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 as means for formally specifying the vocabulary and relationship of concepts are seen playing a key role on the Semantic Web. However, the Web's distributed nature makes ontology translation one of the most difficult problems that web-based agents must cope with when they share information. Ontology translation is required when translating datasets, generating ontology extensions and querying through different ontologies. OntoMerge, an online system by ontology merging and automated reasoning, can implement ontology translation with inputs and outputs in DAML+OIL or other web languages. The merge of two related ontologies is obtained by taking the union of the concepts and the axioms defining them. We add *bridging axioms* not only as "bridges" between concepts in two related ontologies but also to make this merge into a new ontology for further merging with other ontologies. Our uniform internal representation, *Web-PDDL*, is a strong typed first-order logic language for web application, used to separate ontology translation into syntactic translation and semantic translation. Syntactic translation is done by an automatic translator between Web-PDDL and DAML+OIL 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 can process and "understand" data rather than merely display them as at present [21]. 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.

More and more ontologies are being developed and many of them describe similar domains. The distributed nature of the Web allows web-based agents to use different ontologies. In this section, we first describe the syntactic and semantic differences between ontologies on similar domains. We then define ontology translation problem in three categories: datasets translation, ontology extension

* This research was supported by DARPA as DAML program.

R. Meersman et al. (Eds.): CoopIS/DOA/ODBASE 2003, LNCS 2888, pp. 952–969, 2003.

generation and querying through different ontologies. We will also distinguish ontology translation from ontology mapping and talk about some previous related work.

We call our new approach *ontology translation by ontology merging and automated reasoning.* Our focus is on formal inference from facts expressed in one ontology to facts expressed in another. We will have little to say about eliminating syntactic differences, and instead will generally assume that the facts to be translated will be in the same logical notation after translation as before; only the vocabulary will change. The *merge* of two related ontologies is obtained by taking the union of the terms and the axioms defining them, using XML namespaces 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. Devising and maintaining a merged ontology must involve the contribution from 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, a theorem prover optimized for the ontology-translation task, is called *OntoEngine.* We use it for dataset translation (section 2), ontology-extension generation(section 3), and query handling through ontologies (section 4). We also will discuss related work and our future plans for about semi-automatic tools for ontology merging in section 5.

1.1 The Differences between Ontologies on Similar Domains

The differences between two ontologies on similar domains can include syntactic and semantic differences, both of which we must deal with. Although current web-agent languages, including DAML+OIL [1], OWL [9] and WSDL [7], all have XML encodings, they are basically different syntactically.

The semantic differences can be caused by many factors. One simple case is different taxonomic structures of concepts. For example, one genealogy ontology uses two properties - firstname and lastname - to represent a person's name but another genealogy ontology might use only one property fullname to represent it. The following gives another example of simple semantic differences, which is between two bibliography ontologies in the DAML ontology library [2]. One ontology was developed at Yale, and we give it the prefix yale_bib [14]; the other was developed at CMU and gets the prefix cmu_bib [13]. 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 includes all articles which were published in a journal, proceedings or collection. There are no Inproceedings and Incollection classes in the cmu_bib ontology.

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 it's range is String. It means 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 the same class as String. However, yale_bib's booktitle means that there is an Inproceedings or Incollection in some Proceedings or Collection and it is this Proceedings or Collection that has the string 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 [4] and the time ontology in OpenCyc [5]. Those time ontologies have semantic differences among their concepts, such as events. Some special events, such as birth, marriage and death in two genealogy ontologies can be created based on the event concepts of two different time ontologies. The semantic differences between the concepts of birth in different genealogy ontologies can be inherited from the semantic differences of events between the time ontologies.

1.2 Three Kinds of Ontology Translation Problems

Overcoming syntactic and semantic differences between ontologies is one of the most difficult problems that web-based agents must cope with. In general, we call it the *ontology translation problem*. It includes syntactic translation and semantic translation.

Ontology translation for datasets 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 [31]. 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 of professors in the CS department of Yale. And its datasets are written in DAML+OIL using the yale_bib ontology. Although the CMU agent might be able to handle the DAML+OIL syntax, it still can't completely process these datasets because of the semantic differences between the two ontologies. The CMU agent needs an ontology translation service to translate those datasets into the cmu_bib ontology first.

We also found ontology translation is required when generating ontology extensions and querying through different ontologies. The problem of *ontology extension generation* is defined thus: given two related ontologies O_1 and O_2 and an extension (subontology) O_{1s} of O_1, construct the "corresponding" extension O_{2s}. The ontology experts are developing more and more similar subontologies extended from existing ontology(s) manually. This work is tedious at the Web scale and we need some tools to make it easier.

