Published in final edited form as:
Integr Comput Aided Eng. 2009 August ; 16(3): 225–242. doi:10.3233/ICA-2009-0315.

Survey of modular ontology techniques and their applications in the biomedical domain

Jyotishman Pathak[*], **Thomas M. Johnson**, and **Christopher G. Chute**
Division of Biomedical Statistics and Informatics, Mayo Clinic, Rochester, MN, USA

Abstract

In the past several years, various ontologies and terminologies such as the Gene Ontology have been developed to enable interoperability across multiple diverse medical information systems. They provide a standard way of representing terms and concepts thereby supporting easy transmission and interpretation of data for various applications. However, with their growing utilization, not only has the number of available ontologies increased considerably, but they are also becoming larger and more complex to manage. Toward this end, a growing body of work is emerging in the area of modular ontologies where the emphasis is on either extracting and managing "modules" of an ontology relevant to a particular application scenario (ontology decomposition) or developing them independently and integrating into a larger ontology (ontology composition). In this paper, we investigate state-of-the-art approaches in modular ontologies focusing on techniques that are based on rigorous logical formalisms as well as well-studied graph theories. We analyze and compare how such approaches can be leveraged in developing tools and applications in the biomedical domain. We conclude by highlighting some of the limitations of the modular ontology formalisms and put forward additional requirements to steer their future development.

1. Introduction

In the context of computer and information sciences, an ontology models a domain of knowledge or discourse defined by a set of representative primitives such as classes (or sets), attributes (or properties), and relationships (or relations among class members) [35]. They provide a standard way of representing these primitives thereby supporting easy transmission and interpretation of data for various applications. Consequently, there has been a significant body of work in recent years [66] on developing ontology modeling languages [1,19,31], ontology editing environments [18,45,46], ontology reasoning [10,36,40,65], ontology storage and retrieval systems [38], ontology alignment and integration [22,23,43], as well as building ontology-based applications in various domains such as bioinformatics [26] and information retrieval [25,60,70]. These efforts have enhanced the power of the "traditional Web" by introducing a network effect caused by linking of various knowledge sources [37].

In practice, much of this success depends on the ability to share, reuse and personalize existing ontologies since designing and maintaining ontologies is deemed to be a time-consuming and labor intensive task. The reuse of existing ontologies can occur either while designing a new ontology or development of new applications. However, existing and widely-used ontology languages such as OWL [1] do not support reuse (or partial import) of

[*]Corresponding author: Division of Biomedical Statistics and Informatics, Mayo Clinic, Rochester, MN 55905, USA. pathak.jyotishman@mayo.edu. .

parts of other ontologies.[1] This becomes a big problem because even if one would like to use a single concept from a large ontology such as FMA,[2] s/he would end up importing the entire FMA ontology. Hence, much alike software engineering, where issues such as *software reuse*, *black-box behavior*, and *modules* are important [72], the domain of ontological engineering needs to adapt similar notions [30].

Toward this end, research in modular ontologies has drawn significant attention in the recent past. Informally, a module can be considered to be a subset of a "whole" that makes sense (i.e., is not an arbitrary subset randomly built) and can somehow exist separated from the whole, although not necessarily supporting the same functionality as the whole. Thus, in the context of ontologies, a module is a sub-ontology that "makes sense" either from an application (e.g., answering certain queries) or systems (e.g., improving performance) perspective [61]. Ideally, one would prefer modules that are *compact*, yet provide *guarantees* to capture the meaning of the terms and concepts used. In other words, exactly the same answers should be obtained when answering queries against a particular local ontology that imports a module relevant to certain concepts from a foreign ontology, versus the local ontology importing the entire foreign ontology. This ensures that importing the module instead of the whole foreign ontology will have no observable and logical consequence on the local ontology [29].

In general, modularization can be perceived in two orthogonal ways: independently developing modules that can be integrated coherently and uniformly (*ontology composition*) or extracting such modules from an integrated ontology for supporting a particular use case (*ontology decomposition*). Addressing both these requirements is challenging because traditional ontology formalisms such as description logics were primarily designed for single/centralized ontologies rather than multiple/decentralized ones [71]. Furthermore, support for dynamically handling interconnected modules is lacking: if $M_i(O)$ and $M_j(O)$ are modules of an ontology O that are interconnected, how will the updates in $M_i(O)$ reflect changes in $M_j(O)$? Finally, traditional reasoner implementations (such as Pellet)[3] were developed to operate on a single ontology as opposed to multiple ontologies.

We believe that these issues are highly relevant in the biomedical domain since most of the widely used ontologies such as SNOMED CT[4] or NCI Thesaurus[5] are large and complex, and will therefore benefit from the development of tools and techniques that enable proper development and management of multiple, distributed ontologies and corresponding reasoning support. In this paper, we survey and investigate a representative set of approaches for modular ontology languages and formalisms as well as provide use cases from the biomedical domain and additional requirements to steer their future development. While other survey work [34,61,71] in this topic has focused on rigorously evaluating modular ontology techniques from a theoretical perspective, our objective is not meant to do a comprehensive study of a handful of approaches with rigorous logical proofs. Instead, we survey a wider range of modular ontology proposals and analyze their potential for supporting novel applications to advance biomedical research.

[1] OWL provides an owl:imports construct that allows to combine ontologies by including all the axioms contained in an external ontology in one's local ontology. Consequently, everything in the transitive closure of the imported ontology becomes a part of the local ontology.
[2] http://fma.biostr.washington.edu.
[3] http://pellet.owldl.com.
[4] http://www.snomed.org.
[5] http://nciterms.nci.nih.gov.

1.1. Organization

The rest of the paper is organized as follows: Section 2 formulates the definition of an ontology module within the scope of this paper, Section 3 introduces various modular ontology formalisms, Section 4 discusses use cases and applications in biomedicine where such formalisms can be leveraged, and finally Section 5 concludes the paper with a summary and discussion.

2. Background and preliminaries

2.1. Ontology modules

Although the notion of a module is well-understood in the software engineering community, discussions on ontology modularization become murky since the concept can be understood in rather different ways. For the purposes of this paper, we adopt the following definition of an ontology module [21]: An ontology module is a reusable component of a larger or more complex ontology, which is self-contained but bears a definite association to other ontology modules, including the original ontology. This definition, in addition to implying that modules can be reused "as-is" or extended by introducing new concepts and relationships, asserts that they are not isolated entities or disjoint from each other. For example, assuming that a module is entirely based on subsumption relationships, if a concept *A* has sub-concepts *B* and *C*, such that *B* and *C* are disjoint siblings of *A*, creating a module centered on *A* would include all three concepts, whereas creating a module on *B* would only include *B*, thereby making both the modules not entirely disjoint. However, the general expectation is that modules developed by an ontology engineer will be comprised of distinct concepts rather than concepts that are closely related. In particular, we expect an ontology module to be *self-contained* (i.e., given a set of relations, the ontology module should be transitively closed with respect to those relations) and *logically consistent* (i.e., given an ontology that is logically consistent, every module extracted from it should be logically consistent as well). This would enable reasoning based on the axioms contained in just that module, ignoring the rest of the axioms in the ontology. For example, in the Health-e-Child project,[6] an ontology for describing Juvenile Rheumatoid Arthritis (JRA; see Fig. 1) is being developed that contains knowledge about different types of JRA distinguished by several factors such as joints affected or the occurrence of fever, and each type of JRA requires a different treatment [44]. However, some of this information is already modeled in existing ontologies such as NCI Thesaurus and GALEN[7] that are widely used, and hence can be potentially re-used based on two important criteria: (i) when importing axioms from NCI-Thesaurus and GALEN, the developers of the JRA ontology should not change the original meaning of the re-used knowledge, and (ii) for efficiency, only those axioms should be re-used that are relevant to the JRA ontology. The objective of the modular ontology formalisms is to facilitate such a process.

More concretely, let O be an ontology comprising of a set of axioms (classes, relationships such as subclass or equivalence, and individuals) and $\sum(O)$ be the signature of O constituting a set of entity names occurring in the axioms of O, i.e., its vocabulary. Given an axiom α, we define an ontology module $M_i(O)$ of an ontology O, with $\sum(M_i(O)) \subseteq \sum(O)$, such that $M_i(O)$ contains the same information about $\sum(\alpha)$ as O, and hence behaves in exactly the same way as O in all applications using only the symbols in $\sum(\alpha)$. As illustrated previously, the potential list of such applications include (i) importing $M_i(O)$, instead of O, into another ontology O', and (ii) querying a database using $M_i(O)$ instead of O. Intuitively,

[6] http://www.health-e-child.org.
[7] http://www.opengalen.org.

if O' imports $M_i(O)$, $O' \cup M_i(O)$ yields the same logical consequences as $O' \cup O$ for those axioms which are expressed using signatures from any subset of $O' \cap M_i(O)$.

2.2. Goals of ontology modularization

To understand the exact notion of ontology modularization, and calibrate the pros and cons, we outline a few goals that can drive a particular modularization task [61]:

– *Partial Reuse:* With the increasing popularity of Semantic Web and modern ontology development languages such as OWL [1], there has been a proliferation of numerous domain-specific ontologies in the recent years. Thus, to promote their adoption in developing both intra- and inter-domain applications, reuse emerges as an important issue. Since in many cases, only specific segments of a particular ontology might be used by the services and applications, it becomes pivotal to develop rich mechanisms to describe ontology modules. Unfortunately, as opposed to partial reuse, ontology languages such as OWL only provide the option of including (via the owl:imports construct) all the information present in an external ontology into one's local ontology.

– *Complexity:* With the increasing size of an ontology, one is faced with the cognitive burden of comprehending and managing the added complexity. Previous studies [2,56] have shown that it is easier for ontology engineers to follow an approach for collaboratively building individual modules which can later be integrated into a single ontology. Arguably, such a divide-and-conquer approach also contributes to management of the modules as well as improvements in the accuracy of their design.

