Modularization: a Key for the Dynamic Selection of Relevant Knowledge Components

Mathieu d'Aquin, Marta Sabou, and Enrico Motta

Knowledge Media Institute (KMi)
The Open University, Milton Keynes, United Kingdom
{m.daquin, r.m.sabou, e.motta}@open.ac.uk

Abstract. Ontology selection is crucial to support knowledge reuse on the ever increasing Semantic Web. However, applications that rely on reusing existing knowledge often require only relevant parts of existing ontologies rather than entire ontologies. In this paper we investigate how modularization can be integrated with ontology selection techniques. Our contribution is twofold. On the one hand we extend a selection technique with a modularization component. On the other hand we design and implement a modularization algorithm which, unlike many existing approaches, is tightly integrated in a concrete tool.
Keywords: ontology modularization, ontology selection

1 Introduction

The accelerated growth of the Semantic Web in the last years has the potential to cause a paradigm shift in the way that Semantic Web applications are developed. Indeed, the trend is to migrate from knowledge base like applications relying on a single ontology chosen at design time, towards tools that harvest collective intelligence by dynamically integrating online available knowledge [1, 2].

To facilitate knowledge reuse in such a scenario, tools supporting the dynamic selection of reusable knowledge components are crucial. Indeed, what is essential for applications that aim to reuse online knowledge is not so much obtaining one or several ontologies from the Web, but rather getting the relevant piece of knowledge to be used for the task they are performing. This requirement becomes evident when taking into account that several large ontologies are available online (e.g., SUMO), and that, due to their broad domain coverage, they are likely to be often selected. Unfortunately, existing selection techniques return entire ontologies that satisfy criteria like the coverage of a set of query terms and do not consider a more fine grained selection of knowledge [3].

A solution to this problem is to extend selection techniques with modularization capabilities. As such, we are migrating from monolithic ontology selection towards selecting bits of relevant knowledge. Throughout the paper we use the term *knowledge selection* to refer to this process that integrates ontology selection with modularization for retrieving relevant ontology modules. While a variety of modularization techniques exist (see recent surveys in [4, 5]), a general characteristic is that these techniques have been developed independently of the needs of concrete applications. As a consequence, most modularization techniques have been poorly evaluated.

In this paper we investigate using modularization in the concrete scenario of knowledge selection. There are two novel aspects of this work. First, from the perspective of ontology selection, we propose a more fine grained selection of online knowledge. The second aspect, which is the focus of this paper, is that of adapting existing modularization techniques in a concrete scenario that allows their proper evaluation. We propose a modularization algorithm which is based on the *traversal paradigm*, requires no user interaction, poses no assumptions on the ontologies that can be modularized and relies on inferences during the modularization process. Note that the novelty of this algorithm is not so much the actual method that it uses but rather the way in which it satisfies the requirements imposed by our application.

We start by better defining the problem at hand, i.e., providing a more fine grained knowledge selection (Section 2). In Section 3, we overview existing modularization techniques and characterize them from several perspectives that hint on their suitability for being integrated with other applications. Then, based on this view, we identify a set of requirements that a modularization technique should fulfill in the context of selection (Section 4). Our observations are used as design principles for building our algorithm, defined in Section 5. In Section 6 we discuss preliminary experimental results and conclude with a discussion and future work.

2 From Ontology Selection to Dynamic Knowledge Selection

The goal of this section is to better define the scenario of dynamic knowledge selection and to justify the need for modularization. To ground our analysis, we start by describing a concrete tool that relies on dynamic knowledge selection (Section 2.1). Then, we abstract a generic schema for performing dynamic knowledge selection (Section 2.2) that allows to better understand the role of the modularization algorithm in the whole process.

2.1 Magpie: a Scenario for Dynamic Knowledge Selection

Magpie [6] is a Semantic Web browser which helps users to quickly make sense of the information provided by a Web page by allowing them to visually access the semantic data associated with that page. Available as a browser plugin in which a set of concepts are displayed (e.g., *Researcher, Technology*), Magpie can identify instances of these concepts in the current Web page and highlight them with the color associated to each concept. Core to Magpie is an instantiated ontology that contains the information needed to identify the relevant instances in Web pages. Magpie can be easily adapted to work with any ontology. The user simply needs to select an ontology from a list of known ontologies. However, Magpie can only explore one ontology at a time, meaning that the scope of the sense making is limited to the domain described by that ontology.

