Collaborative Ontology Building with Wiki@nt
- A multi-agent based ontology building environment

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

***Abstract*— Collaborative ontology building requires both knowledge integration and knowledge reconciliation. Wiki@nt is an ontology building environment that supports collaborative ontology development. Wiki@nt is based on $\mathcal{OSHOQP}$(D), an extension to $\mathcal{SHOQ}$(D) with $\mathcal{O}$ (partial order on axioms) and $\mathcal{P}$ (localized axioms in package) constructors. Wiki@nt supports integration and reconciliation of multiple independently developed, semantically heterogeneous, and very likely inconsistent ontology modules. A web browser based editor interface is provided, with features to support team work, version control, page locking, and navigation.**

Version: July 30, 2004

I. Introduction

A. Ontology Editing is a Knowledge Integration Process

Semantic Web [BLHL01] aims to support seamless and flexible access, use of semantically heterogeneous, networked data, knowledge, and services. The success of the semantic web enterprise relies on the availability of a large collection of domain or application specific ontologies and mappings between ontologies ontologies to allow integration of data [CSC+03]. However, by its very nature, ontology construction is a collaborative process which involves direct cooperation among individuals or groups of domain experts, knowledge engineers or/and software agents, or indirect cooperation through reuse or adaptation of previously published, autonomously developed ontologies.

In such settings, typically, different participants have only partial knowledge of the domain, and hence can contribute only partial ontologies of the domain. A common task involves refinement of a predefined ontology. Another common task involves integration of several such partial ontologies to obtain a coherent ontology that covers a much larger portion of the domain. Semantic mismatches and logical inconsistencies between independently developed ontologies are unavoidable. Thus, there is an urgent need for principled approaches and flexible tools for allowing individuals to collaboratively build, refine, and integrate existing ontologies as needed in specific contexts or for specific applications e.g., data-driven knowledge acquisition from semantically heterogeneous, distributed data sources [CSC+03], [Car04].

B. Motivating Examples: Pop Music Ontology

Suppose we want to build an ontology about pop music called PopOnt. John is a teenager who knows a great deal about pop music. He'd like to share his knowledge with other pop music enthusiasts. Like John, there are thousands of pop music fans who have knowledge relevant for characterizing the domain of pop music. Suppose some of the pop music of them may willing to spend a few minutes each day to write down a few simple assertions like "`M. Jackson isn't a country music artist`" and review, discuss, and propose changes to assertions made by others in their community.

There are also some information channels, such as mailing lists, newsgroups, weblogs, p2p applications and websites about pop music that can continually provide information about pop music. Some simple assertions could be validated across that information. For example, if "M. Jackson" hardly cooccurs with "country music", it's more likely "`M. Jackson isn't a country music artist`" is true. The goal then is to develop a tool that can be used by virtually anyone to participate in the construction of PopOnt.

C. Proposed Approach

While there has been a great deal of work on ontology languages, inference mechanisms, as well as ontology editing environments, relatively little attention has been paid to the development of principled approaches and tools for collaborative ontology building. Existing ontology editing and discovery tools are mostly focused on stand-alone ontology development rather than collaborative construction of ontologies. In this paper, we propose @nthill, a general architecture of an ontology editing, ontology refinement, and ontology integration environment. @nthill exploits $\mathcal{OSHOQP}$(**D**), a modular ontology representation language with preference partial order on axioms; **Wiki@nt**, a light-weight, browser-based ontology editor which requires minimal user effort and allows concurrent editing and integration of ontologies.

The rest of the paper is organized as follows: Section II describes ontology language features needed to support modular ontology design and ontology reconciliation; Section III gives the architecture of Wiki@nt; Section IV gives brief discussion of related work; Section V concludes with a summary and some directions for future work.

II. Collaborative Ontology Building as Knowledge Integration and Reconciliation

We start with a brief discussion of the theoretical basis of Wiki@nt including logical foundations of ontology languages. We then introduce a modular representation of ontologies and discuss some problems in reasoning with modular ontologies as they relate to the tasks of ontology integration and reconciliation.

A. Description Logic as a Knowledge Representation Language

Ontologies are typically described using ontology languages nowadays, such as DAML+OIL [Hor02] or OWL [SD04]. Description logic [BN03] is used to express the formal semantics of an ontology written in such ontology languages. A description logic 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 $R(a, b)$.

In particular, the description logic $\mathcal{SHIQ}$ is the formal background model for OWL. However, this correspondence is incomplete, since some important features of OWL, such as *concrete data types* and *named individuals*, are not supported by $\mathcal{SHIQ}$. $\mathcal{SHIOQ}$(**D**), an extension of the influential $\mathcal{SHOQ}$(**D**) description logic, attempts overcome these two limitations, by allowing data types (**D**) and named individuals($\mathcal{O}$) and is a proper DL model for OWL. However, ontology languages with $\mathcal{I}$ (i.e. inverse roles) constructor, such as $\mathcal{SHIOQ}$(**D**), suffers from complexity and/or intractability problems when they are used to represent and reason with ontologies when combined with combined with $\mathcal{O}$ or (**D**). Hence, in this paper, we use $\mathcal{SHOQ}(\mathbf{D})$ as the basis for a collaborative ontology development environment.

We assume that we have an abstract domain $\triangle^{\mathcal{I}}$, and a set of data types D and associate with each $d \in D$, a set $d^D \subseteq \triangle_D$ where $\triangle_D$ is the domain of all types. Table I summarizes the constructors that can be used to form complex concept expressions in $\mathcal{SHIOQ}$(**D**). A complete list of $\mathcal{SHIOQ}$(**D**), eg. OWL DL axiom constructors can be found in [HPSvH03].

An example Animal Ontology is given here:

```
SubClassOf( Dog , Carnivore )
SubClassOf( Dog , Pet )
SubClassOf( Carnivore, Animal)
  restriction(eats allValueFrom(Animal))
ObjectProperty( eats) domain( Animal )
individual ( billy type(Dog))
```

B. Package-Extended Ontology

Collaborative ontology building demands modularized ontology representation by its very nature .Current ontology languages like DAML+OIL and OWL, while they offer some degree of modularization by restricting ontology segments into separate XML namespaces, fail to fully support localized semantics, ontology evolution, distinction between semantic and organizational hierarchies over concepts and properties, ontology reuse, and knowledge hiding. In our previous work [BH04], we have argued for package based ontology language extensions to overcome these limitations. In the resulting ontology language P-OWL, a **package** is an ontology module with clearly defined access interface. Mapping between packages is performed by **views** which define a set of queries on the referred packages. Semantics are localized by hiding semantic details of a package by defining appropriate **interfaces** (special views). Packages provide an attractive way to compromise between the need for knowledge sharing and the need for knowledge hiding in collaborative design and use of ontologies. The structured organization of ontology entities (classes, properties, instances) in packages bring to ontology design and reuse, the same benefits as those provided by packages in software design and reuse in software engineering.

