A Survey of Formalisms for Modular Ontologies

Yimin Wang[1], **Peter Haase**[1] **and Jie Bao**[2]
[1]Institute AIFB, University of Karlsruhe, Germany
{pha,ywa}@aifb.uni-karlsruhe.de
[2]Artificial Intelligence Research Laboratory, Department of Computer Science
Iowa State University, Ames, IA 50011-1040, USA
baojie@cs.iastate.edu

Abstract

In this paper, we investigate state-of-the-art technologies in the area of modular ontologies and corresponding logical formalisms. We compare the strengths and weaknesses of different formalisms for modular ontologies in their ability to support networked, dynamic and distributed ontologies, as well as the reasoning capability over these ontologies. The comparison result shows limitations of existing formalisms to fully address the need of modular ontologies in the given setting, and possible future extensions to overcome those limitations.

1 Introduction

Knowledge management with distributed knowledge repositories is usually a challenging task in which each knowledge repository may pertain only a subset of the domain in question and the inconsistencies between those dynamically-changing local repositories are hard to detect or to control. A typical application scenario is that large international organizations, which may have branches around the globe and maintain multiple, distributed, large knowledge bases for each of their branch. Those knowledge bases, often represented as ontologies, are typically maintained by local branches of the organization in a collaborative way. While those ontologies are usually focused on the local knowledge of particular local branches and are physically distributed around the world, they are also very likely to be linked together to offer the necessary global usage of those knowledge.

As a motivating example, we consider one of the case studies of the NeOn project[1] [Dzbor *et al.*, 2005] – a fishery case study in the Food and Agriculture Organization (FAO) of the United Nations (UN). This case study aims to improve the interoperability of FAO information systems in the fishery domain, integrating and using networked ontologies and related methods and technologies, by creating and maintaining distributed ontologies (and ontology mappings) in the fishery domain. FAO has large sets of fishery ontology data with the following features and requirements:

1. *Networking.* The ontology data sets in FAO are intensively interconnected by different subjects, languages, countries and other geopolitical aspects in a secure networked environment with clear boundary for information hiding and encapsulation.
2. *Dynamics.* In FAO, the ontologies are large, interconnected and changing over time. Therefore, an approach that can handle ontology data in a dynamic way with change monitoring and propagation is required.
3. *Distribution.* Because FAO has many branches around the world, the ontologies in FAO are distributed rather than centralized, which arises challenges in loose coupling and autonomous management.
4. *Reasoning.* FAO ontology data needs reasoning support with high efficiency and scalability to handle the ontologies, which usually consist of both terminologies and assertional data in FAO fishery case study.

We argue that this scenario is common to typical semantic applications, especially for ontology management in big international organizations. However, current technologies often have difficulties in handling ontological data in the aforementioned scenario. Typical problems against the requirements mentioned above are:

1. Traditional ontology formalisms, e.g. Frame System or Description Logics, are designed for centralized ontologies rather than decentralized ones. Furthermore, most ontology management systems do not support processing large instance data represented in the form of ontologies in the decentralized scenario.
2. In a scenario with interconnected ontology modules, ideally, when one ontology module is updated, the depending ontology modules should be updated as well to reflect the changes. However, few current technologies can support such dynamic automatic updating.
3. What is still missing is a *principled* approach to support distributed ontologies, where the individual ontology modules are physically distributed, loosely coupled and autonomously managed.
4. Some reasoners are able to support either local TBox reasoning (e.g. FaCT++ [Tsarkov and Horrocks, 2004]) or distributed TBox reasoning (e.g. Drago [Serafini and Tamilin, 2005] and Pellet [Sirin and Parsia, 2004a]), while others are good at ABox reasoning (e.g. KAON2 [Motik and Sattler, 2006]). However, handling both TBox and ABox in a distributed, efficient and scalable

[1] `http://www.neon-project.org`

way for modular ontologies is a challenging task for reasoners and still not satisfactory.

To manage knowledge generated and maintained in such distributed settings, we need knowledge representation formalisms and tools to meet the following challenge: How to properly manage multiple networked, distributed, dynamic ontologies and provide corresponding reasoning support? In this paper, we investigate approaches to represent and exploit such networked ontologies, considering the recent advances in formalisms for modular ontologies and distributed reasoning techniques.

In the following sections, we firstly introduce some preliminaries of description logics in Section 2, we analyze requirements of networked ontologies and comparison criteria for different formalisms in Section 3. We compare several formalisms for the need of networked ontologies in Section 4 and discuss necessary future extensions for such formalisms in Section 6. We summarize the paper and outline future work in Section 7.

2 Preliminaries of Description Logics

The formalisms for modular ontologies studied in this paper all mainly aim to handle ontology modules based on Description Logics (DLs) as the underlying logical formalism. Therefore, we here introduce basic preliminaries of DLs to allow a better understanding of the formalisms introduced in Section 4.

