Reconciling Inconsistencies Between Package-extended Ontology Modules

Jie Bao
Iowa State University

Vasant Honavar
Iowa State University

Follow this and additional works at: http://lib.dr.iastate.edu/cs_techreports

Part of the Artificial Intelligence and Robotics Commons

Recommended Citation
http://lib.dr.iastate.edu/cs_techreports/242

This Article is brought to you for free and open access by the Computer Science at Digital Repository @ Iowa State University. It has been accepted for inclusion in Computer Science Technical Reports by an authorized administrator of Digital Repository @ Iowa State University. For more information, please contact digirep@iastate.edu.
Reconciling Inconsistencies Between Package-extended Ontology Modules

Jie Bao, and Vasant Honavar
Artificial Intelligence Research Laboratory
Computer Science Department
Iowa State University
Ames, IA USA 50010
Email: \{baojie, honavar\}@cs.iastate.edu

December 28, 2005

Abstract

Construction of ontologies in specific domains (e.g., molecular biology, electronic commerce), is invariably a collaborative activity that requires incorporation of independently generated ontology fragments or ontology modules, and hence reconciliation of inconsistencies among ontology modules. We investigate an approach to reconciling the inconsistencies among ontologies using defeasible axioms. In this framework, each ontology module can be viewed as an internally consistent unit (called a package) with well-specified access interfaces. Multiple ontology modules or packages can be combined to obtain larger ontologies. Inconsistencies between ontology modules are handled using defeasible axioms (an axiom in one package can defeat one or more axioms from other packages), thereby making the resulting composite ontology internally consistent. The resulting framework supports collaborative ontology construction as well as integration of preexisting ontologies.

1 Introduction

Ontologies that capture assumptions about objects and relationships among objects in specific domains of interest are essential enablers of effective use of independently developed distributed data and knowledge sources, software components and services in applications that span virtually every area of human activity. Construction of ontologies in specific domains (e.g., molecular biology, electronic commerce), is invariably a collaborative activity that involves direct cooperation among multiple domain experts or ontologists or indirect cooperation among ontologists through the reuse of previously published, autonomously developed ontology fragments or ontology modules. Because different experts typically have only partial knowledge of the domain, and because ontologies
are developed with a specific set of applications or a user community in mind, large-scale ontology building efforts inevitably require integration of multiple independently developed ontology fragments or ontology modules to generate an ontology that offers an adequate coverage of the domain of interest.

Integration of independently developed ontologies is complicated by the inevitable semantic differences and logical inconsistencies between independently developed ontology modules. Inconsistencies among ontology fragments may arise for several reasons: different experts may have different point of view on the same issue (e.g., some view Cantonese as a dialect of Chinese, while others view it as an independent language); new information about a phenomenon may be inconsistent with old beliefs (e.g., the old belief that the earth is flat is inconsistent with the modern belief that the earth is nearly spherical). Thus, there is an urgent need for collaborative ontology building environments that enable participants to collaboratively build ontologies by reusing, adapting, and integrating, independently generated ontology modules that offer partial coverage of a domain of interest in the presence of inconsistencies among the ontology modules. This paper investigates an approach to address this problem. The rest of this paper is organized as follows: Section 2 introduces the basic idea of package-extended description logics to support collaborative ontology building by multiple experts; Section 3 investigates an approach to reconciling inconsistent ontology packages using defeasible axioms; Section 4 concludes with a summary of the paper and discussion of related work.

2 Package-extended Description Logics

2.1 Description Logic

Ontologies are typically described using ontology languages, such as DAML+OIL or OWL. Description logic [2] is used to express the formal semantics of an ontology written in such ontology languages. A description logic (DL) consists of a Tbox and an Abox, where the Tbox is a finite set of terminological axioms such as $C \sqsubseteq D$, and the Abox is a finite set of assertional statements such as $C(a)$ or $R(a, b)$. Thus, for example, a simple ontology about animals can be represented in DL as follows:

- $Dog \sqsubseteq Carnivore$
- $Carnivore \sqsubseteq Animal \sqcap \forall eats.Animal$
- $Dog(billy)$

The ontology asserts that a $Dog$ is a $Carnivore$; a $Carnivore$ is an $Animal$ that only $eats$ $Animal$; and that $billy$ is a $Dog$.