Some feature of the P-OWL language [BH04] include:

- A syntax specification that yields an OWL/RDF compatible language for package-extended ontologies.
- Package-extended ontologies to support localized semantics, flexible knowledge hiding as well as knowledge sharing. Ontology entities are defined with "Scope Limitation Modifier" that restrict their accessibility, and organized in module called "package".
- A mechanism for view-based information integration over modular ontologies with localized semantics. View is a set of queries over one or more ontology packages. Connecting ontologies with views could hiding semantic details of a package to the outside, as while as also be beneficent to flexible reuse of existing ontology.
- A distributed reasoning algorithm over a package-extended ontology to support locally-consistent reasoning across autonomous ontology modules even in scenarios

TABLE I

Part of syntax and semantics of $\mathcal{SHIOQ}$(**D**) expressions

Constructor	Syntax	Semantics
atomic concept C	A	$A^{\mathcal{I}} \subseteq \triangle^{\mathcal{I}}$
abstract role R_A	R	$R^{\mathcal{I}} \subseteq \triangle^{\mathcal{I}} \times \triangle^{\mathcal{I}}$
concrete role R_D	T	$T^{\mathcal{I}} \subseteq \triangle^{\mathcal{I}} \times \triangle_D$
nominal I	$\{o\}$	$\{o\}^{\mathcal{I}} \subseteq \triangle^{\mathcal{I}}, \#o^{\mathcal{I}} = 1$
data types D	d	$d^D \subseteq \triangle_D$
	$\neg d$	$(\neg d)^D = \triangle_D \backslash d^D$
conjunction	$C \sqcap D$	$(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \sqcap D^{\mathcal{I}}$
disjunction	$C \sqcup D$	$(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \sqcup D^{\mathcal{I}}$
negation	$\neg C$	$(\neg C)^D = \triangle^{\mathcal{I}} \backslash C^{\mathcal{I}}$
subclass	$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
exists res.	$\exists R.C$	$(\exists R.C)^{\mathcal{I}} = \{x \mid \exists y : (x, y) \in R^{\mathcal{I}} and\ y \in C^{\mathcal{I}}\}$
value res.	$\forall R.C$	$(\exists R.C)^{\mathcal{I}} = \{x \mid \forall y : (x, y) \in R^{\mathcal{I}} \Rightarrow y \in C^{\mathcal{I}}\}$
atleast res.	$\geq nR.C$	$(\geq nR.C)^{\mathcal{I}} = \{x \mid \#\{y \mid (x, y) \in R^{\mathcal{I}} and y \in C^{\mathcal{I}}\} \geq n\}$
atmost res.	$\leq nR.C$	$(\leq nR.C)^{\mathcal{I}} = \{x \mid \#\{y \mid (x, y) \in R^{\mathcal{I}} and y \in C^{\mathcal{I}}\} \leq n\}$
datatype exists	$\exists T.d$	$(\exists T.d)^{\mathcal{I}} = \{x \mid \exists y : (x, y) \in T^{\mathcal{I}} and\ y \in d^D\}$
datatype value	$\forall T.d$	$(\forall T.d)^{\mathcal{I}} = \{x \mid \forall y : (x, y) \in T^{\mathcal{I}} \Rightarrow y \in d^D\}$
inverse role	R^-	$(R^-)^{\mathcal{I}} = (R^{\mathcal{I}})^-$

TABLE II
SYNTAX AND SEMANTICS OF $\mathcal{SHOQP}(\mathbf{D})$

Constructor	Syntax	Semantics
Package	p	$p^P \in \triangle_P$
View	v	$v^{\mathcal{I}} \in \triangle_P$
Global Pkg	p_0	$p_0 \in \triangle_P$
InPackage	R_P	$R_P^{\mathcal{I}} \subseteq \triangle_T^{\mathcal{I}} \times \triangle_P$
HomePackage	$HP(t)$	$HP(t)^{\mathcal{I}} = \{p \mid (t^{\mathcal{I}}, p) \in R_P^{\mathcal{I}}\}$
NestedIn	$\in_N$	$\in_N^{\mathcal{I}} \in \triangle_P \times \triangle_P$
		$\in_N^{\mathcal{I}} = (\in_N^{\mathcal{I}})^+$
SLM	SLM(t, p)	$p \in \triangle_P$ can access $t \in \triangle_T^{\mathcal{I}}$iff SLM(t,p)=true
	public(t, p)	$\forall p, public(t, p) = true$
	private(t, p)	$\forall p, private(t, p) :=$ $(p = HP(t))$
	protected(t, p)	$\forall p, protected(t, p) :=$ $(p = HP(t) or\ p \in_N HP(t))$
Import	$im(P_1, P_2)$	P_2 is imported into P_1

in which common global semantics is unavailable. The reasoning process is built upon local reasoning offered by individual modules.

Table II gives the syntax and semantics of constructors in P-OWL. Let P be the set of all packages. We define $\triangle_P$ as the domain of P. We assume that the domain of interpretation of all packages $\triangle_P$ is disjoint from the concrete datatype domain $\triangle_D$, the abstract concept domain $\triangle^{\mathcal{I}}$, the abstract role domain $\triangle^{\mathcal{I}} \times \triangle^{\mathcal{I}}$ and concrete role domain $\triangle^{\mathcal{I}} \times \triangle_D$. We define the term domain $\triangle_T^{\mathcal{I}}$ as $\triangle_T^{\mathcal{I}} = \triangle_P \cup \triangle^{\mathcal{I}} \cup (\triangle^{\mathcal{I}} \times \triangle^{\mathcal{I}}) \cup (\triangle^{\mathcal{I}} \times \triangle_D)$. The resulting *package-extended* description logic language is called $\mathcal{SHOQP}(\mathbf{D})$ where $\mathcal{P}$ stands for "package-extended".

C. Ontology Reconciliation

As noted earlier, semantic mismatches and possible logical inconsistencies between independently developed ontology modules make the combining of such modules into larger ontologies a challenging task. Specifically, in the case of two ontology modules α, β, it is possible that although $\alpha \models t$, the module resulting from combining α and β may not entail t i.e., $\{\alpha, \beta\} \not\models t$ That is, any system for collaborative ontology building has to provide mechanisms for handling *nonmonotonicity*.