Syntax. Given $\mathcal{R}$ as a finite set of transitive and inclusion axioms with normal role names N_R, a *$\mathcal{SHOIQ}$-role* is either some $R \in N_R$ or an *inverse role* R^- for $R \in N_R$. $\mathsf{Trans}(R)$ and $R \sqsubseteq S$ represent the transitive and inclusion axioms, respectively, where R and S are roles. A simple role is a *$\mathcal{SHOIQ}$-role* that neither its sub-roles nor itself is transitive. Let N_C be a set of *concept names* , the set of $\mathcal{SHOIQ}$-concepts is the minimal set such that every concept $C \in N_C$ is a $\mathcal{SHOIQ}$-concept and for C and D are $\mathcal{SHOIQ}$-concepts, R is a role, S a simple role and n a positive integer, then $(\neg C)$, $\{o_1, o_2, ...\}$, $(C \sqcap D)$, $(C \sqcup D)$, $(\exists R.C)$, $(\forall R.C)$, $(\leqslant nSC)$ and $(\geqslant nSC)$ are also $\mathcal{SHOIQ}$-concepts. Therefore we have a knowledge base $\mathcal{KB}$ is a triple $(\mathcal{R}, \mathcal{T}, \mathcal{A})$ where (i) a TBox $\mathcal{T}$ is a finite set of axioms representing the concept inclusions with the form $C \sqsubseteq D$; (ii) an ABox $\mathcal{A}$ is a finite set of axioms with the form $C(x)$, $R(x, y)$ that consists equality-relations x is (un)equal y. $\mathcal{A}$ is *simple* if $C(a) \in \mathcal{A}$ implies that C is a *concept literal* that is either a concept name or the negation thereof.

Semantics. The semantics of $\mathcal{KB}$ is given by the interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ that consists of a non-empty set $\Delta^{\mathcal{I}}$ (the domain of $mathcalI$) and the function $\cdot^{\mathcal{I}}$ in the Table 1 [Horrocks *et al.*, 2003]. The satisfiability checking of $\mathcal{KB}$ in expressive DLs is performed by reducing the subsumption, and the reasoning over TBox and role hierarchy can be reduced to reasoning over only role hierarchy [Horrocks and Patel-Schneider, 1999]. The interpretation $\mathcal{I}$ is the model of $\mathcal{R}$ and $\mathcal{T}$ if for each $R \sqsubseteq S \in \mathcal{R}$, $R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$ and for each $C \sqsubseteq D \in \mathcal{T}$, $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$.

Table 1: Semantics of $\mathcal{SHOIQ} - \mathcal{KB}$

Interpretation of Concepts
$(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \backslash C^{\mathcal{I}}, \{o_1, o_2, ...\}^{\mathcal{I}} = \{o_1^{\mathcal{I}}, o_2^{\mathcal{I}}, ...\}$
$(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}, (C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$
$(\exists R.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid R^{\mathcal{I}}(x, C) \neq \emptyset\}$
$(\forall R.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid R^{\mathcal{I}}(x, \neg C) = \emptyset\}$
$(\leqslant nS.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \mathbf{N}R^{\mathcal{I}}(x, C) \leqslant n\}$
$(\geqslant nS.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \mathbf{N}R^{\mathcal{I}}(x, C) \geqslant n\}$
Interpretation of Axioms
$(C \sqsubseteq D)^{\mathcal{I}} = C^{\mathcal{I}} \subseteq D^{\mathcal{I}}, (R \sqsubseteq S)^{\mathcal{I}} = R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$
$(\mathsf{Trans}(R))^{\mathcal{I}}$
$= \{\forall x, y, z \in \Delta^{\mathcal{I}} \mid R^{\mathcal{I}}(x, y) \cap R^{\mathcal{I}}(y, z) \rightarrow R^{\mathcal{I}}(x, z)\}$
$\mathbf{N}R$ is the number restriction of a set R and $R^{\mathcal{I}}(x, C)$ is defined as $\{y \mid \langle x, y\rangle \in R^{\mathcal{I}} and y \in C^{\mathcal{I}}\}$.

3 Criteria for Survey

In this section, we discuss requirements for modular ontologies languages. These will serve as criteria for the investigation of the different formalisms for modular ontologies along the four dimensions introduced in Section 1.

Networking

- *Encapsulation.* In the networked ontology setting, each ontology module may represent knowledge in a subset of the domain in question. Those modules are usually autonomously created and maintained, while may also be inter-connected to form a larger knowledge base. In other words, such ontology modules can *encapsulate* knowledge subdomains.
- *Reusability (Inheritance).* Ontologies are very likely to be reused. Thus, formalisms for modular ontologies should also support managing different modules and identifying module dependencies. In particular, ontology modules should be also transitively reusable, that is, if a module A uses module B, and module B uses module C, then apparently, module A should use module C.
- *Trust and Security.* Many applications call for controllable access of knowledge due to copyright, privacy or safety concerns. Formalisms for modular ontologies should also support trust and security features to ensure safe creation and usage of ontology modules. For example, to enable multi-user access to ontologies, the language supported by the formalism should secure the sessions by explicitly defining the rights for accessing and editing for each module.

Dynamics

- *Networked Ontology Dynamics.* The dynamic role of networked ontologies reflects the importance of monitoring and propagating the ontology changing events, especially when some ontology modules are changed and updated. Such a requirement is closely related to ontology versioning and evolution; on the other hand, it is more focused on the identifying and updating of module changes rather than managing ontology versions.