The popular ontology language OWL is based on the description logic $SHI\tilde{O}Q(D)$, of which a complete axiom constructors list can be found in [10]. The Tableau algorithm [2] is a widely used reasoning algorithm for DL. For simplicity, we restrict the ontologies considered in this paper to those that can be described in
\(SH\Omega Q(D)\), a subset of \(SH\Omega Q(D)\) for which efficient tableaux-based reasoning algorithms are available [11].

2.2 Package-Extended Ontology

As noted above, collaborative ontology construction requires support for modular ontologies. Current ontology languages like OWL, while they offer some degree of modularization by restricting ontology segments into separate XML namespaces, fail to fully support localized semantics for ontology modules and partial ontology reuse. Package based ontology language extensions [5], offer a way to overcome these limitations. The resulting ontology language allows representation of ontology modules using components called packages. Each package typically consists of a set of highly related terms and their relations; packages can be nested in other packages, forming a package hierarchy; the visibility of a term is controlled by scope limitation modifiers such as public and private. Although each package constitutes an internally consistent ontology, there is no requirement that an arbitrary set of packages be globally consistent.

The whole ontology is composed of a set of packages.

Table 1: Part of Notations of Package-extended Ontology

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Syntax</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Package</td>
<td>(P)</td>
<td>(P^I \in \Delta_P)</td>
</tr>
<tr>
<td>Membership</td>
<td>(t \in P) or (member(t, P))</td>
<td>(member^I \subseteq \Delta_S \times \Delta_P)</td>
</tr>
<tr>
<td>Nesting</td>
<td>(N \in \Delta_P)</td>
<td>(N^I \in \Delta_P \times \Delta_P)</td>
</tr>
</tbody>
</table>

\(\Delta_S\) is the ontology term domain (i.e., the set of all possible term names), and \(\Delta_P\) is the domain of all possible packages.

In this paper, we restrict ourselves to a simplified version of package-extended ontology with no scope limitations and package nesting. (See Table 1). For example, the animal ontology may contain knowledge about dog in two different packages:

package(1)

Terms: Dog, Carnivore, Animal, eats, billy \(\in\) package(1)

Axioms:
(1a) \(1 : Dog \sqsubseteq 1 : Carnivore\)
(1b) \(1 : Carnivore \sqsubseteq 1 : Animal\)
(1c) \(1 : Carnivore \sqsubseteq \forall 1 : eats.1 : Animal\)
(1d) \(1 : Dog(1 : billy)\)

package(2)

Terms: PetDog, Pet \(\in\) package(2)

Axioms:
We say package 2 imports package 1 since a term defined in package 1 is referred to in package 2. Here, the ontology term domain \( \Delta_S \) is \{Dog, Carnivore, Animal, eats, billy, Pet, PetDog\} and the package domain \( \Delta_P \) is \{package(1), package(2)\}

Package 2 extends the ontology in package 1 with assertions that a dog may also be a Pet and that billy is a PetDog.

A package-extended Description Logic ontology, or a P-DL ontology consists of multiple packages, each of them expressed in DL. The semantics of P-DL are as follows:

**Definition 1** A local interpretation of a package \( P \) is a pair \( I = \langle \Delta^I, (\cdot)^I \rangle \) where the concept space \( \Delta^I \) contains a nonempty set of objects and the role space \( (\cdot)^I \) is a function that maps each class \( c \in P \) to \( c^I \subseteq \Delta^I \); each property \( p \in P \) to \( p^I \subseteq \Delta^I \times \Delta^I \), and each instance \( i \in P \) to \( i^I \in \Delta^I \).

**Definition 2** A distributed interpretation of a set of packages \( \{P_i\} \), \( i = 1, \ldots, m \) is a family \( \mathcal{I}_d = \{I_i\} \), where \( I_i = \langle \Delta^I_i, (\cdot)^I_i \rangle \) is the local interpretation of \( P_i \). The distributed concept space is \( \Delta^{I_d} = \bigcup_{i=1}^{m} \Delta^I_i \) and the distributed role space \( (\cdot)^{I_d} \) maps each class \( c \in \Delta^{I_d} \subseteq \Delta^{I_1} \); each property \( p \) to \( p^{I_d} \subseteq \Delta^{I_1} \times \Delta^{I_1} \) (for some, possibly equal, \( i, j \)), and each instance \( i \) to \( i^{I_d} \in \Delta^{I_1} \) (for some \( j \)).