An example (adapted from [HV02]) illustrates this problem. A dog is carnivore; however, a sick dog sometimes eats grass. Formally, we add new axioms to the Animal Ontology:

```
DisjointClasses(Plant,Animal)
SubClassOf( SickDog, Dog)
   restriction(eats someValueFrom(Plant))
```

The resulting knowledge base will be inconsistent because a sick dog (which is a dog) now can eat grass (which contradicts the assertion that dogs are carnivores). Several techniques have been developed to reconcile inconsistent ontology system, such as default logic [BH93] [BH92] and defeasible logic [SH02] [HV02]. In this paper, we extend our P-OWL with the $\mathcal{OSHOQ}(\mathbf{D})$ proposed in [SH02]. An axiom is said to be *defeasible* if some other axiom could *defeat* (or override) it. The resulting ontology language is called $\mathcal{OSHOQP}(\mathbf{D})$ where $\mathcal{O}$ denotes a strict partial order on the axioms.

Definition 1: A $\mathcal{OSHOQP}(\mathbf{D})$-knowledge base is a tuple $\langle \mathcal{T}, < \rangle$, where $\mathcal{T}$ is a $\mathcal{SHOQP}(\mathbf{D})$-knowledge base and $<$ is a strict partial order between axioms of $\mathcal{T}$. For each pair $a_1 < a_2$, a_2 is said to be **defeasible** while a_1 is a (possible) **defeater** of a_2

Definition 2: A **local interpretation of a package** P_i is a pair $\mathcal{I}_i =< \triangle^{\mathcal{I}_i}, (.)^{\mathcal{I}_i} >$, where the concept space $\triangle^{\mathcal{I}_i}$ contains a nonempty set of objects and the role space $(.)^{\mathcal{I}}$ is a function over $\triangle^{\mathcal{I}_i} \times \triangle^{\mathcal{I}_i}$ such that for class c, $InPackage(c, P_i) \iff c^{\mathcal{I}_i} \subseteq \triangle^{\mathcal{I}_i}$; property p, $InPackage(p, P_i) \iff p^{\mathcal{I}_i} \subseteq \triangle^{\mathcal{I}_i} \times (\triangle^{\mathcal{I}_i} \cup d^D)$; instance i, $InPackage(i, P_i) \iff i^{\mathcal{I}_i} \in \triangle^{\mathcal{I}_i}$.

Definition 3: A **distributed interpretation** of a set of packages $\{P_i\}$, $i = 1, \cdots m$ is a family $\mathcal{I}_d = \{\mathcal{I}_i\}$ where $\mathcal{I}_i =< \triangle^{\mathcal{I}_i}, (.)^{\mathcal{I}_i} >$ is the local interpretation of P_i. The union of all $\triangle^{\mathcal{I}_i}$ is the distributed concept space $\triangle^{\mathcal{I}_d}$ and $(.)^{\mathcal{I}_d}$ ={functions over $\triangle^{\mathcal{I}_d} \times \triangle^{\mathcal{I}_d}$ } is the distributed role space.

The notion of defeat is formalized in the following definition (adapted from [SH02])

Definition 4: Let $\Sigma = \langle \mathcal{T}, < \rangle$ be an $\mathcal{OSHOQP}(\mathbf{D})$-knowledge base, and $I =< \triangle^{\mathcal{I}}, (.)^{\mathcal{I}} >$ an interpretation of Σ, A terminological axiom $A \sqsubseteq B \in \mathcal{T}$ is
- **applicable** w.r.t $x \in \triangle^{\mathcal{I}}$ and $\mathcal{I}$ iff $x \in A^{\mathcal{I}}$
- **applied** w.r.t $x \in \triangle^{\mathcal{I}}$ iff it is applicable w.r.t. x and $x \in B^{\mathcal{I}}$
- **[classically] satisfied** w.r.t $x \in \triangle^{\mathcal{I}}$ iff it is applied w.r.t x where it is applicable w.r.t x
- **defeated** w.r.t $x \in \triangle^{\mathcal{I}}$ iff $\exists C \sqsubseteq D < A \sqsubseteq B$ such that $C \sqsubseteq D$ is applied w.r.t x. In this case, we say that $C \sqsubseteq D$ defeats $A \sqsubseteq B$ w.r.t x.

Although definition 4 is defined on TBox(terminological axiom) only, it's easy to simulate the ABox with TBox axioms:

$$C(a) \iff \{a\} \sqsubseteq C$$
$$R(a, b) \iff \{a\} \sqsubseteq \exists R.\{b\}$$

For example, if we revisit the Animal Ontology in $\mathcal{OSHOQP}(\mathbf{D})$, The terminology $\mathcal{T}$ could be rewritten as

$package(1)$
(1a) $public(Dog, 1)$
(1b) $1 : Dog \sqsubseteq 1 : Carnivore$
(1c) $1 : Dog \sqsubseteq 1 : Pet$
(1d) $public(Animal, 1)$
(1e) $public(eats, 1)$
(1f) $1 : Carnivore \sqsubseteq 1 : Animal$
(1g) $1 : Carnivore \sqsubseteq \forall 1 : eats.1 : Animal$
(1h) $\{1 : betty\} \sqsubseteq 1 : Dog$

$package(2)$
(2a) $im(2, 1)$; `import package 1`
(2b) $public(Plant, 2)$
(2c) $2 : Plant \sqcap 1 : Animal \sqsubseteq \bot$
(2d) $2 : SickDog \sqsubseteq 1 : Dog \sqcap \exists 1 : eats.2 : Plant$

A simple combination of packages 1 and 2 is inconsistent on (1g) and (2d), i.e. there is no model for it. However, with a partial order $<$, this logical inconsistency can be eliminated. One such possible partial order is (2d)$<$(1g) (read as axiom (2d) is *stronger than* axiom (1g) or axiom (1g) is *weaker than* axiom (2d)). In this case, a specific axiom (2d) *defeats* the general rule (1g). When there is a logical conflict between a pair of axioms, the weaker of the two is discarded.

The specification of the partial 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 [AvH04]: If the source of one axiom may be more reliable than the source of another axiom, or has higher authority or social order, 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 relative to axioms from imported packages in cases where a local package can be seen as an extension or an exception to a general ontology. Other partial order assignment policies may be based on the social order of the agents in the Wiki@nt community, such as ontology administrator, package manager and common user.

D. Reasoning

Ontology editing requires some support for reasoning with ontologies. For example, when we define a new instance of a class C, the properties associated with the instance should be consistent with the superclass hierarchy of C, and the restrictions of each of C's (direct and indirect) superclasses. When an new axiom is proposed to be added to a package, the reasoning engine ideally should verify that the addition of the proposed axiom does not introduce any inconsistencies.