Distribution

- *Loose Coupling.*

  In [Stuckenschmidt and Klein, 2003], Stuckenschmidt and colleagues argue that although the existence of overlaps between the modules is uncertain, the ontologies still should be compatible while being composed. In other words, formalism for modular ontologies should be able to handle modules that are not disjoint with each other. For example, assume concept A is the only possible common concept in modules M_1 and M_2, if A exists, the M_1 and the M_2 are loose coupled, and A is the shared concept. Obviously a usable system should be not affected by the existence of A while handling of the two modules because it is normal that two ontology modules are not fully disjoint with each other.

- *Self-Containment.* An ontology module is likely to be related to other modules, therefore a formalism for modular ontologies should support localized semantics that includes the global information about module dependencies. In other words, the module should be self-contained while can also be semantically interconnected to other modules. A core advantage of self-containment is that reasoning tasks involving only local knowledge of a module do not need the access to other modules [Stuckenschmidt and Klein, 2003].

Reasoning

- *Complexity and Scalability.* Processing modular ontologies is typically more challenging than reasoning with single ontologies. Thus, a desirable formalism for modular ontologies should provide language features and reasoning procedure that are efficient and scalable to large terminologies and knowledge instance set.

- *Reasoning Support for Terminological and Assertional Knowledge.* An important criterion for judging a formalisms for modular ontologies is whether it supports both T-Box and ABox reasoning and querying. Support for modular ABoxes is particularly important since our motivation applications involve knowledge that represented in large and distributed instance sets.

Language expressivity of the formalisms for modular ontologies include the following set of criteria.

- *Module Correspondence.* As stated in the functionality criteria, the correspondences between different modules are essential. For example, if a ontology module that describes a university department is inherited from a university ontology, we can simply state like: $DepartmentModule\ isInheritedFrom\ UniversityModule$. It should also necessary to discover the overlaps between the modules.

- *Class Subsumption.* Having subsumptions relations between classes in different modules is one of the most needed features in modular ontologies. For example, $MasterStudent$ in the university module is a subclass of $Student$ in social role module.

- *Class Interconnection.* A class in a module may have role that relates to a foreign module. For example, $PhDStudent$ in the university module may live in a $City$ included in the geographic module.

- *Role Subsumption.* It is used to allow the subsumption relationship between the local and the foreign role.

- *Role Transitivity.* A foreign transitive role may be used in a module, e.g., the module A reuses the property $hasSuccessor$ which is defined in the module B.

- *Role Inversion.* It is required to specify inverse relations between local and foreign roles.

- *Individual Correspondence.* It is requires to specify that some individuals are be related to individuals in other modules.

Giving the above set of criteria on language functionality and expressivity, we will analyze candidate formalisms in details in the next section.

4 Candidates of Formalisms for Modular Ontologies

In this section we introduce four major formalisms for modular ontologies that has been established in recent years in details.

4.1 Modularization with OWL Import

The OWL ontology language provides limited support to modularize ontologies: an ontology document – identified via its ontology URI – can be imported by another document using the *owl:imports* statement. The semantics of this import statement is that all definitions of the imported ontology (module) are logical part of the importing ontology (module) as if they were defined in the importing ontology directly. Thus, they are forced to share a classical DL semantics, i.e., a global model-theoretical semantics. It should be noted that such a importing is directed: only the importing ontology is affected by the import statement; it is also transitive: if ontology A imports ontology B, and ontology B imports ontology C, then ontology A also imports ontology C. Cyclic imports are also allowed (e.g. A *owl:imports* B, B *owl:imports* A).

Elements of the importing and imported ontology module can be related to each other using legal primitives available in OWL. Typically these relation definitions are part of the importing ontology module.

The OWL imports functionality provides no partial importing of modules, thus it is up to the user to decide the proper level of granularity of ontology modules.

4.2 Distributed Description Logics

Distributed Description Logics (DDL) [Borgida and Serafini, 2002] adopt a linking mechanism. The semantic linkings between modules M_i and M_j are represented by cross-module *Bridge Rules* "INTO" and "ONTO" axioms in one of the following forms:

- INTO: $i : \phi \xrightarrow{\sqsubseteq} j : \psi$, with semantics: $r_{ij}(\phi^{I_i}) \subseteq \psi^{I_j}$

- ONTO: $i : \phi \xrightarrow{\sqsupseteq} j : \psi$,with semantics: $r_{ij}(\phi^{I_i}) \supseteq \psi^{I_j}$

where I_i and I_j are local interpretations of M_i and M_j, respectively, ϕ, ψ are formulae, r_{ij} (called *domain relation*) is a relation that represents as an interpretation of B_{ij}. We will only discuss bridge rules among concepts since it is the only

case that have reported reasoning support. In DDL, the *distributed knowledge base (D-KB)*, $\mathfrak{D} = \langle \{\mathcal{KB}_i\}_{i\in I}, \mathfrak{B}\rangle$ consists a set of knowledge bases $\{\mathcal{KB}_i\}_{i\in j}$ and bridge rules $\mathfrak{B} = \{\mathfrak{B}_{ij}\}_{\{i\neq j\}\in I}$ that represent the correspondences between them. This is the distributed interpretation in DDL.