For example, a possible interpretation for the animal ontology is given in Fig. 1, where foo is the name of some Animal.

Figure 1: Local and Global Interpretation of the Animal Ontology
(a) \( I_1 \) is a local interpretation of \( P_{animal} \);
(b) \( I_2 \) is a local interpretation of \( P_{pet} \);
(c) \( I_2 \) is a global interpretation of the ontology consisting of the two packages.

Reasoning in package-extended ontology can be seen as distributed reasoning among autonomous ontology modules and the reasoning process can be built from local reasoning on individual modules [5].
3 Reconciling Inconsistent Ontology Modules

As noted earlier, semantic mismatches and possible logical inconsistencies between independently developed ontology modules make the combination of such modules into larger ontologies a challenging task. Specifically, in the case of two ontology modules $\alpha, \beta$, it is possible that although $\alpha \models t$, the module resulting from combining $\alpha$ and $\beta$ may not entail $t$, i.e., $\{\alpha, \beta\} \not\models t$. The following example illustrates this problem: A Dog is Carnivore; however, a PetDog sometimes eats DogFood which is CannedFood and not an Animal. Consider a variant of package 2 with the following axioms added:

(2d) 2 : DogFood $\sqsubseteq 2 : CannedFood$
(2e) 2 : CannedFood $\sqsubseteq \neg 1 : Animal$
(2f) 2 : PetDog $\sqsubseteq \exists 1 : eats.2 : DogFood$

The ontology that is obtained by simply merging package 1 and this variant of package 2 is inconsistent because a PetDog (which is a Dog and Carnivore) can eat DogFood (which contradicts the assertion that Carnivore only eats Animal). To construct an interpretation of this ontology, we have to introduce at least one instance, e.g., foo2, which is DogFood but not an Animal, and billy eats foo2 (because of (2f)); however, according to (1c), foo2 has to be an Animal, thus leading to a contradiction.

Hence, any system for collaborative ontology building has to provide mechanisms for detecting and reconciling such sources of inconsistency. Several techniques have been developed to reconcile inconsistencies within an ontology, including default logic [3,4] and defeasible logic [8]. In this paper, we extend our package-based ontology with the defeasible axioms of $OSHQ(D)$ proposed in [8].

3.1 Basic Observations

If $t$ is an axiom and we check the entailment $\Sigma \models t$ with an inconsistent knowledge base (KB) $\Sigma$, then the answer is always true. Also, if we check $\Sigma \models \neg t$ with inconsistent KB, the answer is again true. However, a consistent KB would answer false.

One natural way to handle such inconsistencies is to reduce the KB to a smaller, but consistent subset, with minimal loss of knowledge. For example, in the legislation process, if an article in a specific law contradicts the articles in the constitution, instead of nullifying the whole body of the law, people would only revoke the specific contradictory article and make the remaining part of the law consistent with the constitution.

We observe that a modular KB usually exhibits properties such as:

- Each module can be assumed to be internally consistent.
- In general, only small fragments of the knowledge in different modules are contradictory. When some of them are removed, the remaining modular KB is consistent.
• When different modules stand for different perspectives on the domain, the creators of modules may disagree on what is ‘right’ and ‘wrong’, therefore having different opinions on how to conciliate the inconsistencies.

• In a well-organized knowledge base, there is often, if not always, unidirectional ‘defeating’ between modules. For example, in an ontology of law, the rules in the constitution always take precedence over a specific law; when the constitution is revised, amendments always take precedence over the original articles.

We can exploit such observations to reconcile independently developed ontology modules. This section introduces the formal concept of defeasible modular ontology and provides an algorithm to reduce such an inconsistent ontology into a consistent subset.

3.2 Defeasible axioms

An axiom is said to be defeasible if some other axiom could defeat (or override) it.