– *Ownership and Customization:* Many ontology-based applications require customization and personalization of available ontologies to meet their requirements. This also influences various aspects of ownership which in practice can be attached to existing modules to make them comply to a personalization environment.

– *Efficient Reasoning:* Even with the recent developments in ontology reasoning techniques [55,65] and relatively easy access to large amounts of high-performance computing and memory power, the performance of the existing reasoners decrease significantly with the increasing size of the ontology. Consequently, if the amount of ontological knowledge to be analyzed is limited via creation of relevant modules, one can expect reasonable performance levels. However, at the same time, it remains to be seen whether a given reasoning task that requires a distributed network of modules to be analyzed is more efficient against reasoning with a single (large) ontology [64].

– *Tooling Support:* Similar to advances in building efficient reasoners, various other tools such as ontology editing environments [18,45,46] and storage and querying systems [38] have been developed over the last couple of years. However, despite the continuous efforts to improve these tools, their performance is affected by the size and complexity of the ontology which warrants developing effective techniques for ontology modularization.

2.3. Properties of ontology modules and modularization techniques

The main properties that an ontology module needs to satisfy have been studied and identified in previous work [30,34,47,61,67,71]. While there is no considerable agreement on all the properties, there is a general consensus on some of the main aspects of an ontology module:

– *Size:* A module $M_i(O)$ of an ontology O should be as small as possible.

– *Correctness:* A module $M_i(O)$ of an ontology O should contain only the information that is present in O. In other words, for any knowledge that can be inferred from $M_i(O)$, it should be possible to infer the same knowledge from O.

– *Completeness:* A module $M_i(O)$ of an ontology O, extracted for a particular signature $\sum$, should contain all the information relevant to the elements of $\sum$. This ensures that there is no difference in the logical consequence from importing $M_i(O)$ versus O based on $\sum$.

To ensure that most (if not all) of these properties are satisfied, we outline some evaluation criteria and requirements for the modularization techniques [7,61]:

– *Localized Semantics:* In addition to being syntactically modular (i.e., represented in separate XML namespaces), a modular ontology should also be semantically modular to ensure that a global model is not required to integrate the individual ontology modules.

– *Correct Reasoning:* The logical consequences that can be deduced by reasoning on a single large ontology should be semantically equivalent to what can be deduced by reasoning over a collection of individual ontology modules extracted from the large ontology. For example, consider Fig. 3 where an integrated ontology O contains the axioms $\{P \sqsubseteq Q, Q \sqsubseteq R, R \sqsubseteq S\}$, and modularized version of O has two modules $O_1 = \{P \sqsubseteq Q\}$, $O_2 = \{R \sqsubseteq S\}$, and a semantic connection $Q\overrightarrow{\sqsubseteq}R$, which represents the modularized version of the axiom $Q \sqsubseteq R$. The answer to any reasoning problem (e.g., is $P \sqsubseteq S$?) obtained by integrating O_1, O_2 and $Q\overrightarrow{\sqsubseteq}R$ should be the same as that obtained by applying a sound and complete reasoner on O.

– *Transitivity:* It should be possible to reuse the knowledge contained in a particular ontology either directly or indirectly. That is, for an ontology O, if a module $M_k(O)$ is imported by $M_j(O)$, and $M_j(O)$ in turn is imported by the module $M_i(O)$, then $M_i(O)$ should be able to reuse the knowledge contained in $M_k(O)$. For example, in Fig. 4, knowledge in ontology module O_1 ($P \sqsubseteq Q$) may be used by the module O_3 to infer that $P \sqsubseteq S$.

– *Safe Reuse:* In addition to enabling transitivity, it is important to ensure safe reuse by guaranteeing that the meaning of the imported terms are not changed in the importing ontology. Essentially, this means that while it is possible for the importing ontology to define sub- and super-classes of the imported terms, defining relationships between two imported terms is not allowed.

– *Decidability:* The modular ontology should be decidable so that one can do reasoning on the ontology within a finite time period.

3. Techniques in modular ontologies

As mentioned in Section 1, many approaches and formalisms have been developed for modular ontologies in the recent past. In this section, we review a representative set of such techniques and analyze their fitness with respect to the requirements identified in Section 2.3. Specifically, we categorize these techniques into two sets: the first set of techniques are based on rigorous logical foundations for extracting modules, and the second set of approaches adopt graph-theoretic algorithms for traversing the ontology hierarchy and applying heuristics to extract relevant modules.

3.1. Logic-based approaches

In this Section we discuss approaches that develop formal algorithms based on sound logical foundations for module extraction that are correct and complete. Arguably, correctness of a module (see Section 2.3) is trivial to satisfy since a module computed by extracting a subset

of axioms from an ontology will only contain information about that ontology. Defining completeness, on the other hand, is somewhat non-trivial, and has only been recently defined formally based on conservative extensions by Grau et al. [28] (see Section 3.1.5 for details). Briefly, an ontology *O*' is considered a conservative extension of another ontology *O* for a signature $\sum$, if the definition of the terms contained in $\sum$ remains unchanged in *O*'. According to the authors in [28], this implies that if *O* contains a subset of axioms defined in *O*', such that *O*' is a conservative extension of *O* with respect to the signature $\sum$, then *O* is complete for the terms in $\sum$. To facilitate better understanding of these ideas as well as modular ontology formalisms that we outline in the remainder of this section, we provide a brief introduction to description logics.

3.1.1. Background on description logics—Description Logics (DL) [5] is a family of knowledge representation languages which can be used to represent the terms and concepts and relationships between them for a given domain in a structured and formally well-defined manner. More specifically, the *syntax* of a DL is given by a *signature* $\sum$ which is a disjoint union of a set **AC** of *atomic concepts* (*A*, *B*, …) representing a set of elements, a set **AR** of *atomic roles* (*r*, *s*, …) representing binary relations between elements, and a set **Ind** of *individuals* (*a*, *b*, …) representing the elements. Every DL provides *constructors* for defining the set Rol($\sum$) of (general) *roles* (*R*, *S*, …), the set Con($\sum$) of (general) *concepts* (*C*, *D*, …), and the set Ax($\sum$) of *axioms* (α, β, …) for a signature $\sum$ which is a union of *role axioms* (RBox), *terminological axioms* (TBox), and *assertions* (ABox).

$\mathcal{EL}$ [3,4] is the simplest DL language which allows *conjunction* of atomic concepts to construct complex concepts ($C_1 \sqcap C_2$) as well as defining *existential restriction* ($\exists R.C$) starting with atomic concepts *A*, roles *R*, and the bottom concept $\bot$. The axioms in an $\mathcal{EL}$ TBox can either be *concept definitions* ($A \equiv C$) or *general concept inclusions* ($C_1 \sqsubseteq C_2$), whereas $\mathcal{EL}$ ABox *concept assertions* and RBox *role assertions* are of the form $a : C$ and $r(a, b)$, respectively.

One of the most widely used DLs, which provides the foundation for other more expressive DLs is $\mathcal{ALC}$ (Attributive Language with Completeness) [62], and is essentially obtained from $\mathcal{EL}$ this, by adding *complement of concepts* (¬C). Based on this, $\mathcal{ALC}$ introduces various additional constructors such as the *top concept* $\top$ which can be represented by $\neg\bot$, *disjunction of concepts* which can be represented by $\neg(C_1 \sqcap \neg C_2)$, and *universal restriction* $\forall R.C$ which can be represented by $\neg(\exists R.\neg C)$. For example, a simple ontology in $\mathcal{ALC}$ is shown in Fig. 2. The axioms in the ontology assert that Human is a Mammal (axiom 1); a Woman is a Human who only eats Food (axiom 2); Laura is a Woman (axiom 3); and Laura eats Pizza, which is an individual of type Food (axiom 4).

In addition to above, research by the DL community has proposed numerous extensions to $\mathcal{ALC}$ meet the needs of applications that require more expressivity. In particular, the logic $\mathcal{SH}$ [41] is extended from $\mathcal{ALC}$ with transitive roles and role inclusions ($\mathcal{H}$). This is further extended in [42] to propose the logic $\mathcal{SHOIQ}$ by including nominals ($\mathcal{O}$), inverse roles ($\mathcal{I}$), qualified number restrictions ($\mathcal{Q}$). Notably, $\mathcal{SHOIQ}$ provides the logical vides foundations of the Web Ontology Language (OWL) [1]. Interested readers can refer to [5] to find further details about other DL extensions.

The *semantics* of the DL languages are based on the usual Tarski-style set-theoretic semantics as follows: an *interpretation* $\mathcal{I}$ of a description logic knowledge base is a pair $\mathcal{I} = \left\langle \Delta^{\mathcal{I}}, (.)^{\mathcal{I}} \right\rangle$, where the interpretation domain $\Delta^{\mathcal{I}}$ contains a non-empty set of objects and the interpretation function $(.)^{\mathcal{I}}$ that maps every $A \in$ **AC** to a subset $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, every $r \in$ **AR** to

a binary relation $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, and every $a \in$ **Ind** to an element $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$. Based on this, the semantics of the $\mathcal{ALC}$ constructors is given in Table 1.

We further say that an interpretation $\mathcal{I}$ is a *model* of an ontology O if $\mathcal{I}$ satisfies all the axioms in O. An ontology O *implies* an axiom α (denoted by $O \models \alpha$) if $\mathcal{I} \models \alpha$ for every model $\mathcal{I}$ of O. Given a set **I** of interpretations, an axiom α is *valid* in **I** if every $\mathcal{I} \in$ **I** is a model of α. Furthermore, an axiom α is a *tautology* if it is valid in the set of all interpretations. As mentioned above, more theoretical details can be referred from [5].