DDL bridge rules between concepts covers one of the most important scenarios in modular ontologies. They are intended to simulate concept inclusion with a special type of roles. However, a bridge rule cannot be read as concept subsumption, such as $i : A \sqsubseteq j : B$. Instead, it must be read as a classic DL axiom in the following way [Borgida and Serafini, 2002]:

- $i : A \xrightarrow{\sqsubseteq} j : B \Rightarrow (i : A) \sqsubseteq \forall R_{ij}.(j : B)$
- $i : A \xrightarrow{\sqsupseteq} j : B \Rightarrow (j : B) \sqsubseteq \exists R_{ij}^{-}.(i : A)$

where R_{ij} is a new role representing correspondences B_{ij} between L_i and L_j.

Such relations have semantic differences with respect to concept inclusion (interpreted in classic DLs as subset relations between concept interpretations, e.g. $A^{\mathcal{I}_i} \subseteq B^{\mathcal{I}_j}$) in several ways. For example, empty domain relation r_{ij} is allowed in the original DDL proposal [Borgida and Serafini, 2002], while GCIs between satisfiable concepts enforce restrictions on non-empty interpretations. Arbitrary domain relations may not preserve concept unsatisfiability among different modules which may result in some reasoning difficulties [Bao *et al.*, 2006c]. Further more, while subset relations (between concept interpretations) is transitive, DDL domain relations are not transitive, therefore bridge rules cannot be *transitively reused* by multiple modules. Those problems are recently recognized in several papers [Bao *et al.*, 2006b; 2006c; Stuckenschmidt *et al.*, 2005; Serafini *et al.*, 2005] and it is proposed that arbitrary domain relations should be avoided. For example, domain relations should be one-to-one [Serafini *et al.*, 2005; Bao *et al.*, 2006c] and non-empty [Stuckenschmidt *et al.*, 2005].

The requirements of practical applications raised in the previous section are not fully satisfied by the expressivity of DDL. For example, inter-module role correspondences, which are important to present the relations between concepts in different modules, are not supported in DDL: assume an concept $PhDStudent$ is included in one ontology module and another concept $Thesis$ is include in another ontology module, we cannot define $PhDStudent \sqsubseteq \exists writes.Thesis$ in DDL, where $writes$ is a *inter-module* role.

4.3 Integrity and Change of Modular Terminologies in DDL

Influenced by DDL semantics, [Stuckenschmidt and Klein, 2003] adopt a view-based information integration approach to express relationships between ontology modules. In particular, in this approach ontology modules are connected by correspondences between conjunctive queries. This way of connecting modules is more expressive than simple one-to-one mappings between concept names but less expressive than the logical language used to describe concepts.

[Stuckenschmidt and Klein, 2003] defines an ontology module – abstracted from a particular ontology language – as a triple $M = (C, \mathcal{R}, \mathcal{O})$, where C is a set of concept definitions, $\mathcal{R}$ is a set of relation definitions and $\mathcal{O}$ is a set of object definitions. A conjunctive query Q over an ontology module $M = (C, \mathcal{R}, \mathcal{O})$ is defined as an expression of the form $q_1 \wedge ... \wedge q_m$, where q_i is a query term of the form $C(x)$, $R(x, y)$ or $x = y$, C and R concept and role names, respectively, and x and y are either variable or object names.

In a modular terminology it is possible to use conjunctive queries to define concepts in one module in terms of a query over another module. For this purpose, the set of concept definitions C is divided into two disjoint sets of internally and externally defined concepts C_I and C_E, respectively, with $C = C_I \cup C_E, C_I \cap C_E = \emptyset$. An *internal concept* definition is specified using regular description logics based concept expressions with the form of $C \sqsubseteq D$ or $C \equiv D$, where C and D are atomic and complex concepts, respectively. An *external concept* definition is an axiom of the form $C \equiv M : Q$, where M is a module and Q is a conjunctive query over M. It is assumed that such queries can be later reduced to complex concept descriptions using the query-rollup techniques from [Horrocks and Tessaris, 2000] in order to be able to rely on standard reasoning techniques. A modular ontology is then simply defined as a set of modules that are connected by external concept definitions. The semantics of these modules is defined using the notion of a distributed interpretation as introduced in Section 4.2.

Although the definition of a module, in its abstract form shown above, may allow arbitrary concept, relation and object definitions, only concept definitions is studied in [Stuckenschmidt and Klein, 2003]. This is due to the focus of the approach to improve terminological reasoning with modular ontologies by pre-compiling implied subsumption relations. In that sense it can be seen as a restricted form of DDLs that enables improved efficiency for special TBox reasoning tasks.

4.4 $\mathcal{E}$-Connection

While DDL allows only one type of domain relations, the $\mathcal{E}$-connection approach allows multiple "connection" relations between two modules, such as $liveIn$ and $bornIn$ between $2 : Fishkind$ and $1 : Region$. $\mathcal{E}$-connections between DLs [Kutz *et al.*, 2003; Grau *et al.*, 2004b] restrict the local domains of the $\mathcal{E}$-connected ontology modules to be disjoint. Roles are divided into disjoint sets of *local roles* (connecting concepts in one module) and *links* (connecting inter-module concepts).