In our current work we are extending Magpie towards open semantic browsing in which the tool automatically selects and combines online ontologies and their associated metadata relevant to the content of the current page. As such, the user is relieved from manually choosing a suitable ontology every time he wishes to browse new content. Such an extension of our tool relies on mechanisms that can not only dynamically select appropriate ontologies from the Web, but can also extract from these ontologies the relevant and useful parts to describe concepts and instances in the current Web page.

Concretely, given a Web page, Magpie requires a concise domain model that describes the most important concepts and instances in this page. To achieve this, in a first step, a relevant set of terms describing the Web page (keywords) are extracted by using text processing methods. Then, given these terms, a selection method would dynamically (i.e., while the tool is used by the user rather than during the design of the tool) select and retrieve the appropriate piece of knowledge from ontologies available on the Web. The size of the returned ontology module is crucial since Magpie only needs relevant parts of ontologies, small enough to be visualized within the browser and to be easily interpreted by the user in relation with the current Web page. In the next subsection we describe the generic design of a dynamic knowledge selection system and discuss the role of modularization in this system.

2.2 Dynamic Knowledge Selection

Taking a step back from the concrete Magpie scenario, at a generic level, we expect that a knowledge selection algorithm receives a set of terms and returns an ontology that is usable within an application.

Our previous work and experiences in ontology selection [7] made it clear that modularization may play a crucial role in complementing the current selection techniques. Indeed, selection algorithms tend to run into two major problems. First, if the selection returns a large ontology this is virtually useless for a tool such as Magpie which only visualises a relatively small number of concepts at a time. Unfortunately, in the experiments we have performed large ontologies are often returned (especially WordNet). What is needed instead is that the selection process returns a part (module) of the ontology that defines the relevant set of terms. A second problem relates to the fact that in many cases it is difficult to find a single ontology that covers all terms (we observed this knowledge sparseness phenomenon in [7] and our current systematic experiments in this direction confirm our initial intuitions). However, a combination of one or more ontologies could cover all the query terms. This problem is related to modularization in the sense that it is easier to combine small and focused knowledge modules than ontologies of large size and coverage.

These considerations justify the need to extend selection techniques with modularization capabilities. In Figure 1 we depict the three major generic steps

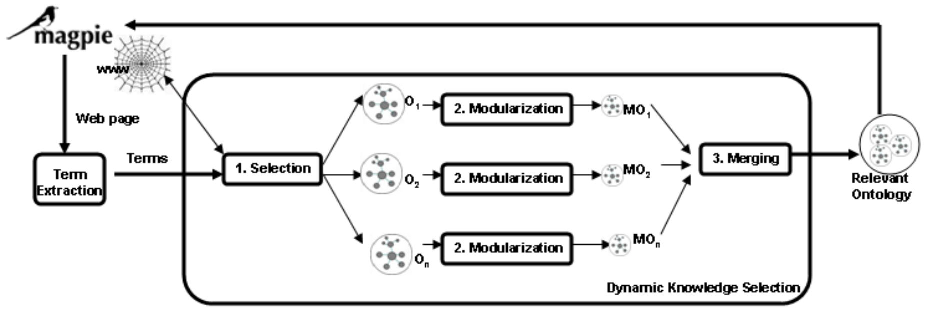

Fig. 1. The knowledge selection process and its use for semantic browsing with Magpie.

of the *knowledge selection* process that integrates ontology selection, modularization and merging.

1. **Selection of relevant ontologies.** In a first step, given a set of terms for which an ontology is required, the selection technique identifies online ontologies or sets of ontologies that cover the given query terms. By coverage we mean that the identified ontologies contain concepts, properties or instances that are semantically related to the query terms. The selection algorithm has been detailed in [7] and is currently being implemented.
2. **Modularization of the selected ontologies.** Given the ontologies discovered in the previous step, a modularization technique is applied on each ontology to identify a module in the ontology that contains relevant knowledge for the query terms that were mapped into that ontology. The design and implementation of this modularization technique is the focus of this paper.
3. **Merging of the relevant ontology modules.** Finally, in the case when the query terms are covered by several different ontologies, the separate modules extracted from each individual ontology need to be merged in a meaningful way to form the final result of the selection. The merging algorithm will be built after the selection and modularization parts are finalized.

As shown by the above description, modularization is tightly integrated in the selection process. Before providing a suitable algorithm we wish to find out which are the concrete requirements that are imposed on the modularization part by this tight integration in the context of another process. In preparation to define these requirements we performed a survey of modularization techniques.

