Ontology Language Extensions to Support Collaborative Ontology Building

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

Abstract. Modular approaches to design and use of ontologies are essential to the success of the Semantic web. We describe P-OWL (Package-based OWL), which extends OWL, that supports modular design, adaptation, use, and reuse of ontologies. P-OWL localizes the semantics of entities and relationships in OWL to modules called packages. P-OWL and the associated tools will greatly facilitate collaborative ontology construction, and reuse.

Keywords: *Modular Ontology, Contextual Ontology, OWL, Local Semantics*

1. Introduction

By its very nature, ontology construction is a collaborative process that involves direct **cooperation** among domain experts, or indirect cooperation through **reuse** of autonomously developed, very likely, semantically heterogeneous ontologies. Typically, multiple relatively autonomous groups contribute parts of a real-world ontology that pertain to their domains of expertise or responsibility. The integrated ontology should be a semantically coherent integration of the constituent ontologies developed by the individual groups. Some desiderata of collaborative ontology construction tools include:

¤ **Localized Semantics**: Unrestricted use of entities and relationships from different ontologies can result in serious semantic conflicts, since the ontologies usually represent *local views* of the world.

¤ **Ontology Reuse:** lack of modularity and localized semantics in ontologies forces an *all or nothing* choice with regard to reuse of an existing ontology. Modular ontologies could facilitate more flexible and efficient reuse of existing ontologies.

¤ **Knowledge Hiding**: In many applications, the provider of an ontology may not wish, because of copyright, privacy or security concerns, to make the entire ontology visible to the outside while willing to expose partial ontology to certain subsets of users.

¤ **Distinction between Organizational and Semantic Structure**: organizational structure is how the terms are put together for practical use while semantic structure is the definitional linkage of them. Mixture of these two structures usually leads to logical ambiguity.

Relatively little attention has been paid to formalisms for collaborative construction in such settings. This state of affairs in ontology engineering is reminiscent of the early stage of software engineering when uncontrolled use of global variables, spaghetti code, absence of well-defined modules leading to uncontrolled interactions between code fragments. Hence, there is an acute need for formalisms that facilitate collaborative modular design, adaptation, and reuse of ontologies. Against this background, this paper describes Package-extended Ontologies to support those needs.

2. Package-extended Ontologies

Current ontology languages, such as OWL, while offer some degree of modularization by restricting ontology segments into separated XML namespaces, fail to fully support above-mentioned requirements. In this paper, we argue for ***package*** based ontology language extensions to overcome these limitations. 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***. Packages provide an attractive way to compromise between the need for knowledge sharing and for knowledge hiding in collaborative building of ontologies. The structured organization of ontology elements in packages bring to ontology design and reuse the same benefits as those provided by packages and reuse in software engineering.

Definition 1 an **Ontology Entity** is an axiom $e=[C|P|I]$ where C is a class definition, P is a property definition and I is an instance definition.

Definition 2 Scope Limitation Modifier (SLM) of an ontology entity e is a boolean function V_e(r), where r is the identifier of a model that refers e. Model(r) could access e iff V_e(r) = **True**.

Definition 3 A **basic package** is a logic model $\boldsymbol{P}_b$= <E, V> where E={e_i} is a set of entities and V={v_i} is the set of their SLMs. A **compositional package** is a logic model of $\boldsymbol{P}_c$=<E, V, P> where E={e_i} and V={v_i} are sets of entities and their SLMs and P is a set of basic/compositional packages. For all P_i ˛ P, we say P_i is ˛$_N$ (***NestedIn***) $\boldsymbol{P}_c$. Packages could be recursively nested to form a package (**organizational**) hierarchy.

Definition 4 $\boldsymbol{P}$ is called the **home package** of e_i and denoted as $\boldsymbol{P}$ = $HP(e_i)$. For compositional package, $\boldsymbol{P}$= $HP(P_i)$ for all P_i ˛ P.

Definition 5 three **default SLMs** are specified as:

- *Public* $_e$(r) := True
- *Protected* $_e$ (r) := (r = identifier of $HP(e)$) ∨ Model(r) ˛$_N$ $HP(e)$
- Private e (r) := (r = identifier of $HP(e)$).

Definition 6 Shallow Default Interface I_s of a package $\boldsymbol{P}$ is a subset of $\boldsymbol{P}$'s signature such that EN_i ˛ I_s iff e_i ˛$_{vi}$ $\boldsymbol{P}$ and V_i(r) = **True**, for ∀r where EN_i and V_i(r) are the name and SLM of entity e_i. **Deep Default Interface,** or for short, **Default Interface,**

I_d of a package P is the union of its own shallow default interface I_s and the deep default interface of its home package. $I_d(P) = I_s(P) \cup I_d(HP(P))$.

Now we turn to connecting the modules by specifying *mappings* between them. We argue that to maintain the local semantics of a package, view-based mappings provide a better alternative than one-to-one name mapping.