Formally, given ontology modules $\{L_i\}$, an (one-way binary) link $E \in \mathcal{E}_{ij}$, where $\mathcal{E}_{ij}, i \neq j$ is the set of all links from the module i to the module j, can be used to construct a concept in module i, with the syntax and semantics specified as follows:

- $\exists E.(j : C) : \{x \in \Delta_i | \exists y \in \Delta_j, (x, y) \in E^M, y \in C^M\}$
- $\forall E.(j : C) : \{x \in \Delta_i | \forall y \in \Delta_j, (x, y) \in E^M \rightarrow y \in C^M\}\}$
- $\leq nE.(j : C) : \{x \in \Delta_i | \#(\{y \in \Delta_j | (x, y) \in E^M, y \in C^M\}) \leq n\}$
- $\geq nE.(j : C) : \{x \in \Delta_i | \#(\{y \in \Delta_j | (x, y) \in E^M, y \in C^M\}) \geq n\}$

where $M = \langle \{m_i\}, \{E^M\}_{E \in \mathcal{E}_{ij}} \rangle$ is a model of the $\mathcal{E}$-connected ontology, m_i is the local model of L_i; C is a concept in L_j, with interpretation $C^M = C^{m_j}$; $E^M \subseteq \Delta_i \times \Delta_j$ is the interpretation of a $\mathcal{E}$-connection E.

However, the applicability of $\mathcal{E}$-connections in our setting is also limited by the need to ensure that the term set as while as local domains of each ontology module are strictly disjoint:

- The requirement for terminology disjointness and local domain disjointness in $\mathcal{E}$-connections are too strict for many applications. For example, $\mathcal{E}$-connections can not be used in the case that a user want to refine knowledge in an existing module with a new module (e.g. $i : Fish$ is less general than $j : Animal$, where $j : Animal$ is a concept in an existing module and $i : Fish$ is a new concept extended from $j : Animal$).
- To enforce local domain disjointness, a concept cannot be declared as subclass of another concept in a foreign module thereby ruling out the possibility of asserting inter-module subsumption and the general support for transitive usability; a property cannot be declared as subrelation of a foreign property; neither foreign classes nor foreign properties can be instantiated; cross-module concept conjunction or disjunction are also illegal.
- $\mathcal{E}$-connected ontologies have difficulties to be used with OWL importing mechanism, since importing may actually "decouple" the combination and result in inconsistency [Grau, 2005].
- $\mathcal{E}$-connected ontologies do not allow a same term be used as both a link name and a local role name, nor role inclusions between links and roles, while such features are widely required in practice [Grau, 2005]. The"punning" approach [Grau, 2005], where a same name can have different interpretations, is rather as a syntactical sugar than a semantic solution to such problems.

4.5 P-DL

P-DL, Package-based Description Logics [Bao *et al.*, 2006d], uses importing relations to connect local models. In contrast to OWL, which forces the model of an imported ontology be completely embedded in a global model, the P-DL importing relation is *partial* in that only commonly shared terms are interpreted in the overlapping part of local models. The semantics of P-DL is given as the follows: the *image domain relation* between local interpretations $\mathcal{I}_i$ and $\mathcal{I}_j$ (of package P_i and P_j) is $r_{ij} \subseteq \Delta_i \times \Delta_j$. P-DL importing relation is:

- one-to-one: for any $x \in \Delta_i$, there is at most one $y \in \Delta_j$, such that $(x, y) \in r_{ij}$, and vice versa.
- compositionally consistent: $r_{ij} = r_{ik} \circ r_{jk}$, where $\circ$ denotes function composition. Therefore, semantic relations between terms in i and terms in k can be inferred even if k doesn't directly import terms from i.

Thus, a P-DL model is a virtual model constructed from partially overlapping local models by merging "shared" individuals.

P-DL also supports selective knowledge sharing by associating ontology terms and axioms with "scope limitation modifiers (SLM)". A SLM controls the visibility of the corresponding term or axiom to entities on the web, in particular, to other packages. The scope limitation modifier of a term or an axiom t_K in package K is a boolean function $f(p, t_K)$, where p is a URI of an entity, the entity identified by p can access t_K iff $f(p, t) = true$. For example, some representative SLMs can be defined as follows:

- $\forall p, public(p, t) := true$, means t is accessible everywhere.
- $\forall p, private(p, t) := (t \in p)$, means t is visible only to its home package.

P-DL semantics ensure that distributed reasoning with a modular ontology will yield the same conclusion as that obtained by a classical reasoning process applied to an integration of the respective ontology modules [Bao *et al.*, 2006c]. However, reported result in [Bao *et al.*, 2006a] only supports reasoning in P-DL as extensions of the $\mathcal{ALC}$ DL. Reasoning algorithms for more expressive P-DL TBox, as well as for ABox reasoning, still need to be investigated.

5 Comparison

In this section compare the existing formalisms for modular ontologies, then we explain the results based on the comparison criteria given in Section 3.

In Table 2, we summarize the comparison of different formalisms against the requirements introduced in Section 3, then we compare the expressivity of the popular formalisms in Table 3. In this section, the integrity and change aspects related to modular ontologies investigated in [Stuckenschmidt and Klein, 2003] will be denoted as "DDL-IC", since it follows the semantics of DDL.