3.1.2. Distributed description logics—Distributed Description Logics (DDL) [11] is a knowledge representation formalism for representing sets of ontologies and semantic relations between them. It provides a mechanism for referring to ontologies and for defining rules that connect "concepts" in different ontologies. This is achieved using the notion of importing and reusing concepts between ontologies and enabling reasoning with multiple ontologies interconnected by directional semantic mapping (called the *bridge rules*). In particular, DDL extends the notion of *interpretation* introduced above to fit the distributed nature of the model and to reason about concept subsumption across ontologies.

More formally, let I be a set of non-empty indices, such that $\{O_i\}_{i \in I}$ is a set of ontologies. Concepts and axioms are represented with the index of the ontology they belong to such that $i : C$ denotes a concept in ontology O_i and $j : C \sqsubseteq D$ represents that concept C is a sub-concept of D in ontology O_j, where $i : C$ and $j : C$ are different concepts. Semantic relations between concepts in different ontologies are represented via axioms, called bridge rules, that are of the following form:

$$-i{:}C \xrightarrow{\sqsubseteq} j{:}D \,(\text{into}-\text{rule}$$
$$-i{:}C \xrightarrow{\sqsupseteq} j{:}D \,(\text{onto}-\text{rule}$$

where C and D are concepts in ontologies O_i and O_j, respectively. Furthermore, the derived bridge rule $i{:}C \xrightarrow{\equiv} j{:}D$ can be defined as a conjunction of the into- and onto-bridge rules. These rules do not represent the semantic relations stated from an external observation point of view such as the Web. Instead, a rule i to j expresses relations between i and j viewed from j-th *subjective* point of view. Specifically, an into-bridge rule $i{:}C \xrightarrow{\sqsubseteq} j{:}D$ states that, from j-th point of view, the concept C in i is less general that its "local" concept D. Equivalently, the onto-relation $i{:}C \xrightarrow{\sqsupseteq} j{:}D$ expresses the more generality relation. In general, note that the into-rule ($i{:}C \xrightarrow{\sqsubseteq} j{:}D$) is not necessarily an inverse of the onto-rule ($i{:}C \xrightarrow{\sqsupseteq} j{:}D$) since these rules reflect a subjective point of view. For example, consider fragments of class hierarchies from two simple ontologies shown in Fig. 5. The following are some bridge rules that can be defined:

$$-1{:}\text{Publication} \xrightarrow{\equiv} 2{:}\text{Publication}$$
$$-1{:}\text{InProceedings} \xrightarrow{\sqsubseteq} 2{:}\text{ConferencePaper} \cup 2{:}\text{WorkshopPaper}$$
$$-1{:}\text{InBook} \xrightarrow{\sqsupseteq} 2{:}\text{BookArticle}$$

Thus, a "distributed ontology" $\mathcal{F}$ can now be defined as a tuple, $(\{O_i\}_{i \in I}, \{R_{ij}\}_{i \neq j \in I})$, where $(\{O_i\}_{i \in I}$ is the set of ontologies, and $\{R_{ij}\}_{i \neq j \in I})$ is the set of bridge rules between those ontologies. Arguably, the notion of bridge rules in DDL has a semantic difference with respect to the notion of concept inclusion in classical DLs, which is regarded as subset

relation between concept interpretations (e.g., $C^{\mathcal{I}_i} \subseteq D^{\mathcal{I}_j}$). As mentioned in [71], depending on the application context, this can be a limitation of DDL. For instance, even though subset relations in classical DLs are transitive, the DDL domain relations are not transitive. Consequently, the bridge rules cannot be transitively reused across modules. Furthermore, DDL allows arbitrary domain relations which can create undesirable consequences from a reasoning point of view (e.g., empty domain relations can be specified in DDL, although general concept inclusions between satisfiable concepts in DL enforce restrictions on non-empty interpretations). Additionally, DDL does not allow *inter-module* role correspondences. For example, based on our example in Fig. 5, it is not possible to define the following relation using DDL: GradStudent ⊑ ∃writes.Thesis, where writes is an inter-module role.

3.1.3. $\mathcal{E}$-connections—As described above, DDL allows only one type of domain relations. The $\mathcal{E}$-Connections approach [33], on the other hand, allows multiple "connections" between modules. Informally, an $\mathcal{E}$-Connection is a set of "connected" ontologies, wherein each $\mathcal{E}$-Connected ontology, similar to OWL, contains informations about classes, properties and their instances, and additionally new kind of properties called *link properties*. The link properties are logically interpreted as binary relations, where the first element belongs to its source ontology and the second to its target ontology. Thus, in essence, the link properties are used to relate individuals belonging to different ontologies in combination. For example, consider the two ontologies in Fig. 6 which shows information about Persons and Pets. Using $\mathcal{E}$-Connections and someValuesFrom OWL restriction, it is possible to define links such as Dog_Owner ≡ Person ⊓ ∃owns.Dog and Unhappy_Pet_Owner ≡ Person ⊓ ∃owns.Unfriendly_Pet.

More formally, let I be a set of non-empty indices, such that $\{O_i\}_{i \boxtimes I}$ is a set of ontologies. The semantics of the combined knowledge base is given with respect to a combined interpretation $\Gamma = \left\langle \{\mathcal{I}_i\}_{i \in I}, \left\{\mathcal{E}_{ij}\right\}_{i \neq j \in I} \right\rangle$, where $\mathcal{I}_i$ is a DL interpretation of the ontology O_i and $\mathcal{E}_{ij}$ is a cross-domain relation such that $\mathcal{E}_{ij} \subseteq \Delta^{\mathcal{I}_i} \times \Delta^{\mathcal{I}_j}$. The existential quantification ($\exists E.C$) and & value restriction ($\forall E.C$), where $E \in \mathcal{E}_{ij}$ is a link and C is a concept in O_j, are interpreted as subsets of $\Delta^{\mathcal{I}_i}$ as follows:

$$-(\exists E.C)^{\Gamma} = \left\{x \in \Delta^{\mathcal{I}_i} | \exists y \in |Delta^{\mathcal{I}_j}, \langle x, y\rangle \in E^{\Gamma}, y \in C^{\Gamma}\right\}$$
$$-(\forall E.C)^{\Gamma} = \left\{x \in \Delta^{\mathcal{I}_i} | \exists y \in |Delta^{\mathcal{I}_j}, if\ \langle x, y\rangle \in E^{\Gamma} \rightarrow y \in C^{\Gamma}\right\}$$

However, $\mathcal{E}$-Connections require that the $\mathcal{E}$-Connected ontologies be disjoint from each other, thereby enforcing strong restrictions in some applications [71]. For example, a concept cannot be declared as subclass of another concept in a foreign module thereby ruling out the possibility of asserting inter-module subsumption. Similarly, it does not allow specification of relationships between other relationships belonging to different ontologies. Both these constraints do not allow general support for transitive usability: foreign classes and foreign properties cannot be instantiated, relations (e.g., sub-property) cannot be specified between local and foreign properties, and cross-module conjunction and disjunction are also not allowed. Furthermore, $\mathcal{E}$-connections do not allow a same term to be used both as a link name, and also as a role name – a feature commonly referred to as "punning" [32], where the same name can have different interpretations.

3.1.4. Package-based description logics—Package-based Description Logics (P-DL) [8] overcomes the strong module disjointness limitations of DDL and $\mathcal{E}$-connections, and OWL importing issues which only allows the model of an imported ontology to be

completely embedded in a global model, thereby hindering partial reuse of ontologies. In P-DL, an ontology comprises of a set of modules called *packages* where each concept, a property or an individual is associated to a *home package*. P-DL introduces the notion of *importing relations* such that a package can use terms defined in other packages (called as *foreign packages*). That is, if a package P_j uses a term $i : C$, with the home package P_i such that $i \neq j$, then in P-DL, the term C is imported in P_j, and the importing relation is denoted by r_{ij}^{C}. This relation is *partial* in that only the commonly shared terms are interpreted in the overlapping part of the local models.

The semantics of P-DL is given with respect to the domain relation, r_{ij}, between local interpretations $\mathcal{I}_i$ and $\mathcal{I}_j$ (of packages P_i and P_j, respectively), where $r_{ij} \subseteq \Delta_i \times \Delta_j$, such that: (i) for any $x \in \Delta_i$, there at most one $y \in \Delta_j$, such that $(x, y) \in r_{ij}$, and vice versa; and (ii) $r_{ij} = \circ r_{kj}$, where $\circ$ denotes the transitive compositional function. As a consequence of (ii), P-DL enables transitive reuse of knowledge in the sense that if a package P_i asserts that $C \sqsubseteq D$, and P_j directly imports that axiom from P_i, then $C \sqsubseteq D$ is also valid from P_j's perspective. Furthermore, P-DL facilitates contextualized interpretation of knowledge by ensuring that the interpretation of assertions in each ontology module is constrained by their context. That is, when knowledge in a particular module is reused by another module, the interpretation of the reused knowledge should be constrained by the context in which the knowledge is being reused. As an example, in western countries, the concept weekend typically refers to the days Saturday and Sunday of a week, whereas in Islamic countries it is Friday and Saturday [6]. Thus, depending on the context, the concept weekend has different interpretations, and it is vital to preserve it. Additionally, P-DL semantics for less expressive logics ensure that both distributed reasoning on modular ontologies and classical reasoning on an integrated ontology comprised of the respective modules will yield the same conclusion; although for more expressive logics this has not been investigated. For example, consider the ontologies shown in Fig. 8: ontology P contains knowledge about People, and ontology W imports some of the knowledge from P to describe new information. Here, the axioms 2: MaleEmployee $\sqsubseteq$ 1: Man and 2: FemaleEmployee $\sqsubseteq$ 1: Woman illustrates that semantic importing approach can realize concept specialization, whereas the axiom 1: Child $\sqsubseteq \neg$2: Employee illustrates concept generalization [8].