Definition 3 A defeasible P-DL (DP-DL) knowledge base is a tuple \( \langle P_1, <_1, \ldots, P_n, <_n \rangle \), where \( P_i \) is a SHOQ(D) knowledge base and \( <_i \) is a strict priority order between axioms of packages. For each pair \( a_1 <_i a_2 \), \( a_2 \) is said to be defeasible by \( a_1 \) w.r.t. \( <_i \), while \( a_1 \) is a (possible) defeater of \( a_2 \) w.r.t. \( <_i \).

For example, the simple combined ontology of packages 1 and 2 is inconsistent on (1c) and (2f). However, with an appropriate priority order \( < \) among the relevant axioms, this logical inconsistency can be eliminated. One such possible priority order, reflecting the point of view of package 2, is (2f)\(<\) (1c) (read as (2f) is stronger than (1c) or (1c) is weaker than (2f)), since (2f) can be seen as the refinement of a more general axiom (1c). In this case, a specific axiom 2f defeats the general rule 1c. When there is a logical conflict between a pair of axioms, the weaker of the two conflicting axioms is ignored.

Note that each package may assert its own axiom defeating priority order, instead of assuming a single global order. For example, the creator of package 1 may give priority to its local knowledge and assert (1c)\(<\) (2f).

The notion of defeat is formalized as follows [8]:

Definition 4 Let \( \Sigma = \langle P_1, <_1, \ldots, P_n, <_n \rangle \) be a DP-DL knowledge base, and \( I = \langle \Delta^T, (.)^T \rangle \) a distributed model of a consistent subset of \( \Sigma \). A TBox axiom \( A \sqsubseteq B \in P_i \) is
- classically satisfied w.r.t. \( x \in \Delta^T \) iff \( x \in A^T \Rightarrow x \in B^T \)
- defeated w.r.t. \( x \in \Delta^T \) iff \( \exists(C \sqsubseteq D) <_i (A \sqsubseteq B) \) such that \( C \sqsubseteq D \) is satisfied w.r.t. \( x \) and \( x \in A^T \). In this case, we say that \( C \sqsubseteq D \) defeats \( A \sqsubseteq B \) w.r.t. \( x \) and \( <_i \). □
Although definition 4 is defined on TBox (terminological axiom) only, it’s easy to simulate the ABox with TBox axioms:

\[
C(a) \iff \{a\} \subseteq C \\
R(a, b) \iff \{a\} \subseteq \exists R.\{b\}
\]

For example, in Fig. 2, \(billy^I \in PetDog^I\) and \(billy^I \in Carnivore^I\); (2f) defeats (1c) since there is \(eats(billy, foo_2)\), therefore \(billy^I \in (\exists eats.DogFood)^I\), and (2f)<(1c).

![Figure 2: Part of an interpretation where (2f) defeats (1c) w.r.t. billy](image)

The specification of the priority order \(<\) for resolving inconsistencies between independently developed ontologies is best left to the user interested in combining the ontologies in question. It may be based on principles of the sort described in [1]: If the source of one axiom may be more reliable than the source of another axiom, the former one may have higher priority; A more recent axiom may be preferred over an earlier one; exceptions are stronger than the general rules.

In collaborative ontology building scenarios, it is reasonable to assign higher priority to local package axioms over axioms from imported packages since a local package can be seen as an extension or an exception to a general ontology.

Other priority order assignment policies can also be applied, such as based on the social order of the package authors.

### 3.3 Preferred Model

Because independently developed ontology modules may be inconsistent, a collaborative ontology building environment has to support identification of a preferred consistent subset of a given set of ontology modules. We will choose a preferred P-DL model based on the specified priority order \(<\), when we construct the interpretation (a.k.a., model) by tableau expansion. For example, for the Animal ontology and the priority order (2f)<(1c), given the fact that \(billy\) is a PetDog, we start searching the possible models from the local models of each package.

In package 1, without (1c), the original ABox is \(Dog(billy)\). It is expanded to \(I_1 = \{Dog(billy), Carnivore(billy), Animal(billy)\}\). If we apply axiom (1c), we get \(I_2 = I_1 \cup \{(\forall eats. Animal)(billy)\}\).
In package 2, without (2f), we get \( J_1 = \{ \text{PetDog}(billy), \text{Dog}(billy), \text{Animal}(billy), \text{Pet}(billy), \text{DogFood}(foo), \text{CannedFood}(foo), \neg \text{Animal}(foo) \} \). If (2f) is added, we get \( J_2 = J_1 \cup \{ \exists \text{eats} \text{.DogFood}(billy) \}, \text{eats}(billy, foo) \} \). Both \( I_2 \) and \( J_2 \) are locally consistent.