We now explain the survey result regarding each requirement in detail.

- *Encapsulation.* All formalisms listed above support knowledge encapsulation at different level. OWL DL provides a basic *owl:import* primitive to import foreign ontologies without encapsulated modules, hence OWL DL partially supports this functionality; DDL, DDL-IC, $\mathcal{E}$-connection and P-DL allow a large knowledge base to be represented by a set of ontology modules each capturing a subset of the domain of interest, thus provide the support for knowledge encapsulation.
- *Reusability*. OWL DL ontologies has only limited reusability. In particular, it is difficult to partially reuse ontologies designed by others. DDL, DDL-IC and P-DL establish good reusability via well-defined encapsulation. The reusability of $\mathcal{E}$-connection is marked with a "*" symbol because the experiment in [Seidenberg and Rector, 2006] shows for some knowledge bases, $\mathcal{E}$-connection is not able to generate reusable ontology modules. The lack of support for inter-module concept inclusion presents strong restriction in reusing $\mathcal{E}$-Connection modules. On the other hand, a unified import/export interface is not formally defined like code modularization in software engineering, although P-DL provides packages to partially enable unified import/export without the definition of interface (i.e. the "+" symbol in the Table 2 means this additional feature).

	OWL DL	DDL	DDL-IC	P-DL	$\mathcal{E}$-Connection
Encapsulation	Partial	Yes	Yes	Yes	Yes
Reusability	Fair	Good	Good	Good$^+$	Good*
Trust and Security	No	No	No	Partial	No
Ontology Dynamics	Yes	No	Yes	No	No
Loose Coupling	No	Yes	Yes$^+$	Yes	Yes*
Self-Containment	Partial	Yes	Yes	Partial	Yes
Scalability	Low	Fair	Fair	Fair	Low
Reasoning Support	T and A	T and Partial A	T	T	T

Table 2: Comparison on language functionality. "T" means this formalism supports terminological (TBox) reasoning and "A" stands for the assertional (ABox) reasoning support. We refer reader to the corresponding detailed analysis of this table for explanations of the "+" and "*" symbols.

	OWL DL	DDL	DDL-IC	P-DL	$\mathcal{E}$-Connection
Module Correspondence.	No	No	Partial	Partial	No
Class Subsumption.	Yes	Yes	Yes	Yes	No
Class Interconnection.	Yes	Yes	Yes	Yes	Yes
Role Subsumption.	Yes	Yes	Yes	Yes	No
Role Transitivity.	Yes	No	Yes	Yes	Partial
Role Inversion.	Yes	No	Yes	Yes	No
Individual Correspondence.	Yes	Yes	Yes	No	No

Table 3: Comparison on expressivity

- *Trust and Security*. Bao et.al. proved that it is possible to integrate authorization information to the OWL concepts to guarantee the secure access and edit of ontology modules [Bao *et al.*, 2006c]. To the best of our knowledge, there is no other reported formalisms support this functionality.
- *Ontology Dynamics*. DDL-IC developed a mechanism to monitor the changes of modular ontologies [Stuckenschmidt and Klein, 2003], which is still missing in other existing formalisms.
- *Loose Coupling*. In principle, the formalisms for modular ontologies normally provide the support of dealing with loosely coupled ontologies, while Stuckenschmidt and colleagues explicitly argued its importance and deployment in DDL-IC [Stuckenschmidt and Klein, 2003] (with plus symbol in the Table 2). Loosely coupled modules are not able to be $\mathcal{E}$-Connected if their domains are not disjoint, therefore $\mathcal{E}$-Connection is again marked with symbol "*" in the Table 2.
- *Self-containment*. Due to the lack of localized semantics, OWL DL does not well support knowledge self-containment. In particular, reasoning in an OWL ontology requires the integration of all directly or indirectly imported ontologies of the given ontology. DDL-IC [Stuckenschmidt and Klein, 2003] introduces the self-containment functionality based on traditional DDL, while $\mathcal{E}$-connections requires strict separation of knowledge terminologies of ontology modules as well as their local interpretation domains [Grau, 2005]. P-DL can maintain the autonomy of individual modules; however, since P-DL adopts a partial semantic importing approach, reasoning in a P-DL ontology may also depend on its imported ontologies.
- *Scalability*. The worst time complexity of the four formalisms studied in this paper are all exponential[Horrocks and Sattler, 2005; Bao *et al.*, 2006a; Grau, 2005; Serafini and Tamilin, 2005], thereby we mainly discuss the scalability of these formalisms in the distributed environment. DDL, DDL-IC and P-DL support reasoning in a distributed setting in which still ontology module can be kept strictly separate, thus the integration of component ontology modules is avoided to obtain higher scalability in handling large ontologies. On the other hand, the $\mathcal{E}$-connection reasoner Pellet [Sirin and Parsia, 2004b] adopts "coloring" but not physical separation approach in reasoning with multiple ontology modules, hence requires implicit ontology integration to a single location, which may deteriorate its scalability.
- *Reasoning Support*. Reasoning support for OWL DL has been successfully implemented in several highly optimized reasoners, such as FaCT++[Tsarkov and Horrocks, 2004], Pellet[Sirin and Parsia, 2004a] and KAON2; in particular KAON2 is optimized for reasoning with large ABoxes. DDL recently supported large ABox reasoning in a limited form [Serafini and Tamilin, 2006]. Other formalisms have reported support for TBoxes [Bao *et al.*, 2006a; Grau *et al.*, 2004a] only.