3.1.5. Conservative extensions and locality-based modularization—Grau et al. [28–30] propose the notion of conservative extensions to support partial reuse of ontologies where the objective is to extract from a "foreign ontology" a small fragment that captures the meaning of terms used in a "local ontology". For example, when building an ontology about research projects, one may use terms such as CysticFibrosis and HeartFailure from a separate ontology in the description of medical research projects. Conservative extensions ensures that the nature of the modules extracted are as small as possible (ideally), yet guarantee that querying using the modules provide the same answers as if the entire foreign ontology was imported. In particular, the main intuition behind conservative extension is to ensure local completeness of the modules such that the knowledge contained in each individual module is not altered even after their integration. That is, integrating modules cannot induce new relationships between existing concepts in any module. More formally, for two ontologies $O' \subseteq O$ and a signature $\sum$, O is a *deductive* $\sum$*-conservative extension* of O', if for every axiom α with $\sum(\alpha) \subseteq \sum$, we have $O \models \alpha$ if and only if $O' \models \alpha$. Furthermore, O is a *model* $\sum$*-conservative extension* of O', if for every model $\mathcal{I}'$ of O', there exists a model $\mathcal{I}$ of O such that $\mathcal{I}|_\Sigma = \mathcal{I}'|_\Sigma$, where $\mathcal{I}|_\Sigma$ and $\mathcal{I}'|_\Sigma$ are interpretations of O and O' with respect to $\sum$, respectively. Intuitively, given $O' \subseteq O$, O is considered a deductive conservative extension for a signature $\sum$ iff every logical consequence α of O constructed using the symbols in $\sum$ can also be deduced from O'. Consequently, deductive conservative extension depends on the description logic **D** in which O and α are expressed. On the other

hand, model conservative extension states that for $O' \subseteq O$, O is a model conservative extension if every model of O' can be expanded to a model of O by interpreting new symbols and leaving the interpretations of the old symbols unchanged. As a result, model conservative extension is *stronger* that deductive conservative extension since it does not depend on the expressivity of the description logic language [53]. For example, consider a simple ontology T in Fig. 7 describing medical diseases and disorders, saying, e.g., that Genetic Fibrosis is a Genetic Disorder. Suppose now that we want to create another ontology R that describes research projects for these diseases and disorders. We extend the signature of T by adding concepts Genetic Disorder Project and Cystic_Fibrosis_Project in R by stating that these projects focus on Cystic Fibrosis and Genetic _Disorder, respectively. Intuitively, this addition should not affect the knowledge present in T , thereby, stating that $T \cup R$ is a conservative extension of T .

However, the problem of determining model conservative extensions is computationally unsolvable, and requires development of approximate algorithms. Toward this end, tractable modularization algorithms based on the notion of *locality* have been proposed in [28,30], where the idea is to, for a signature $\sum$, identify a set of axioms $\alpha \in A$ that do not change the meaning of terms in $\sum$. Such axioms are called *local* to the terms in $\sum$, and the authors in [28,30] define six different types of locality, out of which, the most commonly used are the $\top$-locality (top locality) and $\bot$-locality (bottom locality): if an axiom does not define new sub-concepts for a given concept C, then it is considered $\top$-local for C, and it is $\bot$-local for C if it does not define new super-concepts. Thus, for a given set of concepts, modules extracted using $\bot$-locality will contain all the super-concepts of those concepts, whereas, modules containing sub-concepts can be extracted using $\top$-locality. More details on the locality-based module extraction algorithms are presented in [30].

3.1.6. Module extraction algorithms for $\mathcal{EL}^+$ ontologies—We introduced the $\mathcal{EL}$ description logic towards the beginning of Section 3.1. The $\mathcal{EL}$ family of DLs restrict the expressivity of OWL to gain tractability, but are yet widely used to develop some of the important biomedical ontologies such as the Gene Ontology[8] and SNOMED.[9] Since reasoning in $\mathcal{EL}$ is polynomial time, recently there has been a significant research interest in developing modularization algorithms for $\mathcal{EL}$ and more expressive languages such as $\mathcal{EL}^+$ [49,69].

In particular, Konev et al. [48], proposed two algorithms, CEL and MEX, for extracting relevant fragments from large $\mathcal{EL}^+$ ontologies. CEL is based on the notion of *connected reachability* [69] where, for a given reachability graph (comprising of nodes that correspond to concepts in an ontology and edges representing the axioms) and signature $\sum$, a connected node is reachable from $x \in \sum$ iff all the symbols in the node are reachable to x, such that reachability between two concepts suggests a potential subsumption relationship between them. By using heuristics to traverse this reachability graph, CEL identifies logically complete modules for $\mathcal{EL}^+$ ontologies. The MEX algorithm, similar to CEL, generates modules, but of much smaller sizes, and can only operate on acyclic ontologies.

3.2. Graph theory-based approaches

A common, and arguably, simple approach to modularize an ontology is to traverse the ontology hierarchy (i.e., the set of axioms), and apply heuristics to identify a sub-graph. Even though useful, such approaches do not take into consideration the underlying semantics of the ontology, and hence do not generate modules that are complete (see Section

[8] http://www.geneontology.org/.
[9] http://snomed.org/.

2.3). Nonetheless, these algorithms are tractable, intuitive to an user, and are widely used; we discuss a representative set of graph traversal-based modularization algorithms in the remainder of this section.

3.2.1. GALEN segmentation service—Seidenberg and Rector [63] developed a methodology for extraction of related concepts from GALEN based on one or more classes given as input by the user. Figure 9 (adopted from [63]) gives a pictorial representation of the approach where starting from a "target" concept, the algorithm extracts all the concepts in the paths to the root and all the leaf concepts. Additionally, any links across the hierarchy from any of the previously traversed classes are followed, and the hierarchy is traversed upwards (but not downwards) from any of these classes that the cross-links point to. Links pointing at other classes from these newly traversed classes are also included. This continues until there are no more links left to follow. The authors in [63] shows that using their approach, the size of GALEN was reduced by a factor of 20 and the ontology segments extracted could be classified within seconds. However, it is unclear how the algorithm can be generalized and applied to other ontologies apart from GALEN.

3.2.2. ModTool: A module extraction tool—Doran et al. [21] propose another technique for extracting ontology modules based on a graph/hierarchy traversal approach similar to the GALEN segmentation service. However, unlike [63], the algorithm does not allow upward navigation of the subclass hierarchy from the target concept based on the justification that such a traversal will substantially increase the probability of extracting modules that are as large as the whole ontology. Furthermore, the technique ensures that the modules extracted are transitively closed with respect to the relations that are traversed. A preliminary evaluation based on the NCI Oncology ontology showed that this approach generated modules of smaller size compared to those generated by $\mathcal{E}$-Connections [33].

3.2.3. Modularization via dynamic selection—Similar to above approaches, authors in [17] develop another mechanism for modularization that relies on exploiting the hierarchical relationships in an ontology. Given a set of input terms, the objective is to extract the smallest part of the ontology covering those terms via a fixed-point algorithm. In particular, the algorithm recursively inspects the ontology expressions to include elements that participate in the definition of the input terms. Although, unlike the GALEN segmentation service [63], instead of including all super-concepts of a selected concept, only the most specific common super-concepts are included. Furthermore, the algorithm removes from the hierarchy all the intermediate concepts that do not participate in the semantic definition of the considered terms, and only retains the hierarchical structure. While no rigorous evaluation was presented in [17], the authors claim satisfactory results.

3.2.4. PATO: Partitioning tools for ontologies—Stuckenschmidt and Klein [68] propose an approach for partitioning light-weight ontologies (primarily class hierarchies) into disjoint and covering sets of concepts. Their approach is based on extracting a weighted dependency graph, $D_G =< C,D,w >$, from the class hierarchy, where C represents the set of concepts, and links D between the concepts represent different types of dependencies that can be derived from the ontology and weighted according to the strength of the dependency. The weights, in particular, give an estimate of connectivity between concepts and is used to determine sets or clusters of concepts that are strongly interconnected. The authors perform experiments on realworld ontologies such as NCI Thesaurus and generated modules with an average size of 57 concepts, although with a larger variance as module sizes ranged between 4 and 268 concepts. Figure 10 shows a screenshot of the PATO tool.

3.2.5. Prompt traversal views—The Prompt Traversal View [57] is a Protégé [46] plugin that allows a user to select target concept(s) of interest, and extract additional concepts based on relationships that the user wants to explore until a particular traversal depth is reached. It allows a user to tweak various additional traversal parameters manually and is essentially a tool that is targeted to assist users in manually extracting and inspecting smaller parts of the ontology. While useful for this specific use case, it is unclear how the tool can be applied for (semi-) automatic ontology module extraction. Also, experimental results were not presented in [57] which makes it harder to judge the performance of Prompt compared to other tools, especially when dealing with large ontologies. Figure 11 shows a screenshot of the Prompt Protégé plugin tool.

3.2.6. LexValueSets: Heuristics-based sub-graph extraction—Pathak et al. [59] proposed an approach for extracting *value sets* from ontologies which are essentially an uniquely identifiable set of valid values that can be resolved at a given point in time to an exact set (collection) of codes or classes in the ontology. This subset of values can then be used for various applications such as modeling a picklist for data entry. In particular, given a signature $\sum$, the algorithm proposed in [59] identifies a set of "best paths" between the terms in $\sum$, and extracts all the concepts in those paths to construct a value set. The criteria for identifying a best path p between any two concepts *x, y* in an ontology *O* is captured as follows: (i) p does not contain the root node/concept of hierarchy unless *x* or *y* is the root node; (ii) p is not circular; and (iii) if there exists another path p' between *x* and *y*, such that it satisfies the above conditions and comprises of more granular nodes (i.e., sub-concepts of *x* and *y*) than p, then p' is the best path.