The consistent combination of them is summarized in Table 2 (the combination of \( I_2, J_2 \) is not included because it contains both \( \neg \text{Animal}(foo) \) and \( \text{Animal}(foo) \)). The preferred model is \( A_{12} \) because it defeats only the weakest axioms (1c).

<table>
<thead>
<tr>
<th>Model</th>
<th>Dog</th>
<th>PetDog</th>
<th>( \exists \text{eats. DogFood} )</th>
<th>Yeats, Animal</th>
<th>...</th>
<th>Defeats</th>
</tr>
</thead>
<tbody>
<tr>
<td>( A_{11} \leftarrow I_1, J_1 )</td>
<td>billy</td>
<td>billy</td>
<td>( \emptyset )</td>
<td>( \emptyset )</td>
<td>...</td>
<td>(1c), (2f)</td>
</tr>
<tr>
<td>( A_{12} \leftarrow I_1, J_2 )</td>
<td>billy</td>
<td>billy</td>
<td>billy</td>
<td>( \emptyset )</td>
<td>...</td>
<td>(1c)</td>
</tr>
<tr>
<td>( A_{21} \leftarrow I_2, J_1 )</td>
<td>billy</td>
<td>( \emptyset )</td>
<td>billy</td>
<td>...</td>
<td>(2f)</td>
<td></td>
</tr>
</tbody>
</table>

We denote a consistent subset of \( \Sigma = \{ P_1, <_1, ..., P_n, <_n \} \) as \( \pi(\Sigma) \). Formally, a preferred model is defined as follows.

**Definition 5** The support for a distributed model \( \mathcal{I} \) of \( \pi(\Sigma) \) is the set \( \mathcal{S}^\mathcal{I} = \{ (x, a) | x \in \Delta^\mathcal{I}, \text{axiom } a \in P_i \text{ is satisfied w.r.t. } x \text{ and } \mathcal{I} \} \) ( [8] Definition 3)

**Definition 6** A distributed model \( \mathcal{I} \) of \( \pi(\Sigma) \) is preferred over another distributed model \( \mathcal{J} \) of \( \pi'(\Sigma) \), denoted \( \mathcal{I} \preceq \mathcal{J} \), if \( \forall (x, a) \in \mathcal{S}^\mathcal{I}, \exists (x', b) \in \mathcal{S}^\mathcal{J}, b < a, \) or \( b = a. \)

The intuition behind this definition is that the model that defeats the fewest and the least preferred axioms is to be preferred. For example, \( \mathcal{S}^{A_{12}} = \{ \text{billy}, (1a), \text{billy}, (1b), \text{billy}, (2b), \text{billy}, (2a), \text{billy}, (2f), \text{foo}, (2e) \}, (\text{foo}, (2d)) \}, \mathcal{S}^{A_{21}} = \{ \text{billy}, (1a), \text{billy}, (1b), \text{billy}, (1c), \text{billy}, (2b), (\text{billy}, (2a)), (\text{foo}, (2e)), (\text{foo}, (2d)) \} \). 

\( A_{12} \) is preferred over \( A_{21} \) since for \( \text{(billy, (1c))} \in \mathcal{S}^{A_{21}} \), there is \( \text{(billy, (2f))} \in \mathcal{S}^{A_{12}} \) and \( (2f) < (1c) \). For any other axiom \( a, (\text{billy, a}) \in \mathcal{S}^{A_{21}} \), there is the same support in \( \mathcal{S}^{A_{21}} \).

### 3.4 Constructing a Consistent Subontology

The search for preferred models using a distributed tableau expansion algorithm of the sort outlined above incurs a high computational cost in the case of large ontologies, because of the combinatorial explosion of defeated axioms. However, based on the observations mentioned earlier, we can reduce the computational cost of searching with assumptions such as:

- **Local consistency**: There should be no defeating inside a package, since every package is assumed to be internally consistent.
- Unidirectionality: When the defeating is unidirectional among a package pair, stronger axioms are all in one package, therefore the search can be directed.