1) Reasoning in $\mathcal{SHOQ}(\boldsymbol{D})$ Ontology: One of the most important reasoning problems is the subsumption reasoning problem - the problem of determining if a class is a subclass of another class. Many other reasoning problems can be reduced to subsumption. For example

1. C and D are equivalent $\iff$ C is subsumed by D and D is subsumed by C.
2. C and D are disjoint $\iff$ C $\sqcap$ D is subsumed by $\bot$(bottom concept).
3. a is a member of C $\iff$ {a} is subsumed by C

The standard reasoning algorithm in DL is the Tableau algorithm. We restrict our discussion to the language of $\mathcal{SHOQ}(\mathbf{D})$ [HS01]. The general idea behind the standard Tableau algorithm is to reduce the subsumption problem to (un)satisfiability problem and try to construct a possible interpretation for given terminology. The reduction is easy to understand since $C \sqsubseteq D \iff C \sqcap \neg D$ is unsatisfiable. Transform $C \sqcap \neg D$ into negation normal form (NNF), i.e. negation occurs only in front of concept names. Denote the transformed expression as C_0, the algorithm starts with an ABox $A_0 = \{C_0\{x_0\}\}$, and apply consistency-preserving transformation rules (**tableaux expansion**) to the ABox as far as possible. If one possible ABox is found, C_0 is satisfiable and the subsumption is not true. If no possible ABox could be found, the subsumption is true. Interested reader is refered to [BN03] for Tableau algorithm and [HS01] for NNF transformation rules and tableaux expansion rules of $\mathcal{SHOQ}(\mathbf{D})$.

2) Distributed Reasoning in Modular Ontology: Now we turn to the reasoning in package-extended ontology $\mathcal{SHOQP}(\mathbf{D})$. Reasoning in package-extended ontology can be seen as distributed reasoning among autonomous ontology modules where no global semantics is guaranteed. Therefore, the whole reasoning process has to be built on local reasoning offered by individual modules.

SubsumptionAnswer (C, D, O)
Input: Concept C and D, Ontology $O =< P, W >$
Return: **True** or **False**

1) Construct an ABox $A = \{(C \sqcap \neg D)(x)\}$
2) Transform A into NNF according to [BN03]
3) FOR all package/views P being referred in A
4) RETURN Satisfiable (A, P)
5) END FOR

Satisfiable (S, P)
Input: Initial ABox set S, package/view P
Return: **True** or **False**

1) FOR all ABoxes A_i in S
2) Transform concepts in A_i into NNF w.r.t. visible entities from P
3) Do $\mathcal{SHOQ}(\mathbf{D})$ tableau transformation on A_i, get an augmented set of ABoxes S_i
4) $S' = S \cup S_i$
5) IF $\exists A \in S_i$ is complete and consistent
6) RETURN **True**;
7) ELSE
8) FOR $\forall P', im(P, P')$
9) IF Satisfiable (S', P') = True
10) RETURN **True**;
11) END IF
12) END FOR
13) END IF
14) END FOR
15) RETURN **False**;

The basic idea of **Satisfiable** algorithm is that a package or view can answer a **Satisfiable** request if a possible interpretation is found locally; otherwise it has to consult the packages and views in its domain. Although no global semantics is available, an interpretation of the "global" model is incrementally constructed by querying the relevant packages through the corresponding views. (Note that if two packages are mutually imported, one or both of them could be called more than once while with different parameter S. If the tableaux expansion can yield a complete and logical clash-free completion ABox set, **Satisfiable** will terminate even if mutually importing, or more generally, cyclic importing is possible).

It is easy to prove from the properties of the Tableau Algorithm that the **SubsumptionAnswer** algorithm is sound, terminable, complete and decidable, when each of the modules is limited to use $\mathcal{SHOQ}(\mathbf{D})$-concept description. Since we know satisfiability of $\mathcal{SHOQ}(\mathbf{D})$-concept description is PSPACE-complete in each of the package, the **SubsumptionAnswer** is also PSPACE-complete for this case.

3) Dynamically Loaded Distributed Reasoning: The cost of communication between modules is an important consideration in distributed reasoning In the **Satisfiable** algorithm, the communication occurs on line 9). The local communication cost in a single call to **Satisfiable** is the size of fed parameters $size(S)+size(P)$. $size(S) = \sum_{i=1}^{n} size(S_i)$, where $size(S_i)$ is the number of axioms in S_i. $size(S)$ increases along recursive iteration pathes since S is augmented by the tableaux expansions. $size(P)$ is trivially 1. The total communication cost in a single call to **Satisfiable** includes both the local communication cost and the communication costs in all its sub (recursive) calls. It depends on both the complexity of ABox set of expanded $A = \{(C \sqcap \neg D)(x)\}$ and the importing topology of the ontology. Suppose the domain of each module (package or view) is finite and the expanded importing path for every package has finite length, the final call times of **Satisfiable** is PSPACE-complete. Generally, the simpler the importing topology, the lower is the communication cost. We can reduce communication cost with loading or caching referred ontology modules in local storage such as memory. However, creating a centralized ontology model in memory defeats the very purpose of having a modular ontology. Hence, a tradeoff should be made between communication cost and memory cost. A (local or remote) partial ontology model will be dynamically loaded into local memory in a reasoning process only if it is needed. The partial model could be a package, a small part of a package, or even an axiom. As we will see in the next section, dynamically loaded distributed reasoning is essential in Wiki@nt since there is even no persistent in-memory model for a package when it is referred.

4) Nonmonotonic Reasoning in Modular Ontology: As noted earlier, collaborative ontology development using the proposed approach requires support for nonmonotonic reasoning in $\mathcal{OSHOQP}(\mathbf{D})$. Hence, we developed an algorithm for nonomonotonic reasoning in a package-extended ontology based on the result of reasoning in $\mathcal{OSHOQ}(\mathbf{D})$ [SH02]. The basic idea is to choose a preferred $\mathcal{SHOQP}(\mathbf{D})$ model based on the specified partial order $<$ when we construct the interpretation(model) by tableau expansion. For example, given the Animal Ontology (in II-C), if new axioms are added to say that if a sick dog is sent to pet hospital (i.e., it is a $TreatedDog$), it will not eat grass:
(2e) $2 : TreatedDog \sqsubseteq 2 : SickDog$
(2f) $2 : TreatedDog \sqsubseteq \neg\exists 1 : eats.2 : Plant$
(2g) $\{1 : billy\} \sqsubseteq 2 : SickDog$
and the partial order is (2f)<(2d)<(1g). Given the fact that betty is a sick dog, two possible interpretations of the ontology are given in table III. [SH02] gives a formal definition of a *preferred model*:

Definition 5: The **support** for a model $\mathcal{I}$ of $\Sigma =< \mathcal{T}, <>$ is the set $S^{\mathcal{I}} = \{(x, A \sqsubseteq B) | x \in \triangle^{\mathcal{I}}, A \sqsubseteq B \in \mathcal{T}$ is classically satisfied (see definition 3) w.r.t. x and $\mathcal{I}\}$ ([SH02] Definition 3)