Expressivity comparison of those formalisms is explained as the following:

- DDL-IC defines modules may be related to other modules by discovering hidden information before the integration, while other languages do not define correspondences between modules.
- All formalisms support class interconnections across modules. OWL DL allows arbitrarily complex relations between concepts in different ontologies (i.e. concept

connections have the same expressivity as that of the local ontology language in each module). On the other hand, other formalism restrict the expressivity of concept connections to obtain both localized semantics and decidability. DDL, DDL-IC support concept subsumptions. $\mathcal{E}$-connection does not allow cross-module subsumption relationships, but allows two concepts being connected by links. P-DL supports both inter-module concept subsumptions and inter-module concept connections by roles.

- Role subsumption is provided by DDL, DDL-IC and OWL DL. $\mathcal{E}$-connection doesn't allow a same name being shared by links and roles, thus, role relationships between the modules may lead to a same name can have different interpretations (such roles is called generalized links[Parsia and Grau, 2005] in $\mathcal{E}$-connections). Role transitivity and inversion are supported by DDL, DDL-IC. Reported P-DL formalism [Bao *et al.*, 2006b] does not allow role name importing hence does not support inter-module role subsumption, inversion and the reuse of transitive roles.
- OWL DL supports simple individual correspondence by predicate *owl:sameIndividualAs*. Among other formalisms, only DDL has investigated individual correspondence between ontology modules [Serafini and Tamilin, 2006]. $\mathcal{E}$-connections does not allow cross-module individual correspondence since local domains of ontology modules are strictly disjoint. Such a feature is also missing in reported P-DL formalism (which only allows concept importing among ontology modules). DDL-IC allows a view with no variable being defined, which may be used to establish individual correspondence between ontology modules.

6 Discussion

The survey in the previous sections shows that existing formalisms are not satisfactory in representing modular ontologies that is needed by our motivated applications, i.e. the FAO fishery case study in NeOn project.

First of all, a commonly accepted definition of "What is a good ontology module?" is still missing. Secondly, an efficient and scalable reasoning approach for large ABox data is currently not well provided by existing formalisms for modular ontologies. Thirdly, most of the existing approaches can not support trust and authority requirements. Finally, on the aspects of managing the dynamics of ontologies, the approach proposed by Stuckenschmidt and colleagues [Stuckenschmidt and Klein, 2004] is able to monitor the changes and integration of the ontologies, which is still missing in other formalisms.

Existing formalisms are also limited in expressivity. Most formalisms provide means to deal with the terminologies that are classes in ontologies. However, mechanisms to handling other ontological entities, such as roles and individuals, is not supported by $\mathcal{E}$-connection or P-DL. On the other hand, interconnections and relationships between modules, which is needed by many applications using modular ontologies, is currently not well-defined and lacks implementations.

Existing formalisms require further extensions in order to serve as successful formalisms for modular ontologies as discussed in Section 3. Practical reasoning support for expressive formalisms for modular ontologies are also to be investigated.

7 Conclusions and Future Work

In this paper we studied different formalisms for modular ontologies. We proposed a set of criteria to compare formalisms for modular ontologies based on the requirements of networked ontology applications. We compared several formalisms for modular ontologies against these requirements. The comparison result suggests that no existing approach could satisfactorily meet the requirement of our networked ontology applications. Several possible extension of existing formalisms were discussed.

Work in progress includes the development of a networked ontology based formalism that meets the requirements raised in the Section 3, as well as efficient and scalable reasoning support for such a formalism that is able to handle large distributed ontology terminologies and instance data sets.

Acknowledgments

Research reported in this paper is partially support by the EU in the IST project NeOn (IST-2006-027595, `http://www.neon-project.org/`. We appreciate the fruitful discussion and comments from our colleagues).

References

[Bao *et al.*, 2006a] Jie Bao, Doina Caragea, and Vasant Honavar. A distributed tableau algorithm for package-based description logics. In *the 2nd International Workshop On Context Representation And Reasoning (CRR 2006), co-located with ECAI 2006*. 2006.

[Bao *et al.*, 2006b] Jie Bao, Doina Caragea, and Vasant Honavar. Modular ontologies - a formal investigation of semantics and expressivity. In *R. Mizoguchi, Z. Shi, and F. Giunchiglia (Eds.): Asian Semantic Web Conference 2006, LNCS 4185*, pages 616–631, 2006.

[Bao *et al.*, 2006c] Jie Bao, Doina Caragea, and Vasant Honavar. On the semantics of linking and importing in modular ontologies. In *I. Cruz et al. (Eds.): ISWC 2006, LNCS 4273 (In Press)*, pages 72–86. 2006.

[Bao *et al.*, 2006d] Jie Bao, Doina Caragea, and Vasant Honavar. Towards collaborative environments for ontology construction and sharing. In *International Symposium on Collaborative Technologies and Systems (CTS 2006)*, pages 99–108. IEEE Press, 2006.