3.2.7. Taxonomy-based partitioning of gene ontology—Kunierczyk [51] described an approach for partitioning the Gene Ontology (GO) by relating terms in GO to the Taxonomy of Species (TS). This work is based on the premise that "GO Slims", which are essentially cut-down versions of GO containing a subset of terms that are created by users according to their application needs, are imprecise with respect to their semantics, and require error-prone manual construction. To reduce this effort, the author proposes the idea of establishing explicit relationships between GO terms and *taxa* which are classes of organisms, and leverage them to generate taxon-dependent (e.g., species-specific) views of the GO on-demand. In particular, three different types of relationships, namely, *validity*, *specificity* and *relevance*, are proposed that determine how the GO hierarchy should be navigated to automatically generate different types of views. For example, if there is a *valid* link between a GO term suckling behavior and the taxon Mammalia, then all the terms for which suckling behavior is a successor can be automatically included in a "valid for Mammalia" view of GO, even though none of those terms are explicitly marked as *valid* for Mammalia.

4. Applications of modular ontology techniques in biomedicine

The domain of life sciences, biology and medicine in particular, has a strong tradition of structuring their terminological knowledge in terms of controlled vocabularies, thesauri or classifications. Such scientific vocabularies have been successfully and widely used in various applications ranging from clinical trials to patient billing. However, at the same time, it has become apparent that the challenges for proper construction, management and usage of biomedical ontologies is far from trivial as initially expected, in part due to their sheer size and complexity. We believe that research in modular ontologies is poised to address some of these challenges and can be leveraged for building novel applications. We outline a few of them in the following.

4.1. Ontology reuse

Knowledge-based systems that support applications such as decision support in healthcare typically depend on large amounts of domain knowledge. However, capturing domain knowledge, in the form of ontologies, is expensive and reuse is always encouraged. In particular, the ability to *partially* reuse ontologies is vital because for many applications only specific ontology segments are relevant as opposed to the entire ontology. However, existing ontology languages such as OWL only support importing the whole ontology. As illustrated earlier, modular ontology techniques such as P-DL natively support partial ontology import with contextualized semantics, thereby ensuring that the inferences are drawn from the point of view of the (local) ontology importing (foreign) ontologies.

4.2. Ontology alignment

Ontology alignment is the process of determining correspondences between multiple concepts in multiple ontologies. Over the last several years, enormous amounts of effort have been invested in aligning and harmonizing biomedical ontologies. However, manually aligning ontologies is a labor intensive process and often semi-automatic approaches [19,52] are sought. Nevertheless, in spite of many improvements, such systems generate incorrect mapping to varying degrees. Consequently, to avoid incorrect utilization of mappings (e.g., for query answering) it is vital to detect and fix defective mapping elements. Recently, techniques [54] based on DDL have been proposed to address this problem where logical reasoning is used to analyze the impact of mapping to ontologies it connects where the assumption is that a mapping, if incorrect, will cause inconsistency (unsatisfiability) in mapped ontologies.

4.3. Value set construction

In the context of vocabularies, a value set is an uniquely identifiable set of valid values that can be resolved at a given point in time to an exact set (collection) of codes. The main objective of modeling value sets is to specify a *concept domain* with certain *attributes* of interest such that the attribute-values can be obtained from one or more vocabularies of interest. An example of a concept domain could be “world countries”, and the representative value set will include countries such as USA and UK. In practice, these value sets are constructed manually from pre-existing ontologies such as SNOMED and ICD by constraining the value selection based on logical expressions (e.g., all sub-concepts of the concept ColonCancer) which, arguably, is a tedious and cumbersome process. We believe that all the techniques for ontology segmentation introduced earlier can be leveraged for semiautomatic value set construction. In particular, given a set of target concepts, the algorithms in [17,21,59,63] can act as a guide in traversing the hierarchy and select super-, sub-, sibling-concepts etc., and extract an initial value set that can be validated and refined by a domain expert.

4.4. Secure information exchange

Addressing issues related to privacy and selective sharing of information in the biomedical domain is of utmost importance. In particular, the ability to develop techniques for privacy-preserving query answering with ontologies, where the objective is to enable “selective” answering of queries against ontologies, will be of immense benefit in a healthcare setting. Toward this end, recent research in privacy-preserving reasoning with hidden knowledge using P-DL [9] has shown promising results. The crux of the technique is to divide an ontology into two mutually exclusive parts, visible and hidden, and answering queries by inferencing on both the visible and hidden parts, however at the same time ensuring that the hidden knowledge is never revealed. The approach is based on an Open World Assumption, and hence, a query that cannot be answered without disclosing the hidden knowledge to the

reasoner will be answered such that the reasoner is lacking the complete information to answer the query. While there are other approaches [24] that also prevent unwanted inference, they are based on a Closed World Assumption which make them overly restrictive for querying purposes.

4.5. Collaborative authoring

The process of building large ontologies such as SNOMED is often collaborative and involves a team of experts that communicate and reconcile their changes. Since their objective is to follow principled approaches to generate, manage, and integrate multiple components of the ontology, there is a requirement to minimize the impact of changes performed by a modeler on other parts of the ontology developed by other modelers. We believe that this is a major application area for modular ontologies where individual ontology mod- ules can be extracted (ontology decomposition), modified and updated, and then combined to form an integrated ontology (ontology composition). While other techniques for collaborative ontology authoring, such as Collaborative Protégé [56] and Semantic Wiki [50] exists, they cannot support modular representation of an ontology, and hence lack a principled way for controlling conflicts in editing or propagation of editing errors.

4.6. Distributed and incremental reasoning

In practice, various biomedical ontologies are autonomously created, maintained and interlinked/reused within different application contexts. This results in the formation of a distributed information resource thereby necessitating the support for distributed reasoning (as opposed to global reasoning) to address scalability and context-specific nature of the ontologies involved. Recently, distributed reasoning techniques such as DRAGO [64] based on DDL have been proposed where reasoning with multiple ontologies is accomplished by leveraging the bridge rules.

On a different note, modular ontology formalisms are playing a key role in providing support for incremental reasoning with changing knowledge bases [27]. The idea behind incremental classification is to persist the reasoner state such that in the event of an update, the previous classification tree can be reused. Arguably, if the classification tree can be partitioned into modules, the updates can be applied to the targeted module(s) and changes propagated to other relevant module(s). Such an approach is highly relevant in dealing with ontologies such as the Gene Ontology (GO) which are revised constantly since loading a new version of GO can easily be handled as an incremental update to the previous version.

4.7. Scalable querying

As more and more instance data gets tagged with concepts from ontologies (e.g.,in biomedical grids), the ability to efficiently and effectively support ontology-based querying is pivotal. Since reasoning is required for instance-level querying (e.g., if $X \sqsubseteq Y$ and $x \in X$, where X, Y are classes and x is an instance of X, querying for all instances of Y should also return x), modularized ontology reasoning support becomes important. Specifically, the ability to compute modules on-demand based on the queries asked and classifying such modules on-the-fly will allow one to efficiently reason about the relationships that may exist among the instance data resident in the data nodes. Such reasoning could be used for example to determine if data tagged with a given concept code is also a member of the set of instances belonging to other concept(s). However, to the best of our knowledge, none of the tools can provide such a support for querying based on modularized ontology reasoning, although promising results have been demonstrated by IBM's SHER reasoner [20,39,58] where the main idea is to group together individuals which are instances of the same class into a single individual to generate a "summary" ABox of smaller size. Then, reasoning can

be done (e.g., for query answering) on the simplified summary ABox, thereby improving query performance by a significant degree.

4.8. Semantic information integration

The explosion of massive amounts of biomedical data over the last several years has posed the challenge of integrating and querying distributed, and often, semantically heterogeneous "information silos". Our previous work in this area [12,13,15] has led to the development of approaches for explicitly associating ontologies to the database schemas as well as the content, and leveraging the ontologies for integration and querying of information. In particular, we demonstrated novel approaches for information extraction by proposing ontology-aware classifiers [14,16]. However, the work was based on associating one single global ontology to a data source (its content and schema), which could be a limitation in representing, for example, patient records containing information about demographics, diseases, medication, and drugs. Consequently, modular ontologies formalisms can be extended such that relevant fragments of a particular ontology (or a set of ontologies) can be associated to data repositories and enable semantics-based querying.

5. Summary and discussion

As new ontologies in the biomedical and life sciences domain begin to populate and existing ones continue to grow and become more complex, there is an increasing need to leverage, develop and expand modular ontology formalisms for a wide variety of applications. In this paper, we reviewed some of the recent developments in modular ontology formalisms beginning with a widely accepted definition of ontology modules and illustrating the important goals of ontology modularization. We identified several properties, namely size, correctness, and completeness, that need to be satisfied by the ontology modules. Even though module correctness can be trivially satisfied by all the algorithms we evaluated, satisfying module completeness and identifying minimal modules are much harder problems.

Based on these important properties, we analyzed a representative set of research work on modular ontologies, which were primarily categorized into logic-based approaches comprising of techniques that are based on rigorous logical foundations for extracting modules, and graph traversal-based approaches comprising of algorithms that apply heuristics to navigate the ontology hierarchy and identify relevant modules. While our preference would be to propose the adoption of logic-based approaches since they guarantee correctness and completeness of the extracted modules, some of the techniques such as $\mathcal{E}$-connections and P-DL are based on non-standard semantics, and hence will require the development of new ontology editors, reasoners as well as standards (apart from OWL), making them less attractive. The only exception to this is the approach based on Conservative Extensions and Locality-based Modularization since it ensures semantic modularity under classic first-order semantics. We also introduced various application scenarios, albeit not only specific to the biomedical domain, but nonetheless highly relevant, that could benefit from further development of modular ontology formalisms. We argue that with the increasing maturity of the modular ontology languages and tools, they are poised to address various challenges in the development, management and usage of large biomedical ontologies in particular for collaborative ontology authoring and partial ontology reuse.