Definition 6: A model $\mathcal{I}$ of a knowledge base Σ is **preferred** over another model $\mathcal{J}$ of Σ, denoted $I \preceq J$ if $\forall (x, A \sqsubseteq B) \in \mathcal{S}^{\mathcal{J}} \backslash \mathcal{S}^{\mathcal{I}}, \exists (x, C \sqsubseteq D) \in \mathcal{S}^{\mathcal{I}} \backslash \mathcal{S}^{\mathcal{J}}, C \sqsubseteq D < A \sqsubseteq B$ ([SH02] Definition 4)

Definition 7: A model $\mathcal{I}$ of a knowledge base Σ is a **preferred model** of Σ if there is no other model $\mathcal{J}$ of Σ each that $J \preceq I$ ([SH02] Definition 5)

The intuition behind this definition is a "good" model should defeat least preferred axioms. Intuitively, I_1 is less preferred than I_2 because I_1 makes stronger assumption that billy is a $TreatedDog$ and defeats 2d, while I_2 defeats only a weaker axiom 1g. Thus I_2 is a preferred model. A modified version of **Satisfiable** is given as following:

Satisfiable' $(S, P, <)$
Input: Initial ABox set S, package/view P, partial order $<$
Return: **True** or **False**

```
1)  FOR all ABoxes A_i in S
2)    Transform concepts in A_i into NNF w.r.t. visible
      entities from P
3)    Do SHOQ(D) tableau transformation on A_i, get an
      augmented set of ABoxes S_i
4)    S' = {x | x ∈ S ∪ S_i and x is a preferred model}
5)    IF ∀A ∈ S' are complete and consistent
6)      RETURN True;
7)    ELSE
8)      FOR ∀P', im(P, P')
9)        IF Satisfiable (S', P', <) = False
10)         RETURN False;
11)       END IF
12)     END FOR
13)     RETURN True;
14)   END IF
15) END FOR
```

The basic idea of this algorithm is to limit the search only to preferred models; if all preferred models assert that $C \sqsubseteq D$, then $C \sqsubseteq D$ is acceptable in the $\mathcal{OSHOQP}(\mathbf{D})$ ontology. Formally, we call C is defeasibly subsumed by D if **SubsumptionAnswer**(C,D,Σ) returns **True**

Definition 8: For concept C and D in KB Σ , C is **defeasibly subsumed** by D , denote as $\Sigma \rightsquigarrow C \sqsubseteq D$, iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for each preferred model $\mathcal{I}$ of Σ ([SH02] Definition 7)

III. ARCHITECTURE

$\mathcal{OSHOQP}(\mathbf{D})$ gives us an expressive language to build ontology from autonomous, distributed, and possibly inconsistent ontology modules. Wiki@nt is the implementation of an ontology editor based on $\mathcal{OSHOQP}(\mathbf{D})$ to support collaborative ontology building by a community of autonomous domain experts, organizations, or even software agents.

TABLE III
POSSIBLE INTERPRETATIONS OF THE ANIMAL ONTOLOGY

	Dog	SickDog	TreatedDog	$\exists eats.Plant$	$\forall eats.Animal$	...	satisfies
I_1	billy	billy	billy	$\varnothing$	billy	...	all but 2d
I_2	billy	billy	$\varnothing$	billy	$\varnothing$	...	all but 1g

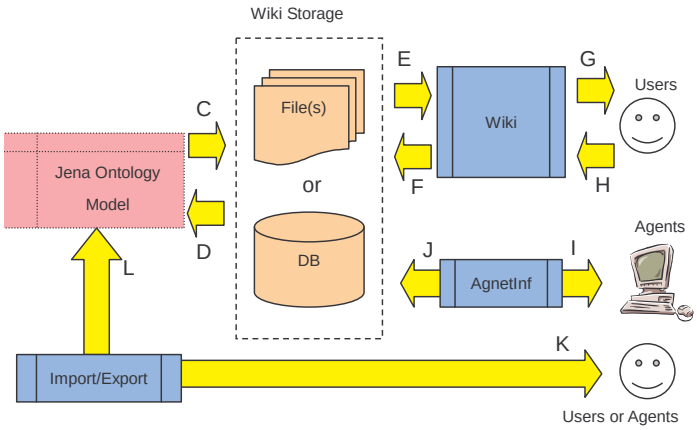

Fig. 1. The Architecture of Wiki@nt

The name "Wiki@nt" suggests that it has a wiki-like editing environment. Wiki is originally a collaborative documentation writing/website building tool. Typical wiki system includes a script language (usually a simplified subset of HTML tags), a set of wiki pages written in the script language and shown in translated HTML pages,a RCS version control system to record modification of contents, an user profile and concurrent conflict management system to enable multiple user editing the same contents, a content navigation system such as showing link-in and link-out pages, and a simple-to-use, browser-based editing environment to generate or modify content on the fly.

We find that those features are quite desirable in a collaborative ontology editor. While most widely-used ontology editors, such as Protege and OilEd, work very well for the task of developing a single ontology module, they do not lend themselves to collaborative ontology building. This is due to the lack of a built-in formalism to support modular ontology representation, and the lack of support for communication and cooperation among multiple individuals in editing a shared ontology consisting of multiple, independently developed, possibly partially overlapping modules. To overcome those deficiencies, we propose to use wiki to edit $\mathcal{OSHOQP}(\mathbf{D})$ ontology. An ontology module is composed of one or more wiki pages; multiple users can edit the same content, with version control and transaction management; Ontology are loaded into or uploaded from a set of wiki pages and managed by a ontology repository. Figure 1 shows the architecture of Wiki@nt.

A. Wiki: markup script and the editor

We defined a set of markup script tags to correspond to the syntax of the ontologies. When a wiki page is under editing, its wiki markup script is loaded and translated to user friendly text, such as HTML web page. The syntax is a extension to OWL to support package and partial order on axioms. Wiki markup script is a human readable syntax equivalent to the N-Triple syntax. N-Triple syntax is an alternative to the RDF/XML syntax and each line in N-Triple serialization is a triple statement with subject, predicate and object. For example, axiom SubClassOf(Dog , Carnivore) in the Animal Ontology could be represented by N-Triple syntax as: <http://mydomain.org/animal#Dog> <http://www.w3.org/2000/01/rdf-schema#subClassOf> <http://mydomain.org/animal#Carnivore>, or in short form <animal:Dog> <rdfs:subClassOf> <animal:Carnivore>. It's wiki script is [animal:Dog] [rdfs:subClassOf][animal:Carnivore]

Each axiom is assigned a URI (uniform resource identifier) as label. Thus, for example, http://mydomain.org/animal/package1#Dog represents $Dog \sqsubseteq Carnivore$ in package(1), Animal Ontology. A partial order can be specified as [uri1] [wiki:stronger][uri2].

User can create a new page or modify the source script of an existing page. The editing action is assisted by several wizards (such as class creating wizard) and a browser (eg. Show subclass and superclass of the class in question).

