://cs-www.cs.yale.edu/homes/dvm/papers/DouMcDermottQi02.ps
29. Dejing Dou, Drew McDermott, and Peishen Qi. Ontology Translation on the Semantic Web. In *Proceedings of International Conference on Ontologies, Databases and Application of SEmantics (ODBASE) 2003*, pages 952–969.
30. M. R. Genesereth, A. Keller, and O. Duschka. Infomaster: An information integration system. In *Proc 97 ACM SIGMOD International Conference on Management of Data*, pages 539–542, 1997.
31. B. N. Grosof, I. Horrocks, R. Volz, and S. Decker. Description logic programs: Combining logic programs with description logic. In *Proc. International World Wide Web Conference*, 2003.
32. T. Gruber. Ontolingua: A Translation Approach to Providing Portable Ontology Specifications. *Knowledge Acquisition*, 5(2):199–200, 1993.
33. Jeff Heflin and James Hendler Searching the web with SHOE. In *Artificial Intelligence for Web Search. Papers from the AAAI Workshop. WS-00-01*, pp. 35–40. Menlo Park, CA: AAAI Press. 2000
34. Douglas B. Lenat and R.V. Guha *Building Large Knowledge-Based Systems.* Reading: Addison-Wesley 1990

35. J. Madhavan, P. A. Bernstein, P. Domingos, and A. Halevy. Representing and Reasoning about Mappings between Domain Models. In *Proc. AAAI 2002*, 2002.
36. A. Maedche, B. Motik, N. Silva, and R. Volz. MAFRA - A Mapping Framework for Distributed Ontologies. In *Proceedings of the 13th International Conference, EKAW 2002.*
37. D. McDermott. The Planning Domain Definition Language Manual. Technical Report 1165, Yale Computer Science, 1998. (CVC Report 98-003).
38. D. McDermott, M. Burstein, and D. Smith. Overcoming ontology mismatches in transactions with self-describing agents. In *Proc. Semantic Web Working Symposium*, pages 285–302, 2001.
39. D. McDermott and D. Dou. Representing Disjunction and Quantifiers in Rdf. In *Proceedings of International Semantic Web Conference 2002*, pages 250–263, 2002.
40. D. L. McGuinness, R. Fikes, J. Rice, and S. Wilder. An Environment for Merging and Testing Large Ontologies. In *Proceedings of the Seventh International Conference on Principles of Knowledge Representation and Reasoning (KR2000)*, 2000.
41. E. Mena, A. Illarramandi, V. Kashyap, and A. Sheth OBSERVER: An approach for query processing in global information systems based on interoperation across pre-existing ontologies. In *Int. J. Distributed and Parallel Database (DAPD)* **8**(2), pp. 223–271
42. P. Mitra, G. Wiederhold, and M. Kersten. A graph-oriented model for articulation of ontology interdependencies. In *Proceedings of Conference on Extending Database Technology (EDBT 2000)*, 2000.
43. N. F. Noy and M. A. Musen. Prompt: Algorithm and tool for automated ontology merging and alignment. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI-2000)*, 2000.
44. Rachel Pottinger and Alon Levy. A scalable algorithm for answering queries using views. In *Proceedings of the 26th VLDB Conference*, 2000.
45. S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach.* Prentice-Hall, Inc, 1995.
46. L. Serafini, P. Bouquet, B. Magnini, and S. Zanobini. An algorithm for matching contextualized schemas via sat. In *Proceedings of CONTEXT'03.*
47. Abraham Silberschatz, Henry F. Korth, and S. Sudarshan. *Database System Concepts, 4th Edition.* McGraw-Hill Companies, 2001.
48. Larry Wos. *The Automation of Reasoning: An Experimenter's Notebook with Otter Tutorial.* Academic Press, 1996.