EXAMPLE 1.2.2. DAML-S [3] is a general ontology describing web services in the application level and WSDL Schema [8] is another general ontology describing web services in the communication level about messages and protocols. Ontology experts have manually developed some subontologies extended from DAML-S, such as a book seller web service called Congo and an air ticket reservation web service called BravoAir. To make a web service really work, we need to describe it on the communication level. In other words, we need the corresponding subontologies extended from WSDL Schema for Congo and BravoAir. This process is also called "grounding" in [20]. It is possible to get the grounding of Congo or BravoAir by "translating" Congo or BravoAir from DAML-S to the corresponding subontology of WSDL Schema.

In our view, the most obvious feature of querying on the Semantic Web is: the knowledge to be used to answer a query may be in multiple knowledge bases, and these knowledge bases may describe their content using different ontologies from the ontology that the querying agent uses. On the Semantic Web model, the querying agent doesn't need to specify which knowledge base(s) can answer it's query, it also doesn't need to know what ontologies the answering knowledge base(s) uses. Without ontology translation, querying across these knowledge bases with different ontologies is very difficult.

EXAMPLE 1.2.3. Suppose a web agent wants to find the marriage date of Henry_VI, once a King of England. It need not know which knowledge base can answer its question, and can construct a query using the concepts in the drc_ged [15] genealogy ontology. One web knowledge base storing the information about the individuals and families of European royalty should be able to answer this question, but it uses a different bbn_ged [16] genealogy ontology and only can answer a query in that ontology. The two genealogy ontologies surely will have some semantic differences between them. Ontology translation is required for the agent to get its query answered.

1.3 Ontology Translation Is Different from 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. The mappings should be expressed by some mapping rules which explain how those concepts correspond. Obviously, ontology translation needs to know the mappings of two ontologies first, then it can use the mapping rules. The mappings are generated either by ontology experts or by some automatic tools. Automating the process of ontology mapping is an active area of research [33,34, 24,29]. In our view, automatic ontology mapping can save time and give suggestions to ontology experts. But without ontology experts' involvements [31], an ontology translation system can not directly use the mapping rules generated by automatic mapping tools if it doesn't expect any wrong translation. There are mainly two reasons. First, automatic mapping tool can't generate 100% accurate mappings and the result need ontology experts' manually correction. Second, although existing automatic mapping tools can express simple semantic difference

between two concepts using "subclassOf," "subpropertyOf" or "equivalent" relationships, they are not very useful for complicated semantic differences because only ontology experts can figure them out. Ontology experts need a more expressive formal representation language to express the complicated mapping rules. In this paper, we usually presuppose that there is a way to find the correspondence of two ontologies with the help of ontology experts and some automatic mapping tools. We will focus on how to express complicated mapping rules and how to implement ontology translation itself.

1.4 Previous Work

Previous work on ontology translation for datasets has made use of two strategies. One is to translate a dataset in any source ontology to a dataset in one big, centralized ontology that serves as an interlingua which can be translated into a dataset in any target ontology. Ontolingua [28] is a typical example for this strategy, but this strategy can't really work well unless a global ontology can cover all existing ontologies, and we can get agreement by all ontology experts to write translators between their own ontologies and this global ontology. Even if in principle such harmony can be attained, in practice keeping all ontologies – including the new ones that come along every day – consistent with the One True Theory is very difficult. 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 other strategy is to do ontology translation directly from a dataset in a (source) ontology to a dataset in another (target) ontology, on a dataset-by-dataset basis, without the use of any kind of interlingua. OntoMorph [23] is a typical example of this strategy. For practical purposes this sort of program can be very useful, but it tends to rely on special properties of the datasets to be translated, and doesn't address the question of producing a general-purpose translator that handles any source dataset.

Previous work on ontology translation for query handling is closely related to database mediators [34]. This kind of work always more focuses on the different taxonomic structures of ontologies because the features of databases. In the Semantic Model, the difference between the ontologies describing web knowledge resources will be more complicated.

2 Deductive Ontology Translation between Datasets

In this section we briefly summarize our new approach for ontology translation, and how OntoMerge translates datasets on the Semantic Web. A more detailed account on the forward chaining algorithm for our generalized modus ponens reasoner appears in [25].

2.1 Separate Syntactic and Semantic Translation

Past work [28,23] on ontology translation has addressed both syntactic and semantic-issues, but tends to focus more on syntactic translation [23] 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 another. It can't be fully automated.