B. Wiki Engine

A wiki engine

- Provides a web interface for ontology editor and browser.
- Translates the Wiki@nt script to HTML code to be shown in the web browser.
- Manages the storage of wiki pages, in plain file or database.
- Provides version control. When a modification for an axiom is submitted, the previous version is stored and could be restored when the committed version is found incorrect or impropriate.
- Provides transaction management. Wiki@nt denies the write-access of agents to a page,or possibly also related pages such as subclasses in the class hierarchy, if it is locked by some other agents.
- Generates reference report for each wiki pages. All terms be used in an axiom group, and all other groups that referring this group, are listed for browsing purpose.
- Generates a RSS feed for ontology repository updates.

The wiki engine we utilized is based on the JSPWiki(http://www.jspwiki.org) and implemented in Java and JSP.

C. Wiki pages : axiom groups

While most of the popular ontology editors have in-memory model for edited ontology, Wiki@nt doesn't maintain a in-

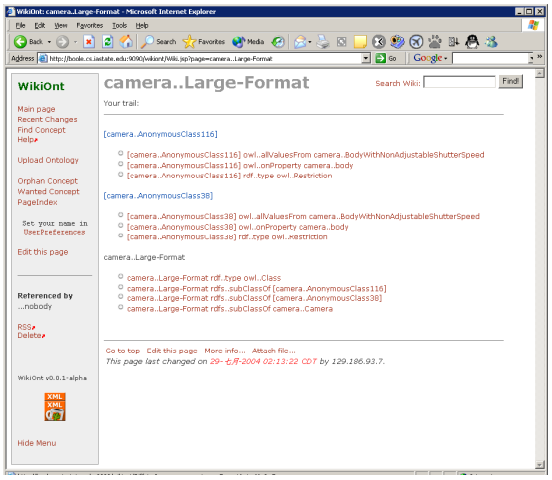

Fig. 2. a Wiki@nt page

memory model for each resident ontology for several reasons: An in-memory model limits the scalability of the system with respect to both the axioms number in one ontology and the number of ontologies in the Wiki@nt ontology repository; In-memory model implicitly assumes the existence of a global ontology during the ontology development process and requires monotonic behavior of the ontology - neither of these assumptions is desirable in a collaborative ontology building scenario.

Note that that even when the size of the ontology in question is huge, usually only a small fraction of its axioms are involved during an editing action. Hence, we store the ontology as a set of separate, possibly distributed blocks in Wiki@nt. Each block is serialized to external storage when it's not being actively edited, and being loaded into the memory only if it's edited or referred. This is inspired by widely used techniques of database memory management where partial content of the database is dynamically loaded and unloaded to allow manipulation of of a much larger volume of data than can fit in limited memory.

Although different decompositions of an ontology package are logically equivalent, the size of each ontology block will affect the convenience and efficiency of ontology editing and reasoning. It should not be too big (i.e. the whole package), or too small (e.g., a single triple). In Wiki@nt, we choose axiom groups as ontology blocks. Each axiom group contains triples with same subject. For example, the axiom groups in Animal Ontology package(1) will be Dog, Carnivore, eats, and billy. Restrictions and anonymous classes, are assigned to the terms from where they are referred. Each axiom group is translated to wiki markup script and stored as a wiki page. An ontology could be stored distributedly in multiple pages, physically in file or database ,and could be dynamically, partially loaded when necessary. Figure2 shows a example of axiom group.

D. Ontology exporting/importing

When an ontology is needed e.g., for reasoning, we export wiki pages as a single ontology file or read an ontology file into Wiki Ontology Repository. The relevant portion of an ontology is extracted or assembled from the wiki pages. We use the Jena toolkit to create the in-memory model and as parser/writer for ontology files.

Each loaded ontology is assigned a unique name, eg. http://mydomain.org/animal/, and it's member packages, eg. http://mydomain.org/animal/package1, are registered to that ontology. It's also possible that packages from different ontologies could be reassembled into a new ontology, thus provide a flexible way for ontology reuse and integration.

E. The Agent Interface

While fully automatic ontology construction and inter-ontology mapping are impossible, software agents can assist humans in several aspects of collaborative ontology development e.g., finding useful concepts and relations among concepts from original data sources. Small pieces of ontologies, such as consistent concept (term) in data or concurrence of two concepts, can be generated by software agents. The results may be subjected to review of domain experts, or even other software agents. Hence, although our current design of Wiki@nt does not include support for software agents, we do reserve an RPC interface that enables agents to communicate with Wiki@nt. Thus, in principle, it is possible for software agents to participate in collaborative ontology building using Wiki@nt.

IV. Related Work

A. Modular Representation and Reasoning in Description Logic

1) Distributed Logics: : A number of distributed logics system have been studied during recent years. Examples include Local Model Semantics [GG01] and Distributed First Order Logic (DFOL) [GS98] which emphasize local semantics and the compatibility relations among local models. Inspired by DFOL, [BS02] extends 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". Bridge rules are unidirectional thereby ensuring that there is no "back-flow" of information among modules being connected by a bridge rule. When the number of modules to be connected is large, the explicit declaration of such bridge rules becomes tedious. [QG04] extends local model semantics and harmonizes local models via agreement on vocabulary provenance.

2) Modular Ontologies: Two approaches to integration of separate ontologies have been developed based on DFOL and DDL. The Modular Ontology, [SK03] offers a way to exploitat modularity in reasoning. It also defines an architecture that supports local reasoning by compiling implied subsumption relations. It also offers a way to maintain the semantic integrity of an ontology when it undergoes local changes. In the "view-based" approach to integrating ontologies, all external concept definitions are expressed in the form of queries. However, A-Box is missing in the query definition, and the mapping be-

tween modules is unidirectional making it difficult to preserve local semantics.

3) Contextual Ontology: : Contextual logic, a formalism based on DDL, emphasizes localized semantics in ontologies. Contextual ontology keeps contents local and maps the content to other ontologies via explicit bridge rules. [BDSZ02] proposed CTXML, which includes a hierarchy-based ontology description and a context mapping syntax. [PB03] combined CTXML and OWL into Context OWL (C-OWL), a syntax for bridge rules over OWL ontology. Our approach based on P-OWL has several improvements over C-OWL by introducing scope limitation modifiers (SLM) and query-based views. Bridge rules can be viewed as special cases of queries and SLM offers a controllable way to keep content local by definition.

B. Non-monotonic Reasoning in Description Logic

Nonmontonic reasoning in description logic has received considerable attention in the literature. [BH93] and [BH92] introduced defaults in the description logic. [QR93] studied preferred models and split axioms into defeasible and not defeasible axioms. [SH02], [HV02] extended defeasible reasoning to description logic, with a partial 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 tableau-based reasoning algorithm.