- Loss of least knowledge: If two subsets of the ontology, $\pi_1(O)$ and $\pi_2(O)$, are consistent, but $\pi_1(O)$ is a subset of $\pi_2(O)$, then $\pi_2(O)$ is used, as it contains more knowledge.

- The stronger is better: Given two consistent ontology subsets $\pi_1(O)$ and $\pi_2(O)$ that are not subset of each other, if for any axiom in $\pi_1(O)$ there exists always a possible defeater in $\pi_2(O)$, then $\pi_2(O)$ is preferred.

The assumptions above lead to some natural rules that, when given a set of (possibly mutually inconsistent ontology packages), aid in the search for a preferred model that corresponds to a consistent ontology.

- If there is no chain defeating, i.e. $a_1 < a_2$ and $\not\exists a_0, a_3$, such that $a_0 < a_1, a_2 < a_3$, then discard the weaker axiom $a_2$.

- If there is chain defeating, always keep the strongest axioms. For example, for $a_1 < a_2 < a_3 < a_4$, we discard $a_2, a_4$, but not $a_2, a_3$ or $a_1, a_3$.

We denote the reduced version of (not necessarily modular) ontology $O$ under axiom priority order $<$ with such defeating rules as $\pi_<(O)$.

**Theorem 1** For an ontology $O$ and defeating order $<$, if a model of $\pi_<(O)$ is found, then it is preferred to any model of another consistent subset of $O$, different from $\pi_<(O)$.

If the package-extended ontology has unidirectional defeating among packages and no cyclic defeating, we can also search for a consistent subset with the following algorithm.

- Construct the package defeating graph, a package $p_1$ is defeater of another package $p_2$ if an axiom in $p_1$ is stronger than an axiom in $p_2$.

- Starting from any package, keep its axioms with no defeaters, defeat its weaker axioms in neighbor packages; if no other package is its defeater, remove it from the defeating graph.

- Repeat the process until no axiom can be defeated.

**Theorem 2** Consider a DP-DL $\Sigma = (P_1, <_1, ...$. We have the following: for any $<_i$, if a distributed model of $\pi_{<_i}(O)$ is found, then this model is preferred to any distributed model of another consistent subset of $O$, different from $\pi_{<_i}(O)$.

The intuition is a model of $\pi_{<_i}(O)$ always contains a support that can’t be found in models of other sub ontologies.
4 Summary and Discussion

Ontology construction in complex real-world domains e.g., molecular biology, electronic commerce, is invariably a collaborative activity that requires incorporation of independently generated ontology modules, and hence, reconciliation of inconsistencies among ontology modules. In this paper, we have described how a consistent ontology can be extracted from a set of independently developed, internally consistent, but possibly mutually inconsistent ontology modules. Our approach exploits user-supplied priority order among axioms.

Nonmonotonic reasoning in description logic has received some attention in the literature. Badder et al. [4] introduced defaults in the description logic. Quanza et al. [13] studied preferred models and split axioms into defeasible and not defeasible axioms. Heymans et al. [9] extended defeasible reasoning to description logic, with a priority order defined on axioms, “stronger” axioms can defeat “weaker” axioms. Our approach further extends the non-monotonic DL to the distributed setting, and provides a simplified algorithm to find consistent subset of package-extended ontology. The idea of reducing an inconsistent ontology into a small consistent subset is also presented in [7, 12].

Borgida et al. [6] extended the description logic to obtain a distributed description logic (DDL) system. A DDL system consists of a set of distributed TBoxes and ABoxes connected by “bridge rules”. Serafini et al. [14] defines a sound and complete distributed tableau-based reasoning procedure which is built as an extension to standard Description Logic tableau. They introduce the concept ‘hole’ in the reasoning process for possible inconsistencies between packages. However, if a module has ‘hole’ model, the semantics in the whole module is sacrificed. Our approach only deny part of axioms in a module and use the remaining to construct a consistent ontology.

Our work in progress is aimed at more extensive investigation of the theoretical issues that arise in collaborative ontology construction as well as development of user-friendly software tools for collaborative ontology construction.

Acknowledgments

This research is supported by grants from the NSF (0219699) and the NIH (GM 066387) to Vasant Honavar

References