However, at the same time, there are many potential areas of future development that have to be pursued for modular ontology approaches to become mainstream. For instance, even though the techniques for ontology segmentation based on graph traversal algorithms [17,21,63] are efficient and useful in practice, they have the risk of generating incomplete results since the hierarchy traversal is done without considering the semantics of the

underlying ontology language. This is in particular true for OWL ontologies because depending on the presence or absence of OWL defined and primitive classes in a particular ontology, its asserted class hierarchy will be different compared to an inferred hierarchy (obtained after classification). Since the traversal algorithms cannot make a distinction between defined and primitive classes, they ignore classes and relationships between them. As illustrated above, logic-based approaches such as conservative extensions address this issue by generating complete modules. However, on the other hand, they suffer from the lack of user-intuitiveness that the graph-based techniques provide to an ontology developer. Also, as mentioned earlier, all the modularization techniques focus on generating modules from the ontology and reasoning on them (known as TBox modularity). It is unclear what role modular ontology formalisms will play when dealing with instance data (known as ABox modularity). Addressing this issue is important because many applications (e.g., distributed data retrieval in a grid) require interaction with multiple data nodes storing instance information. Another shortcoming of modular ontology formalisms is the lack of a query language, unlike SPARQL which is the query language for RDF graphs. Just like SPARQL can be used to express queries across diverse data sources, whether the data is stored natively as RDF or viewed as RDF via middleware, it should be feasible to query distributed ontology modules (TBoxes) and corresponding instance data (ABoxes) in a principled way. Finally, since many modular ontology languages such as DDL and ε-Connections leverage mappings or bridge rules, the ability to detect, and potentially rectify, inconsistent mappings is vital. While approaches such as [54] have been proposed for debugging DDL bridge rules, there is a requirement to expand such work in a more general setting that will enable resolution of inconsistencies between ontology modules that are interlinked.

Acknowledgments

The authors would like to thank Jesse Erdmann for his assistance and participation in numerous discussions leading to this work. This research has been supported in part by the NHGRI grant (U54HG004028).

References

[1]. Antoniou, G.; van Harmelen, F. Web ontology language: OWL. In: Staab, S.; Studer, R., editors. Handbook on Ontologies in Information Systems. Springer-Verlag; 2003.

[2]. Arpirez, J.; Corcho, O.; Fernandez-Lopez, M.; Gomez-Perez, A. WEBODE: A scalable workbench for ontological engineering. International Conference on Knowledge Capture; 2001.

[3]. Baader, F.; Brandt, S.; Lutz, C. Pushing the EL Envelope. 19th International Joint Conference on Artificial EL Intelligence; 2005. p. 364-369.

[4]. Baader, F.; Brandt, S.; Lutz, C. Pushing the EL Envelope Further. OWLED 2008 DC Workshop on OWL:EL Experiences and Directions; 2008.

[5]. Baader, F.; Calvanese, D.; McGuinness, D.; Nardi, D.; Patel-Schneider, P. The Description Logic Handbook. Cambridge University Press; 2003.

[6]. Bao, J. PhD thesis. Department of Computer, Iowa State University; Ames, Iowa, USA: 2007. Representing and Reasoning with Modular Ontologies.

[7]. Bao, J.; Caragea, D.; Honavar, V. Modular Ontologies – A Formal Investigation of Semantics and Expressivity. 1st Asian Semantic Web Conference; Springer-Verlag; 2006. p. 616-631.LNCS 4185

[8]. Bao, J.; Slutzki, G.; Honavar, V. A Semantic Importing Approach to Knowledge Reuse from Multiple Ontologies. 2nd AAAI Conference on Artificial Intelligence; AAAI Press; 2007. p. 1304-1309.

[9]. Bao, J.; Slutzki, G.; Honavar, V. Privacy-Preserving Reasoning on the Semantic Web. IEEE/ACM/WIC International Conference on Web Intelligence; IEEE CS Press; 2007. p. 791-797.

[10]. Besnard P, Cordier M-O, Moinard Y. Ontology-based Inference for Causal Explanation. Integrated Computer-Aided Engineering. 2008; 15(4):351–367.

[11]. Borgida A, Serafini L. Distributed description logics: Assimilating information from peer sources. Journal of Data Semantics. 2003; 1:153–184.

[12]. Caragea, D.; Bao, J.; Pathak, J.; Silvescu, A.; Andorf, C.; Dobbs, D.; Honavar, V. Information Integration from Semantically Heterogeneous Biological Data Sources. 3rd International Workshop on Biological Data Management; IEEE CS Press; 2005. p. 580-584.

[13]. Caragea, D.; Pathak, J.; Bao, J.; Silvescu, A.; Andorf, C.; Dobbs, D.; Honavar, V. Information Integration and Knowledge Acquisition from Semantically Heterogeneous Biological Data Sources. 2nd International Workshop on Data Integration in Life Sciences; Springer-Verlag; 2005. p. 175-190.

[14]. Caragea, D.; Pathak, J.; Honavar, V. Learning Classifiers from Semantically Heterogeneous Data Sources. 3rd International Conference on Ontologies, DataBases, and Applications of Semantics for Large Scale Information Systems; Springer-Verlag; 2004. p. 963-980.LNCS 3291

[15]. Caragea, D.; Zhang, J.; Bao, J.; Pathak, J.; Honavar, V. Algorithms and Software for Collaborative Discovery from Autonomous, Semantically Heterogeneous, Distributed, Information Sources. Invited paper. 16th International Conference on Algorithmic Learning Theory; Springer-Verlag; 2005. p. 13-44.LNCS 3734

[16]. Caragea, D.; Zhang, J.; Pathak, J.; Honavar, V. Learning Classifiers from Distributed, Ontology-Extended Data Sources. 8th International Conference on Data Warehousing and Knowledge Discovery; Springer-Verlag; 2006. p. 363-373.LNCS 4081

[17]. d'Aquin, M.; Sabou, M.; Motta, E. Modularization: A Key for the Dynamic Selection of Relevant Knowledge Components. 1st International Workshop on Modular Ontologies; 2006.

[18]. Day-Richter J, Harris MA, Haendel M, Lewis S. OBO-Edit – An Ontology Editor for Biologists. Bioinformatics. 2007; 23(16):2198–2200. [PubMed: 17545183]

[19]. Dinakarpandian D, Tong T, Lee Y. A Pragmatic Approach to Mapping the Open Biomedical Ontologies. International Journal of Bioinformatics Research and Applications. 2007; 3(3):341–365. [PubMed: 18048196]

[20]. Dolby, J.; Fokoue, A.; Kalyanpur, A.; Kershenbaum, A.; Schonberg, E.; Srinivas, K.; Ma, L. Scalable Semantic Retrieval through Summarization and Refinement. 22nd AAAI Conference on Artificial Intelligence; AAAI Press; 2007. p. 299-304.

[21]. Doran, P.; Tamma, V.; Iannone, L. Ontology Module Extraction for Ontology Reuse: An Ontology Engineering Perspective. 16th ACM International Conference on Information and Knowledge Management; ACM Press; 2007. p. 61-70.

[22]. Euzenat J, Mocan A, Scharffe F. Ontology Alignments. Ontology Management. 2008:177–206.

[23]. Euzenat, J.; Shvaiko, P. Ontology Matching. Springer; 2007.

[24]. Farkas C, Brodsky A, Jajodia S. Unauthorized Inferences in Semistructured Databases. Information Science. 2006; 176(22):3269–3299.

[25]. Gao X, Vuong LPB, Zhang M. Detecting Data Records in Semi-Structured Web Sites based on Text Token Clustering. Integrated Computer-Aided Engineering. 2008; 15(4):297–311.

[26]. Goble C, Stevens R. State of the Nation in Data Integration for Bioinformatics. Journal of Biomedical Informatics. 2008; 41(5):687–693. [PubMed: 18358788]

[27]. Grau, BC.; Halaschek-Wiener, C.; Kazakov, Y. History Matters: Incremental Ontology Reasoning Using Modules. 6th International Semantic Web Conference; Springer-Verlag; 2007. p. 183-196.LNCS 4825

[28]. Grau, BC.; Horrocks, I.; Kazakov, Y.; Sattler, U. A Logical Framework for Modularity of Ontologies. 20th International Joint Conference on Artificial Intelligence; 2007. p. 298-303.

[29]. Grau, BC.; Horrocks, I.; Kazakov, Y.; Sattler, U. Just the Right Amount: Extracting Modules from Ontologies. 16th International Conference on World Wide Web; ACM Press; 2007. p. 717-726.

[30]. Grau BC, Horrocks I, Kazakov Y, Sattler U. Modular Reuse of Ontologies: Theory and Practice. Journal of Artificial Intelligence. 2008; 31:273–318.

[31]. Grau BC, Horrocks I, Motik B, Parsia B, Patel-Schneider P, Sattler U. OWL 2: The Next Step for OWL. Journal of Web Semantics. 2008 To Appear.

[32]. Grau, BC.; Horrocks, I.; Parsia, B.; Patel-Schneider, P.; Sattler, U. Next Steps for OWL. Workshop on OWL: Experiences and Directions; 2006.

[33]. Grau BC, Parsia B, Sirin E. Combining OWL Ontologies using epsilon-Connections. Journal of Web Semantics. 2006; 4(1):40–59.

[34]. Grau, BC.; Parsia, B.; Sirin, E.; Kalyanpur, A. Modularity and Web Ontologies. 10th International Conference on Principles of Knowledge Representation and Reasoning; AAAI Press; 2006. p. 198-209.

[35]. Gruber, T. Ontology. In: Liu, Ling; Ozsu, M. Tamer, editors. Entry in the Encyclopedia of Database Systems. Springer-Verlag; 2008.

[36]. Haarslev, V.; Moller, R. RACER system description. International Joint Conference on Automated Reasoning; 2001.

[37]. Hendler JA, Golbeck J. Metcalfe's law, Web 2.0, and the Semantic Web. Journal of Web Semantics. 2008; 6(1):14–20.

[38]. Hertel, A.; Broekstra, J.; Stuckenschmidt, H. Handbook of Ontologies. Springer-Verlag; 2008. RDF Storage and Retrieval Systems.