Specifically, $\mathcal{OSHOQP}(\mathbf{D})$ proposed in this paper offers a possible extensions to the OWL language to support default reasoning. The system allows inherited values to be overridden by more specific classes, treating the inherited values as default. $\mathcal{OSHOQP}(\mathbf{D})$ embodies a closed-world assumption (a statement is assumed to be true when it negation cannot be proved), as opposed to the open-world assumption (a statement cannot be assumed true on the basis of failure to prove its negation) currently adopted by OWL. A careful investigation of the relative advantages of the closed versus open world assumptions in specific application scenarios deserves further attention.

C. Collaborative Ontology Editor

Several ontology editors have been reported in the literature [Den02] [Ont]. However, most existing ontology editors including the most widely used ontology editors Protege [GMF+02] and OilEd [BHGS01] provide little support for collaborative ontology development. The ontology editors that support collaborative ontology editing are listed in TableIV. Most of them provide concurrent access control with transaction oriented locking, and in some cases, even rollback. However, none of the existing ontology editors, to the best of our knowledge, provides principled approaches for manipulating independently developed, semantically heterogeneous ontology modules or for reconciling logical inconsistencies between such modules.

D. Collaborative Knowledge Base Construction

Some collaborative knowledge base construction projects, although not focused on ontology building, address similar problems.

1) Nooron: Nooron[1] is a knowledge publishing system and has a wiki for ontology browsing.

2) MnM: MnM [VVMD+02] is an annotation tool which provides both automated and semi-automated support for annotating web pages with semantic contents. MnM integrates a web browser with an ontology editor and provides open APIs to link to ontology servers and for integrating information extraction tools. However, it doesn't have concurrent access control.

3) FOAF: FoaF[2] is an acronym for "Friend of a Friend" , an experimental project and vocabulary for the Semantic Web. It is based on the idea of a machine-readable version of the current World Wide Web, with homepages, mailing lists, travel itineraries, calendars, address books and the likes. The project is open and allows participants to add their own information. The result is an RDF based knowledge base containing contact and acquaintance information about the participants.

4) WikiPedia: WikiPedia [3] is a wiki-based open-content encyclopedia that is available in several languages. There are 315,000 articles in English alone as of July, 2004. It is an open encyclopedia that is editable by participants. WikiPedia works and assumes that that most of people in the community behave in a manner that benefits the community. Articles in WikiPedia are written in natural language, and the relation between items is not formal. Nevertheless, articles can be seen as concepts and links between them seen as properties among them, in a informal sense.

5) Open Directory Project: or called DMOZ [4] is an online, open, collaborative taxonomy building project for web catalog. Now it has about 64,200 editors and a taxonomy tree of over 590,000 categories and over 4 million sites classified into categories. The relations between DMOZ concepts is just strict "subClassOf". Both WikiPedia and DMOZ knowledge base are open source inspired and freely available.

Although these projects lack formalized and full-fledged ontologies, they offer interesting demonstrations of the feasibility of collaborative ontology development. The Wiki@nt collaborative ontology development environment proposed in this paper is inspired by the success of DMOZ and WikiPedia, and aims to provide support for such efforts using a formal ontology language to facilitate machine interpretable annotations of data.

V. Summary and Discussion

In this paper we have described

- A $\mathcal{OSHOQP}(\mathbf{D})$ description logic to support defeasible reasoning with modular ontologies for collaborative

[1] http://www.nooron.org
[2] http://www.foaf-project.org/
[3] http://en.wikipedia.org/
[4] http://www.dmoz.org/

TABLE IV

COLLABORATIVE ONTOLOGY EDITORS

Tool	Base Language	Import/Export	More Information
DOME	CLASSIC & FaCT	OKBC, XML	http://more.btexact.com/projects/ibsr/dome/index.htm
IODE	KIF	KIF, UML, RDB, XML,DTD	http://www.ontologyworks.com/
KAON	KAON	RDFS	http://kaon.semanticweb.org/
LinKFactory Workbench	Extended description logic	XML, RDF(S), DAML+OIL/OWL	http://www.landc.be/
Onto-Builder	LOK	DAML+OIL; XML, LOK, KIF	http://ontology.univ-savoie.fr
OntoEdit [SEA+02]	F-Logic	RDFS; F-Logic; DAML+OIL RDB	http://www.ontoprise.de/com/ontoedit.htm
Ontolingua [FFPR95]	Ontolingua	DAML+OIL, KIF, OKBC, Loom, Prolog, Ontolingua, CLIPS,...	http://www.ksl.stanford.edu/software/ontolingua/
Ontosaurus	Loom	KIF, Loom, OKBC	http://www.isi.edu/isd/ontosaurus.html
OpenKnoMe	GRAIL	CLIPS, XML	http://www.topthing.com/
WebKB	FS (extended CGs)	DAML/RDF; CGIF; KIF	http://meganesia.int.gu.edu.au/ phmartin/WebKB/
WebODE	Prolog	DAML+OIL, RDFS, X-CARIN, FLogic, Prolog, XML	http://delicias.dia.fi.upm.es/webODE/
WebOnto	OCML	RDF, RDFS, GXL, Ontolingua, OIL	http://kmi.open.ac.uk/projects/webonto/

ontology construction, ontology integration and reconciliation.

- Distributed reasoning algorithms in both monotonic modular ontology and defeasible modular ontology.
- A distributed ontology representation and storage methodology based on wiki.
- A Light-weight ontology editor to support collaborative ontology building

Some interesting directions for future work include:

- Incorporation of transaction management and incorporation of safe mechanisms for handling simultaneous editing and modification of ontologies
- Detailed complexity analysis of the reasoning algorithm in $\mathcal{SHOQP}(\mathbf{D})$ and $\mathcal{OSHOQP}(\mathbf{D})$ including bounds on the communication cost for the tableu based reasoing for $\mathcal{SHOQ}(\mathbf{D})$
- Investigation of useful policies for assigning partial order among axioms, including those that are base don machine learning or probabilistic approaches [GL02], [DP04].
- Investigation of learning agents for generating ontologies (e.g., taxonomies over attribute values) from data [KSZH04]. Such agents can assist domain experts in ontology building.
- Applications of collaborative ontology building environments for information integration from autonomous, distributed, semantically heterogeneous information sources [CPH04] in knowledge and data intensive domains like bioinformatics, security informatics, and more generally, semantic web [BLHL01].

ACKNOWLEDGMENTS

This research is supported in part by grants from the National Science Foundation (0219699) and the National Institutes of Health (GM 066387) to Vasant Honavar

REFERENCES

[AvH04] Grigoris Antoniou and Frank van Harmelen. *A Semantic Web Primer*. The MIT Press, 2004.

[BDSZ02] P. Bouquet, A. Dona, L. Serafini, and S. Zanobini. Conceptualized local ontologies specification via ctxml. In *Working Notes of the AAAI-02 workshop on Meaning Negotiation, Edmonton (Canada)*, 2002.