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* [17] for translating between our internal representation and DAML+OIL [1]. If a new web language becomes more popular for the Semantic Web, we only need extend PDDAML to handle it (e.g. PDDAML also can handle OWL now). This allows us to focus on semantic translation from one ontology to another.

Our internal representation language is *Web-PDDL* [32], a strongly typed first order logic language with Lisp-like syntax. It extends the Planning Domain Definition Language (PDDL) [30] with XML namespaces 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 [32], that's exactly what they become. Types start with capital letters. A constant or variable is declared to be of a type T by writing "x - T". Assertions are written in the usual Lisp style: (author pub20 "Tom Jefferson"), for instance. Compared with other web languages, such as DAML+OIL, Web-PDDL can express more complicated axioms about the relationships between the concepts of different ontologies, such as the axioms with functions. Some example bridging axioms in Web-PDDL are in the following sections.

2.2 Ontology Merging and Automated Reasoning

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.

For the foreseeable future the merged ontology has to be constructed by human experts. If necessary, when the source and target ontologies are very large, automatic mapping tools can give some suggestions to human experts. As we have said, the merged ontology contains all the symbols and facts from both the source and target ontologies, but in addition it must contain *bridging axioms* that relate symbols in one ontology to symbols in the other. The merged ontology itself is a new ontology and it can be used for further merging with other ontologies.

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") ...)
```

In the merged ontology, we must be careful to distinguish the two senses of (booktitle a s), which in the source means "Publication a appeared in a book with title s" and in the target means "The title of book a is s". Namespace prefixes suffice for that job. 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)))))
```

The symbols without a prefix are native to the merged ontology. Note that the bridging axioms can be used to go from either ontology to the other.

The second axiom uses an existential quantifier. When used from left to right, the rule causes the inference engine to introduce a new constant (proc38) to designate the proceedings that the article (BretonZucker96) appears in. Such *skolem terms* [36] are necessary whenever the translation requires talking about an object that can't be identified with any existing object.

We also introduced term-generating functions into the merged ontologies. These functions can give finer controls over term generation than skolemization would give and improve the accuracy of the inference. We use the prefix @control as a convention our inference engine requires for the term-generating functions. So the second axiom can be rewritten as:

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

Our decision to use theorem proving for 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 source to target ontologies.
- Backward chaining from queries in one ontology to datasets in another.
- Introduction of skolem terms and term-generating functions as explained above.
- Use of equalities to substitute existing constant terms for skolem terms.

Our theorem prover, called "OntoEngine", is specialized for these sorts of inference. To avoid infinite loops, we set a limit to the complexity of terms that OntoEngine generates; and, of course, the deductive engine stops when it reaches conclusions (or, in the case of backward chaining, goals) in the target ontology. In addition, OntoEngine has a good type checking system making use of strong typed feature of Web-PDDL.

Instead of full-fledged resolution, OntoEngine uses chaining through implications with specified directions. That means it is not *complete* in the logical sense; we trade completeness for efficiency. In any case, the kind of completeness that might seem appropriate for ontology translation is that anything that can be expressed in the source ontology can be translated into the target ontology; call this *translation completeness*. Even a logically complete theorem prover would in general fail to achieve translation completeness because the source ontology and target ontology might not totally overlap.

2.3 OntoMerge and Experiments for Translating Datasets

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 files. The architecture of OntoMerge for translating datasets is shown in Figure 1.

When receiving an input dataset to translate, OntoEngine needs a merged ontology that covers the source and target ontologies. If no such merged ontology

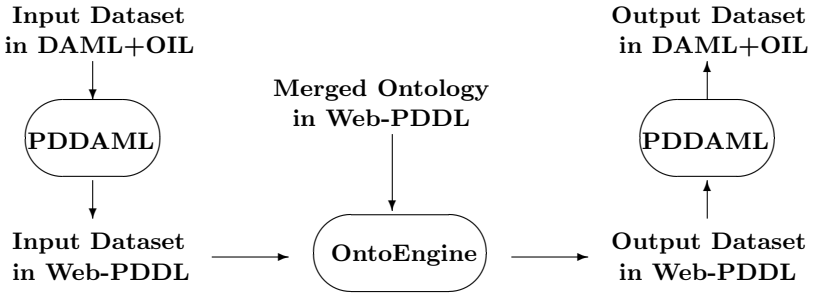

Fig. 1. The OntoMerge Architecture for Translating Datasets

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[1] 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[2] with 7564 facts about the geography of Afghanistan using more than 10 ontologies into a dataset in the map ontology [11]. 4611 facts are related to the geographic features of Afghanistan described by the geonames ontology [12] and its airports described by the airport ontology [10]. 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 given below:

```
(forall (x - Airport y z  - Object)
    (if (and (@airport:latitude x y) (@airport:longitude x z))
```