[39]. Heymans, S.; Ma, L.; Anicic, D., et al. Ontology Management. Vol. volume 7 of Semantic Web And Beyond Computing for Human Experience. Springer; 2008. Ontology Reasoning with Large Data Repositories; p. 89-128.

[40]. Horrocks I, Patel-Schneider P. FaCT and DLP. Automated Reasoning with Analytic Tableaux and Related Methods. 1998

[41]. Horrocks I, Sattler U. A Description Logic with Transitive and Inverse Roles and Role Hierarchies. Journal of Logic and Computation. 1999; 9(3):385–410.

[42]. Horrocks I, Sattler U. A Tableau Decision Procedure for SHOIQ. Journal of Automated Reasoning. 2007; 39(3):249–276.

[43]. Hu W, Qu Y. Falcon-ao: A practical ontology matching system. Journal of Web Semantics: Science, Services and Agents on the World Wide Web. 2008; 6(3):237–239.

[44]. Jiménez-Ruiz, E.; Grau, BC.; Sattler, U.; Schneider, T.; Llavori, RB. Safe and Economic Re-Use of Ontologies: A Logic-Based Methodology and Tool Support. 5th European Semantic Web Conference; Springer-Verlag; 2008. p. 185-199.LNCS 5021

[45]. Kalyanpur A, Parsia B, Sirin E, Grau BC, Hendler JA. Swoop: A Web Ontology Editing Browser. Journal of Web Semantics. 2006; 4(2):144–153.

[46]. Knublauch, H.; Fergerson, RW.; Noy, NF.; Musen, MA. The Protégé OWL Plugin: An Open Development Environment for Semantic Web Applications. International Semantic Web Conference; Springer-Verlag; 2004. p. 229-243.3298 LNCS

[47]. Konev, B.; Lutz, C.; Walther, D.; Wolter, F. Formal Properties of Modularization. In: Stuckenschmidt, H.; Spaccapietra, S., editors. Springer's LNCS Monograph on Ontology Modularization. Springer; 2008.

[48]. Konev, B.; Lutz, C.; Walther, D.; Wolter, F. Logical Difference and Module Extraction with CEX and MEX. 21st International Workshop on Description Logics; 2008.

[49]. Konev, B.; Lutz, C.; Walther, D.; Wolter, F. Semantic Modularity and Module Extraction in Description Logics. In: Ghallab, M.; Spyropoulos, CD.; Fakotakis, N.; Avouris, N., editors. 18th European Conference on Artificial Intelligence; IOS Press; 2008. p. 55-59.

[50]. Krötzsch M, Vrandecic D, Völkel M, Haller H, Studer R. Semantic Wikipedia. Journal of Web Semantics. 2007; 5(4):251–261.

[51]. Kunierczyk W. Taxonomy-based Partitioning of the Gene Ontology. Journal of Biomedical Informatics. 2008; 41(2):282–292. [PubMed: 17921072]

[52]. Lambrix P, Tan H. SAMBO: A System for Aligning and Merging Biomedical Ontologies. Web Semantics: Science, Services and Agents on the World Wide Web. 2006; 4(3):196–206.

[53]. Lutz, C.; Walther, D.; Wolter, F. Conservative Extensions in Expressive Description Logics. 20th International Joint Conference on Artificial Intelligence; 2007. p. 453-458.

[54]. Meilicke, C.; Stuckenschmidt, H.; Tamilin, A. Repairing Ontology Mappings. 22nd AAAI Conference on Artificial Intelligence; AAAI Press; 2007. p. 1408-1413.

[55]. Motik, B.; Shearer, R.; Horrocks, I. Optimized Reasoning in Description Logics Using Hypertableaux. 21st International Conference on Automated Deduction; Springer-Verlag; 2007. p. 67-83.LNCS 4603

[56]. Noy, NF.; Chugh, A.; Liu, W.; Musen, MA. A Framework for Ontology Evolution in Collaborative Environments. 5th International Semantic Web Conference; Springer-Verlag; 2006. p. 544-558.LNCS 4273

[57]. Noy, NF.; Musen, MA. Specifying Ontology Views by Traversal. 3rd International Semantic Web Conference; Springer-Verlag; 2004. p. 713-725.LNCS 3298

[58]. Patel, C.; Cimino, JJ.; Dolby, J.; Fokoue, A.; Kalyanpur, A.; Kershenbaum, A.; Ma, L.; Schonberg, E.; Srinivas, K. Matching Patient Records to Clinical Trials Using Ontologies. 6th International Semantic Web Conference; Springer-Verlag; 2007. p. 816-829.LNCS 4825

[59]. Pathak, J.; Jiang, G.; Dwarkanath, SO.; Buntrock, JD.; Chute, CG. Adopting Graph Traversal Techniques for Context-Driven Value Sets Extraction from Biomedical Terminologies. 2nd IEEE International Conference on Semantic Computing; IEEE CS Press; 2008. p. 460-467.

[60]. Pera MS, Ng Y-K. Utilizing Phrase-Similarity Measures for Detecting and Clustering Informative RSS News Articles. Integrated Computer-Aided Engineering. 2008; 15(4):331–350.

[61]. Rector AL, Napoli A, Stamou G, et al. Report on Modularization of Ontologies. Technical report, Knowledge Web Deliverable D2.1.3.1. 2005

[62]. Schmidt-Schaubßand M, Smolka G. Attributive concept descriptions with complements. Artificial Intelligence. 1991; 48(1):1–26.

[63]. Seidenberg, J.; Rector, AL. Web Ontology Segmentation: Analysis, Classification and Use. 15th International Conference on World Wide Web; ACM Press; 2006. p. 13-22.

[64]. Serafini, L.; Tamilin, A. DRAGO: Distributed Reasoning Architecture for the Semantic Web. 2nd European Semantic Web Conference; Springer-Verlag; 2005. p. 361-376.LNCS 3532

[65]. Sirin E, Parsia B, Grau BC, Kalyanpur A, Katz Y. Pellet: A Practical OWL-DL Reasoner. Journal of Web Semantics. 2007; 5(2):51–53.

[66]. Staab, S.; Studer, R., editors. International Handbooks on Information Systems. Springer; 2008. Handbook on Ontologies.

[67]. Stuckenschmidt, H.; Klein, M. Integrity and Change in Modular Ontologies. 18th International Joint Conference on Artificial Intelligence; Morgan Kaufmann; 2003. p. 900-905.

[68]. Stuckenschmidt, H.; Klein, MCA. Structure-Based Partitioning of Large Concept Hierarchies. 3rd International Semantic Web Conference; Springer-Verlag; 2004. p. 289-303.LNCS 3298

[69]. Suntisrivaraporn, B. Module Extraction and Incremental Classification: A Pragmatic Approach for EL^+ Ontologies. 5th European Semantic Web Conference; Springer-Verlag; 2008. p. 230-244.

[70]. Tao X, Li Y, Nayak R. A Knowledge Retrieval Model using Ontology Mining and User Profiling. Integrated Computer-Aided Engineering. 2008; 15(4):313–329.

[71]. Wang, Y.; Bao, J.; Haase, P.; Qi, G. Evaluating Formalisms for Modular Ontologies in Distributed Information Systems. 1st International Conference on Web Reasoning and Rule Systems; Springer-Verlag; 2007. p. 178-193.LNCS 4524

[72]. Yusof Y, Rana OF. Combining Structure and Function-based Descriptors for Component Retrieval in Software Digital Libraries. Integrated Computer-Aided Engineering. 2008; 15(4): 279–296.

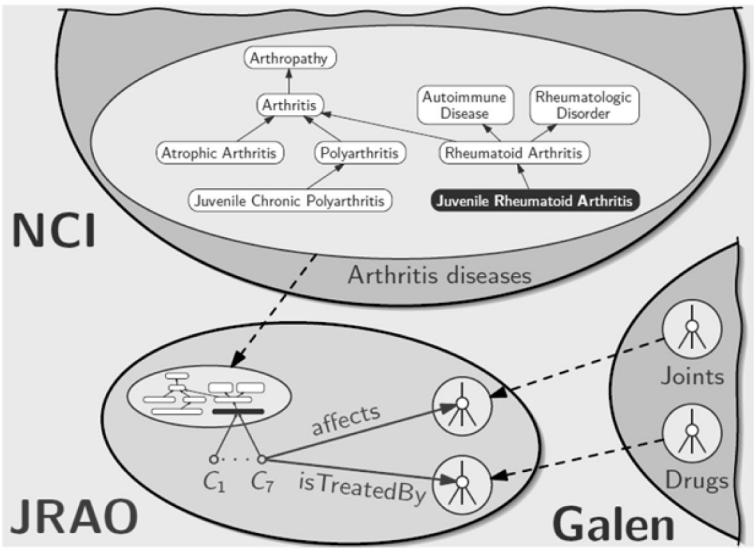

Fig. 1.
Modular ontology example.

$$\mathsf{Human} \sqsubseteq \mathsf{Mammal} \tag{1}$$

$$\mathsf{Woman} \sqsubseteq \mathsf{Human} \sqcap \forall \mathsf{eats}.\mathsf{Food} \tag{2}$$

$$\mathsf{Woman}(\mathsf{Laura}) \tag{3}$$

$$\mathsf{eats}(\mathsf{Laura},\mathsf{Pizza}) \tag{4}$$

Fig. 2.
Example ontology expressed in DL $\mathcal{ALC}$.

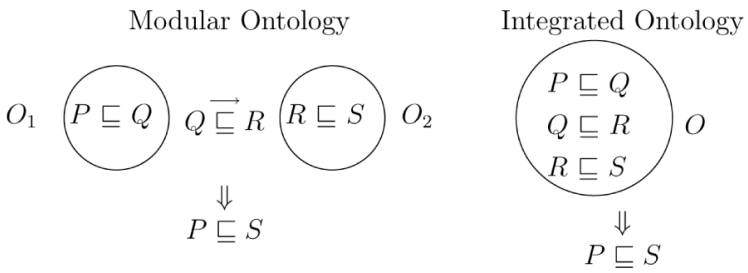

Fig. 3.
Exact reasoning example.