[Borgida and Serafini, 2002] Alexander Borgida and Luciano Serafini. Distributed description logics: Directed domain correspondences in federated information sources. In *CoopIS/DOA/ODBASE*, pages 36–53, 2002.

[Dzbor *et al.*, 2005] Martin Dzbor, Enrico Motta, Rudi Studer, York Sure, Peter Haase, Asuncin Gmez-Prez, Richard Benjamins, and Walter Waterfeld. Neon - lifecycle support for networked ontologies. In *Proceedings of 2nd European Workshop on the Integration of Knowledge, Semantic and Digital Media Technologies (EWIMT-2005)*, pages 451–452, London, UK, NOV 2005. IEE.

[Grau *et al.*, 2004a] Bernardo Cuenca Grau, Bijan Parsia, and Evren Sirin. Tableau algorithms for e-connections of description logics. Technical report, University of Maryland Institute for Advanced Computer Studies (UMIACS), TR 2004-72, 2004.

[Grau *et al.*, 2004b] Bernardo Cuenca Grau, Bijan Parsia, and Evren Sirin. Working with multiple ontologies on the semantic web. In *International Semantic Web Conference*, pages 620–634, 2004.

[Grau, 2005] Bernardo Cuenca Grau. *Combination and Integration of Ontologies on the Semantic Web*. PhD thesis, Dpto. de Informatica, Universitat de Valencia, Spain, 2005.

[Horrocks and Patel-Schneider, 1999] Ian Horrocks and Peter F. Patel-Schneider. Optimizing description logic subsumption. *Journal of Logic and Computation*, 9(3):267–293, 1999.

[Horrocks and Sattler, 2005] Ian Horrocks and Ulrike Sattler. A Tableaux Decision Procedure for SHOIQ. In *IJCAI*, pages 448–453, 2005.

[Horrocks and Tessaris, 2000] I. Horrocks and S. Tessaris. A conjunctive query language for description logic aboxes. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 399–404. AAAI Press / The MIT Press, 2000.

[Horrocks *et al.*, 2003] Ian Horrocks, Peter F. Patel-Schneider, and Frank van Harmelen. From $\mathcal{SHIQ}$ and RDF to OWL: The making of a web ontology language. *J. of Web Semantics*, 1(1):7–26, 2003.

[Kutz *et al.*, 2003] Oliver Kutz, Carsten Lutz, Frank Wolter, and Michael Zakharyaschev. E-connections of description logics. In *Description Logics Workshop, CEUR-WS Vol 81*, 2003.

[Motik and Sattler, 2006] Boris Motik and Ulrike Sattler. A comparison of reasoning techniques for querying large description logic aboxes. In Miki Hermann and Andrei Voronkov, editors, *Proc. of the 13th Int. Conf. on Logic for Programming Artificial Intelligence and Reasoning (LPAR 2006)*, LNCS, Phnom Penh, Cambodia, 2006. Springer. To appear.

[Parsia and Grau, 2005] Bijan Parsia and Bernardo Cuenca Grau. Generalized link properties for expressive epsilon-connections of description logics. In *AAAI*, pages 657–662, 2005.

[Seidenberg and Rector, 2006] Julian Seidenberg and Alan Rector. Web ontology segmentation: Analysis, classification and use. In *Proceedings of the World Wide Web Conference (WWW)*, Edinburgh, June 2006.

[Serafini and Tamilin, 2005] Luciano Serafini and Andrei Tamilin. Drago: Distributed reasoning architecture for the semantic web. In *European Semantic Web Conference - ESWC*, pages 361–376, 2005.

[Serafini and Tamilin, 2006] Luciano Serafini and Andrei Tamilin. Instance retrieval over a set of heterogeneous ontologies. In *the 2nd International Workshop On Context Representation And Reasoning (CRR 2006), co-located with ECAI 2006*. 2006.

[Serafini *et al.*, 2005] L. Serafini, H. Stuckenschmidt, and H. Wache. A formal investigation of mapping languages for terminological knowledge. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence - IJCAI05*, Edinburgh, UK, August 2005.

[Sirin and Parsia, 2004a] Evren Sirin and Bijan Parsia. Pellet: An OWL DL Reasoner. In *Description Logics Workshop*, 2004.

[Sirin and Parsia, 2004b] Evren Sirin and Bijan Parsia. Pellet: An owl dl reasoner. In *Description Logics*, 2004.

[Stuckenschmidt and Klein, 2003] Heiner Stuckenschmidt and Michel C. A. Klein. Integrity and change in modular ontologies. In *IJCAI*, pages 900–908, 2003.

[Stuckenschmidt and Klein, 2004] Heiner Stuckenschmidt and Michel C. A. Klein. Structure-based partitioning of large concept hierarchies. In *International Semantic Web Conference*, pages 289–303, 2004.

[Stuckenschmidt *et al.*, 2005] Heiner Stuckenschmidt, Luciano Serafini, and Holger Wache. Reasoning about ontology mappings. Technical report, Department for Mathematics and Computer Science, University of Mannheim ; TR-2005-011, 2005.

[Tsarkov and Horrocks, 2004] Dmitry Tsarkov and Ian Horrocks. Efficient reasoning with range and domain constraints. In *Description Logics*, 2004. FaCT++.