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

```
(and (location (@control:aPoint x) - Point
               (@control:aLocation x) - Location)
     (latitude (@control:aLocation x) y)
     (longitude (@control:aLocation x) 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)
```

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[3] with 21164 facts (on 3010 individuals and 1422 families of European royalty) in the bbn_ged genealogy ontology [16] to 26956 facts in the drc_ged genealogy ontology [15]. 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 but 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

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

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[4]. 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 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 has first order logic background to write bridging axioms in Web-PDDL by themselves. We encourage other people to develop their own merged ontology to solve ontology translation problems they encounter.

3 Ontology Extension Generation

As we have said, manually developing subontologies extended from existing ontology(s) is tedious at the Web scale. Tools are needed to make it easier because the number of subontologies 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 subontologies of the existing ontology(s), and they want to generate the corresponding subontologies of other related existing ontology(s). If they know the relationships between those existing ontologies, some 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 subontologies? Since they know the relationships between the old and updated ontologies, new subontologies can be generated automatically.

In *Example* 1.2.2, if ontology experts can merge DAML-S and WSDL Schema first, they can translate Congo or BravoAir into their groundings. 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 subontologies. For *Example* 1.2.2, we have experimented with the following idea, which works in simple cases: Take a property in the Congo ontology:

[4] http://cs-www.cs.yale.edu/homes/dvm/daml/ontology-translation.html

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

where SpecifyDeliveryDetails is a subtype of @DAML-S:Process. Create an instance of it, with anonymous skolem constants for the variable:

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

Hypothetically assume that this fact is true, and draw conclusions using forward chaining. This inference process use facts like these form and the axioms in Congo ontology and the bridging axioms in the merged ontology for DAML-S and WSDL Schema like:

```
(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 subproperties 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 subproperty 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 subontology 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.

Our automatically generated WSDL_congo is very similar to the manually produced grounding by the DAML-S group[5]. It is encouraging but this particular technique has taken us only so far. The main gap is that it can translate the types, predicates and only those axioms about subproperties of one ontology extension O_{1s} to corresponding of O_{2s}. It works well for some sub-ontologies, such as Congo and BravoAir, because most axioms in them are those about subproperties. But we can expect there are more general axioms in other sub-ontologies, how to derive general axioms from one subontology to the other by inference will be one of our future work.

4 Querying through Different Ontologies

Forward-chaining deduction is a data-driven inference and it works well for translating datasets and ontology extension generation. We also embed a backward chaining reasoner into OntoEngine and get it run in demand-driven way. To test our backward chaining reasoner, we can extend OntoMerge to handle querying problem through different ontologies, as we have mentioned in *Example* 1.2.3.

There are some query languages for the Semantic Web. One of them is DQL [6]. We suppose some web-based agents use DQL and we have extended our PDDAML to handle syntax translation between DQL query and Web-PDDL.

To extend OntoMerge to handle querying problem through different ontologies, we embedded some tools for query selection and query reformulation. One input query can be the conjunction of some subqueries and each of them may 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. We have experimented with the following idea for *Example* 1.2.3. And besides that question, the query agent also wants to know the name of the king's wife who married him on that date. It constructs the query in DQL query triples using the drc_ged ontology. Due to the limit of space, we only give out the corresponding query in Web-PDDL. The @xsd prefix is for XML Schema Datatype.

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

```
(: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))))
```

It is the conjunction of some subqueries in Web-PDDL. The required answer must give the bindings for variables ?d and ?n.

In the Semantic Web model, if that agent asks help from OntoMerge, it might have tried some web knowledge bases using the drc_ged ontology but got no answer. Now the ontology translation is necessary. When activated, OntoMerge will search its library of merged ontologies to see if any merged ontology includes the drc_ged ontology as its component. If yes, it means OntoMerge might be able to help answer the query with those web resources described by the other component ontologies of the merged one. So far, there is a merged ontology for the drc_ged and bbn_ged ontologies in the library of OntoMerge . So OntoMerge 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 it can answer queries described by the bbn_ged ontology.

The whole process is described as follows. OntoMerge calls the query selection tool to select one subquery. Here, the tool will first select (@drc_ged:name ?k "Henry_VI") because it only has one variable. OntoEngine then does backward chaining for this subquery and "translate" 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 subqueries 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 subquery

```
(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"}.

This example is quite simple because the bindings all happen to come from one knowledge base. We just use it to test our backward chaining reasoner in OntoEngine.