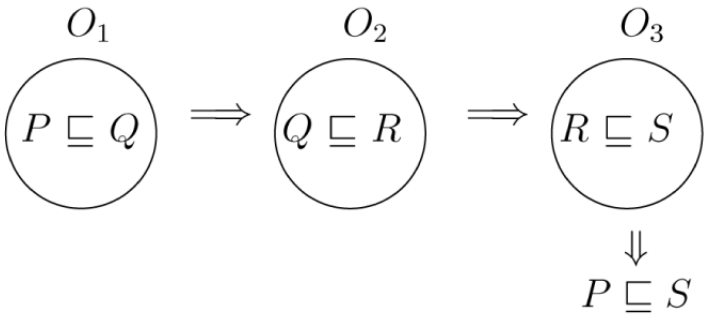

Fig. 4.
Knowledge transitivity example.

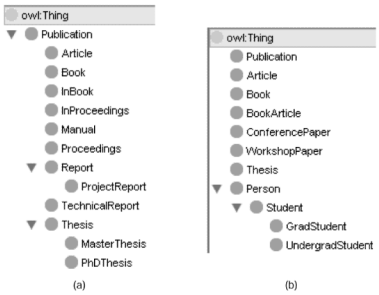

Fig. 5.
Sample class hierarchies.

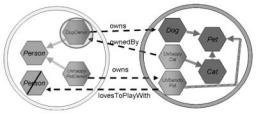

Fig. 6.
$\mathcal{E}$-Connections example.

Medical ontology T:

$$\text{Cystic\_Fibrosis} \equiv \text{Fibrosis} \sqcap \exists \text{located\_In.Pancreas} \sqcap \exists \text{has\_Origin.Genetic\_Origin}$$

$$\text{Genetic\_Fibrosis} \equiv \text{Fibrosis} \sqcap \exists \text{has\_Origin.Genetic\_Origin}$$

$$\text{Genetic\_Fibrosis} \equiv \text{Fibrosis} \sqcap \exists \text{located\_In.Pancreas}$$

$$\text{Genetic\_Fibrosis} \sqsubseteq \text{Genetic\_Disorder}$$

$$\text{DEFBI\_Gene} \sqsubseteq \text{Immuno\_Protein\_Gene} \sqcap \text{associated\_With.Cystic\_Fibrosis}$$

Research projects ontology R:

$$\text{Genetic\_Disorder\_Project} \equiv \text{Project} \sqcap \exists \text{has\_Focus.Genetic\_Disorder}$$

$$\text{Cystic\_Fibrosis\_Project} \sqsubseteq \text{Project} \sqcap \exists \text{has\_Focus.Cystic\_Fibrosis}$$

Fig. 7.
Conservative extensions example.

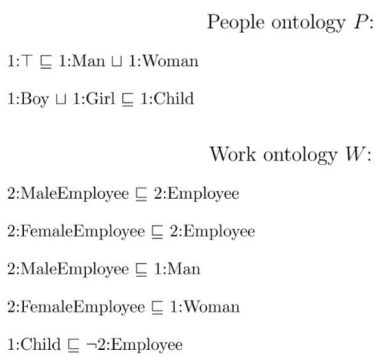

Fig. 8.
Package-based description logics example.

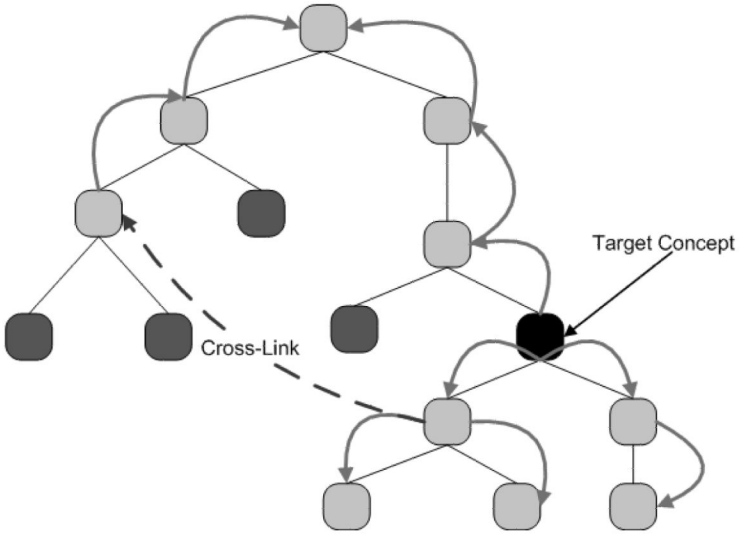

Fig. 9.
GALEN segmentation algorithm.

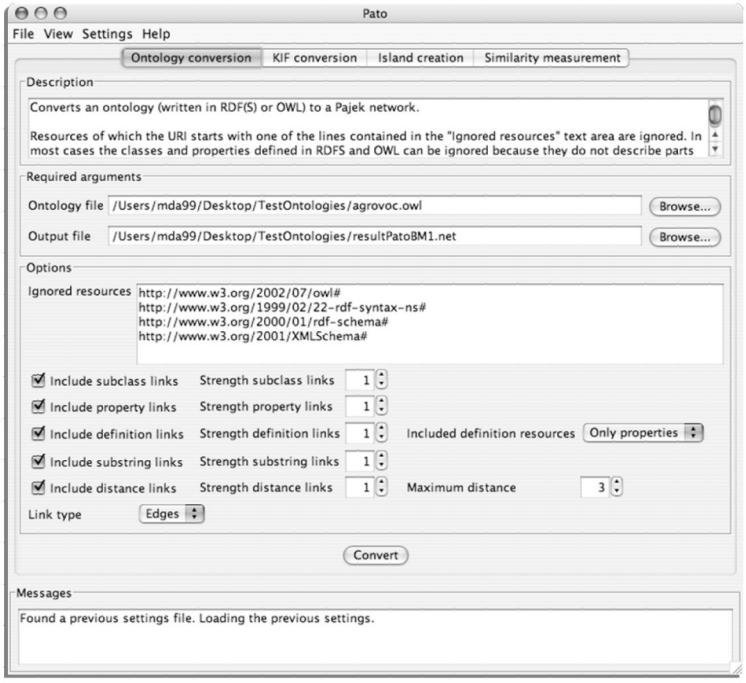

Fig. 10.
PATO: Partitioning tools for ontologies.

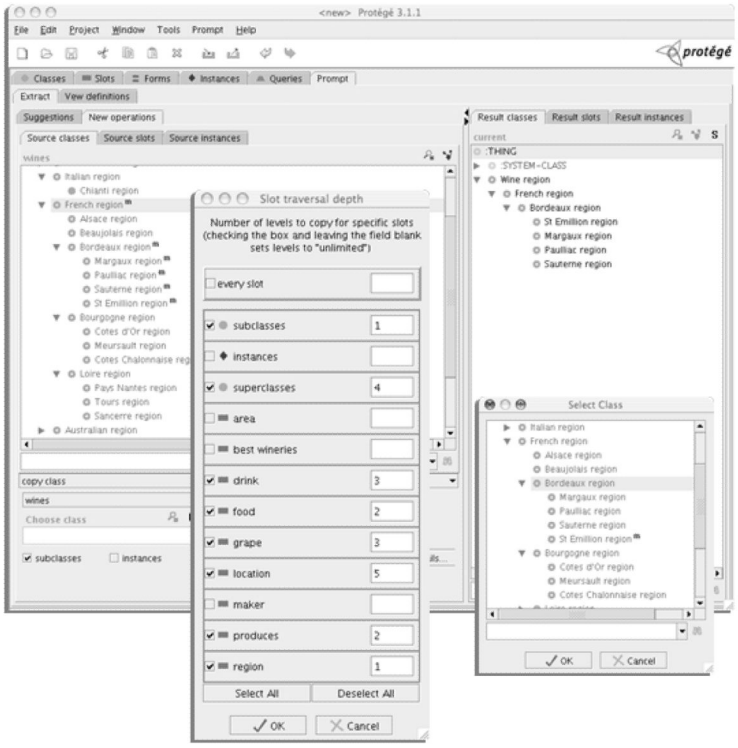

Fig. 11.
Prompt traversal views.

Table 1

Syntax and semantics of $\mathcal{ALC}$ description logic

Constructor	Syntax	Semantics
Top	$(\top)^{\mathrm{I}}$	Δ^{I}
Bottom	$(\bot)^{\mathrm{I}}$	$\emptyset$
Atomic Concept	C^{I}	$C^{\mathrm{I}} \subseteq \Delta^{\mathrm{I}}$
Abstract Role	R^{I}	$R^{\mathrm{I}} \subseteq \Delta^{\mathrm{I}} \times \Delta^{\mathrm{I}}$
Intersection	$(C \sqcap D)^{\mathrm{I}}$	$C^{\mathrm{I}} \cap D^{\mathrm{I}}$
Union	$(C \sqcup D)^{\mathrm{I}}$	$C^{\mathrm{I}} \cup D^{\mathrm{I}}$
Negation	$(\neg C)^{\mathrm{I}}$	$\Delta^{\mathrm{I}} \setminus C^{\mathrm{I}}$
Value Restriction	$\forall\ R.C$	$\{a \in \Delta^{\mathrm{I}} \mid \forall\ b, (a, b) \in R^{\mathrm{I}} \rightarrow b \in C^{\mathrm{I}}\}$
Existential Quantification	$\exists\ R.C$	$\{a \in \Delta^{\mathrm{I}} \mid \exists\ b, (a, b) \in R^{\mathrm{I}} \wedge b \in C^{\mathrm{I}}\}$