[BH92] Franz Baader and Bernhard Hollunder. How to prefer more specific defaults in terminological default logic. Technical Report RR-92-58, 1992.

[BH93] Franz Baader and Bernhard Hollunder. Embedding defaults into terminological knowledge representation formalisms. Technical Report RR-93-20, 1993.

[BH04] J Bao and V Honavar. Ontology language extensions to support localized semantics, modular reasoning, collaborative ontology design and reuse. 2004.

[BHGS01] Sean Bechhofer, Ian Horrocks, Carole Goble, and Robert Stevens. OilEd: A reason-able ontology editor for the semantic Web. *Lecture Notes in Computer Science*, 2174:396–408, 2001.

[BLHL01] T Berners-Lee, J Hendler, and Ora Lassila. The semantic web. *Scientific American*, 284(5):34–43, May 2001.

[BN03] F. Baader and W. Nutt. Basic description logics. In Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors, *The Description Logic Handbook: Theory, Implementation, and Applications*, pages 43–95. Cambridge University Press, 2003.

[BS02] A. Borgida and L. Serafini. Distributed description logics: Directed domain correspondences in federated information sources, 2002.

[Car04] Doina Caragea. *Learning Classifiers From Distributed, Semantically Heterogeneous, Autonomous Data Sources.* PhD thesis, Iowa State University, 2004.

[CPH04] Doina Caragea, J Pathak, and Vasant Honavar. Learning classifiers from semantically heterogeneous data. In *VLDB 2004 Workshop on the Semantic Web and Databases (SWDB 2004)*, To appear, 2004.

[CSC+03] Jaime A Reinoso Castillo, Adrian Silvescu, Doina Caragea, Jyotishman Pathak, and Vasant G Honavar. Information extraction and integration from heterogeneous, distributed,autonomous information sources - a federated ontology-driven query-centric approach. In *Proceedings of the IEEE International Conference on Information Reuse and Integration*, 2003.

[Den02] Michael Denny. Ontology building: A survey of editing tools. Technical report, O'Reilly XML.com, November 06, 2002.

[DP04] Z. Ding and Y Peng. A probabilistic extension to the web ontology language owl. In *Thirty-Seventh Hawaii International Conference on System Sciences (HICSS-37)*, 2004.

[FFPR95] A. Farquhar, R. Fikes, W. Pratt, and J. Rice. Collaborative ontology construction for information integration, 1995.

[GG01] C. Ghidini and F. Giunchiglia. Local model semantics, or contextual reasoning = locality + compatibility. *Artificial Intelligence*, 127(2):221–259, 2001.

[GL02] Rosalba Giugno and Thomas Lukasiewicz. P-shoq(d): A probabilistic extension of shoq(d) for probabilistic ontologies in the semantic web. In *8th European Conference on Logics in Artificial Intelligence (JELIA'02), Cosenza, Italy*, Lecture Notes in Artificial Intelligence. Springer, September 2002 2002.

[GMF+02] J. Gennari, M. A. Musen, R. W. Fergerson, W. E. Grosso, M. Crubzy, H. Eriksson, N. F. Noy, and S. W. Tu. The evolution of protege: An environment for knowledge-based systems development. Technical Report SMI-2002-0943, SMI, Stanford, 2002.

[GS98] C. Ghidini and L. Serafini. *Frontiers Of Combining Systems 2, Studies in Logic and Computation*, chapter Distributed First Order Logics, pages 121–140. Research Studies Press, 1998.

[Hor02] Ian Horrocks. DAML+OIL: a description logic for the semantic web. *IEEE Data Engineering Bulletin*, 25(1):4–9, 2002.

[HPSvH03] Ian Horrocks, Peter F. Patel-Schneider, and Frank van Harmelen. From SHIQ and RDF to OWL: The making of a web ontology language. *Journal of Web Semantics*, 1(1), 2003.

[HS01] Ian Horrocks and Ulrike Sattler. Ontology reasoning in the shoq(d) description logic. In B. Nebel, editor, *Proceeding of the 17th Int. Joint Conf. on Artificial Intelligence*, pages 199–204. AAAI, Morgan Kaufmann, 2001.

[HV02] S. Heymans and D. Vermeir. Using preference order in ontologies, 2002.

[KSZH04] Daeki Kang, A. Silvescu, Jun Zhang, and Vasant Honavar. Generation of attribute value taxonomies and their use in data-driven construction of accurate and compact naive bayes classifiers. In *Proceedings of the ECML/PKDD Workshop on Knowledge Discovery and Ontologies (KDO-2004)*, In press, 2004.

[Ont] Ontoweb. Deliverable 1.3: A survey on ontology tools. Technical report.

[PB03] F. van Harmelen etc. P. Bouquet, F. Giunchiglia. C-OWL: Contextualizing ontologies. In *Second International Semantic Web Conference*, volume 2870 of *Lecture Notes in Computer Science*, pages 164–179. Springer Verlag, 2003.

[QG04] Yuzhong Qu and Zhiqiang Gao. Interpreting distributed ontologies. In *Alternate track papers & posters of the 13th international conference on World Wide Web*, pages 270–271. ACM Press, 2004.

[QR93] J.J. Quantz and M. Ryan. Preferential default description logics. Technical Report KIT 110, Technische Universitat, Berlin, 1993.

[SD04] G. Schreiber and M. Dean. Owl web ontology language reference, February 2004.

[SEA+02] Y. Sure, M. Erdmann, J. Angele, S. Staab, R. Studer, and D. Wenke. OntoEdit: Collaborative ontology development for the semantic web. In *Proceedings of the first International Semantic Web Conference 2002 (ISWC 2002), June 9-12 2002, Sardinia, Italia*. Springer, LNCS 2342, 2002.

[SH02] D. Vermeir S. Heymans. A defeasible ontology language. In et al. (Eds.) R. Meersman, Z. Tari, editor, *On the Move to Meaningful Internet Systems 2002: CoopIS, DOA, and ODBASE : Confederated International Conferences CoopIS, DOA, and ODBASE 2002, Lecture Notes in Computer Science*, volume 2519, pages 1033–1046. Springer-Verlag Heidelberg, 2002.

[SK03] Heiner Stuckenschmidt and Michel Klein. Modularization of ontologies. Technical report, WonderWeb: Ontology Infrastructure for the Semantic Web, IST Project 2001-33052,, 2003. Version 1.0.

[VVMD+02] Maria Vargas-Vera, Enrico Motta, John Domingue, Mattia Lanzoni, Arthur Stutt, and Fabio Ciravegna. Mnm: Ontology-driven tool for semantic markup. In Siegfried Handschuh, Niegel Collier, Rose Dieng, and Steffen Staab, editors, *Proceedings Workshop on Semantic Authoring, Annotation and Knowledge Markup (SAAKM 2002)*, pages 43–47, 2002.