A full treatment of answering query by backward chaining across ontologies would raise the issue of *query optimization*, which we have not focused

much on yet, although there are some query selection and reformulation tools in OntoMerge. There is a lot of work in this area, and we will cite just two references: [26,19]. We intend to more focus on overcoming the complicated semantic differences when querying across different ontologies.

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:

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

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.

5 Related Work

So far, our discussion has focused more on how to express the semantic differences between two ontologies in a merged ontology, and how to implement ontology translation by inference. Although we think the process of ontology merging needs human experts' involvements and can't be fully automated for the foreseeable future, it will be helpful to develop some semi-automatic tools for ontology merging.

Our ontology merging is rather different from what some other people have emphasized in talking about ontology combination because we focus more on bridging axioms for inference. The PROMPT [35] and Chimaera [33] systems focus on ontology editing for merging two similar ontologies. They try to do ontology matching semi-automatically according to name similarity and taxonomic structure. The matching provides user with some suggestions for further refinement. Some recent work, such as GLUE [24], has used machine learning and exploit information in the data instances to generate mapping rules of two ontologies. GLUE still only generates simple mapping rules about "subclassOf," "superclassOf," and "equivalent" relationships. Ontology experts can check the accuracy of these simple mapping rules and write the remaining, more complicated, mapping rules by themselves.

We are not the only ones who have realized that deductive rules are an important component of inference and translation systems. The emerging standard is RuleML [22], 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. A recent paper [27] on the relation between rules and description logics attempts to restrict rules even further. Our approach is to "layer" logic on top of RDF in a way that leaves it completely independent of the constraints of description logics [32].

The idea of building up merged ontologies incrementally, starting with local mergers, has been explored in a recent paper [18], 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.

6 Conclusions

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 *ontology merging*. 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 simple typed, first-order inference, easily implemented using a language such as Web-PDDL for expressing type relationships and axioms.

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. These long-range goal is to allow domain experts to generate their own merged ontologies without being familiar with the technicalities of Web-PDDL.

References

1. http://www.daml.org/2001/03/daml+oil-index.html.
2. http://www.daml.org/ontologies/
3. http://www.daml.org/services/.
4. http://www.ai.sri.com/daml/ontologies/time/Time.daml.
5. http://opencyc.sourceforge.net/daml/cyc.daml.
6. http://www.daml.org/2003/04/dql/.
7. http://www.w3c.org/TR/wsdl.
8. http://schemas.xmlsoap.org/wsdl/.
9. http://www.w3.org/TR/webont-req/.
10. http://www.daml.org/2001/10/html/airport-ont.daml.
11. http://www.daml.org/2001/06/map/map-ont.daml.
12. http://www.daml.org/2002/04/geonames/geonames-ont.daml.
13. http://www.daml.ri.cmu.edu/ont/homework/atlas-publications.daml.
14. http://www.cs.yale.edu/homes/dvm/daml/ontologies/daml/yale_bib.daml.
15. http://orlando.drc.com/daml/Ontology/Genealogy/3.1/Gentology-ont.daml.
16. http://www.daml.org/2001/01/gedcom/gedcom.daml
17. http://www.cs.yale.edu/homes/dvm/daml/pddl_daml_translator.html.
18. K. Aberer, P. Cudré-Mauroux, and M. Hauswirth. The chatty web: emergent semantics through gossiping. In *Proc. International World Wide Web Conference*, 2003.
19. 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.
20. 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.
21. T.Berners-Lee, J.Hendler, and O.Lassila. The Semantic Web. *Scientific American*, 284(5):34–43, 2001.
22. H. Boley, B. Grosof, M. Sintek, S. Tabet, and G. Wagner. RuleML Design, September 2002. http://www.dfki.uni-kl.de/ruleml/indesign.html.
23. 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.
24. 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.
25. 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
26. 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.
27. 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.

28. T. Gruber. Ontolingua: A Translation Approach to Providing Portable Ontology Specifications. *Knowledge Acquisition*, 5(2):199–200, 1993.
29. J. Madhavan, P. A. Bernstein, P. Domingos, and A. Halevy. Representing and Reasoning about Mappings between Domain Models. In *Proc. AAAI 2002*, 2002.
30. D. McDermott. The Planning Domain Definition Language Manual. Technical Report 1165, Yale Computer Science, 1998. (CVC Report 98-003).
31. 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.
32. D. McDermott and D. Dou. Representing Disjunction and Quantifiers in Rdf. In *Proceedings of International Semantic Web Conference 2002*, pages 250–263, 2002.
33. 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.
34. 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.
35. 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.
36. S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Inc, 1995.