Definition 7 a **distributed interpretation** of a set of packages $\{P_i\}$, i=1,…m is a family $\mathcal{I}_d=\{\mathcal{I}_i\}$ where $\mathcal{I}_i=<\Delta^{\mathcal{I}_i}, (.)^{\mathcal{I}_i}>$ is the local interpretation of P_i. The union of all $\Delta^{\mathcal{I}_i}$ is the distributed concept space $\Delta^{\mathcal{I}_d}$ and $(.)^{\mathcal{I}_d} = \subseteq \{ \Delta^{\mathcal{I}_d} \cdot \Delta^{\mathcal{I}_d} \}$ is the distributed role space.

Definition 8 Given a set of packages $\{P_i\}$, and $e_1,\ldots,e_m$ are some entity names in $\{I_d(P_i)\}$. $\mathcal{I}^d$ is the distributed interpretation of $\{P_i\}$. A **query** over $\{P_i\}$ is an expression of one of the following forms:

- **Class Query:** $C_q(x) := f_c(e_1,\ldots, e_m) \subseteq \Delta^{\mathcal{I}_d}$ where f_c is a unary logic construction function.
- **Property Query**: $P_q(x,y) := f_p(e_1,\ldots, e_m) \subseteq (.)^{\mathcal{I}_d}$ where f_p is a binary logic construction function.
- **Instance Query**: $I_q := f_i(e_1,\ldots, e_m)$ ¸ $\Delta^{\mathcal{I}_d}$ where f_i is logic construction function with no variable.

Definition 9 a **view** W over a set of packages $\{P_i\}$ is a set of queries over $\{P_i\}$. $\{P_i\}$ is called the **domain** of the view. An **interface** F over package P is a view over and only over P.

One module can have multiple interfaces thus allow partial reuse of the huge ontology. Views also offer a reusable mechanism to connect packages if they (the views) are defined over multiple packages.

Definition 10.(**Imported**) a package P_1 /view W is said being imported into a package P_2 if the default interface of P_1 / subset of the signature of W is used in some entity definition axioms in P_2. The set of all imported packages and views of a package P is called the **domain** of P.

Definition 11 a **package-extended ontology** $O = <P,W>$ where P is a set of packages , W is a set of views defined on P. P and W constitute an importing closure.

When a package P or view W is imported into a package, note that only the signature (name set)of that P /W is used. The referring package only takes care of the set of referred names, while semantics of its domain are maintained intact.

3. Reasoning in Packages

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. We focus on the most important reasoning task, **subsumption**. The algorithm is an extension to the standard *Tableau Algorithm* [BC2003].

SubsumptionAnswer (C, D, O)

```
Input: Concept C and D, Ontology O=<P, W>
Return: True or False
1.  Construct an ABox A = {C⊓ D (x)}, Transform A
    into negation normal form (NNF).
2.  FOR all package/views $P$ being referred in A
3.  { RETURN   Satisfiable ({A}, $P$) ; }
```

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, wrt
    visible entities from $P$;
3.    Do ABox transformation as that in standard
      tableau algorithm, result in an augmented set of
      ABoxes $S_i$, $S' = S \cup S_i$.
4.      IF ∃A ¸ $S_i$ is complete(no transformation rule
    applies to it) and consistent (no logic clash)
5.          {RETURN True;}
6.      ELSE{
7.          FOR all imported packages/views $P'$
8.          {IF Satisfiable ($S'$, $P'$) RETURN True;} } }
9.  RETURN False;
```

The basic idea of **Satisfiable** algorithm is that a package or view could answer a **Satisfiable** request if a possible interpretation is found locally; otherwise it will consult the packages/views in its domain. Although no global semantics is available, an interpretation of the "global" model is incrementally constructed by the queries among packages/views.

4. Discussion

Compared with *Modular Ontology* [SK2003], our approach includes A-Box query definition, and the mapping between modules is directional thus local semantics is preserved. The improvement of P-OWL over Contextual ontology/C-OWL [BG2003] is the introduction of SLM and query-base view. Bridge rules could be seen as special cases of query and SLM offers a controllable way to keep content local by definition.

Future work includes more careful investigation of the reasoning algorithm; the study of the basic operations needed in reasoning with package and view, such as the construction of default interface of a package; Efficient representation of mapping between packages; and tools to support P-OWL, such as ontology editor and reasoner.

Acknowledgement: This work is supported by grants from NSF (0219699) and NIH (GM 066387).

References

[BC2003] F. Baader,D. Calvanese, etal(ed) The Description Logic Handbook. Cambridge Univ. Press, 2002.

[BG2003] P. Bouquet, F. Giunchiglia, etal. {C-OWL}: Contextualizing ontologies. In 2nd ISWC, LNCS, (2870),164-179. Springer Verlag, 2003.

[SK2003] H. Stuckenschmidt, M.Klein. Modularization of Ontologies. WonderWeb report. Version 1.0. 26.6.2003