3 A Survey of Ontology Modularization Techniques

The objective of this section is to analyze existing modularization approaches in order to evaluate which of them has the potential to be integrated in our knowledge selection process. The two modularization surveys that we are aware of analyze the field from perspectives that are not suitable for our purpose. The survey in [4] takes a broad view by considering modularization as the process used to obtain modular ontologies both from existing ontologies and when designing a new ontology. Our focus is narrower, as we are interested in extracting modules from existing and potentially large scale ontologies. The survey in [5] is focused towards comparing existing methods to the approach presented by that paper and therefore it is too narrow for our needs. We start our analysis by briefly describing a set of modularization systems according to the three major approaches that they rely on (Section 3.1). In Section 3.2 we define a set of characteristics that define different aspects of modularization and then analyze the techniques described in Section 3.1 from this perspective.

3.1 Modularization Systems

The survey in [5] distinguishes three main categories of modularization approaches based on the technique that is used: query based methods, network partitioning, and extraction by traversal.

Query based methods obtain ontology modules as answers to queries written in a SQL-like query language (e.g., SparQL, RQL). Due to their inspiration drawn from the view mechanism in databases, most of these approaches result in a virtual view. They don't actually extract a proper ontology module, i.e. a materialized and self-contained piece of ontology. These characteristics make this set of techniques inappropriate for our scenario.

Network partitioning based methods consider ontologies as graphs and exploit graph partitioning algorithms to extract structurally relevant modules. In [8], the authors present a method for dividing ontologies into several partitions using an algorithm from network analysis. A dependency graph is first computed between the ontology elements, and these elements are clustered on the basis of criteria like intra and inter-connectivity. [9] considers the partitioning of a knowledge base for improving the scalability of theorem proving. A graph is built on the co-occurrence of symbols in the axioms of a logical theory, and tree-decomposition algorithms are used on this graph to generate tightly connected clusters of symbols. The created modules are then used in a so-called *partition-based theorem proving* mechanism, exploiting the distribution of the knowledge in multiple modules.

Traversal based modularization methods start from one or several elements of the ontology (concepts, properties, or individuals), and include in the module the concepts, properties and individuals that are (generally syntactically) linked to these elements. For example, the system described in [10] extracts a part of the ontology around a given concept, by recursively following the concept properties until a given distance is reached. [5] also starts from a concept of the ontology and extracts related elements on the basis of concept subsumption and OWL restrictions. Some optional filters can be used to reduce the size of the generated module. In [11], modules are extracted from an ontology with respect to the user requirements. These requirements are expressed by labels indicating for each ontology element whether or not it must be selected. In [12], the author defines a viewpoint as being a sub-part of an ontology that only contains the knowledge concerning a given sub-vocabulary (a set of concept and property names). The computation of a viewpoint is based on the definition of a viewpoint dependent subsumption relation. Finally, [13] does not entirely comply with the traversal paradigm, except that it relies on the ontology structure. It gives a formal approach for creating modular ontologies from OWL ontologies, on the basis of ε-Connections. The presented method is based on the computation of the so-called *structural compatibility* and *semantic compatibility* between the original ontology and the generated ε-Connections.

3.2 Characteristics of Modularization Systems

In this section we define a set of characteristics that allow a more fine-grained analysis of modularization techniques than it has been done by existing surveys. In particular, we are interested in aspects that denote how these techniques could be integrated within other systems or processes. This should allow us to evaluate their usefulness for knowledge selection as well as to derive concrete requirements for knowledge selection. We consider the following characteristics:

Modularization criteria denote the characteristics that the resulting modules should fulfill, such as, coverage of a sub-vocabulary, reduced size.

Assumptions on the ontology denote the assumptions that the modularization techniques rely on, such as, complying to a certain representation language, providing a certain level of quality or axiomatization.

Level of user interaction describes the user involvement in the modularization process, and can range from no actions required by from the users, to the user interacting with the system by adding new requirements or fine-tuning some variables.

Taking inferences into account refers to the modularization process using inference mechanisms or providing modules that can be reasoned upon.

Use of modules refers to the ways modules are used after being obtained. We distinguish two aspects here: 1) the process of combining modules and 2) the use and evaluation of modules in concrete applications.

Modularization criteria. The criteria of modularization often depend on the purpose of the modularization, for example, to enhance reusability and scalability. The goal of the work in [9] is to distribute the theorem proving mechanism among several knowledge partitions in order to improve its scalability. Therefore, these partitions have to be *loosely coupled and highly cohesive.* In traversal approaches such as [5, 10, 12], the goal is to facilitate the reuse and the interpretation of the knowledge contained in the ontology. The resulting modules are required both to *cover a given sub-vocabulary* and to have an *appropriate size.*

Assumptions on the ontology. Most of the existing approaches rely on some assumptions. For example, those described in [13] and [5] are explicitly made to work on OWL ontologies, whereas [8] can be used either on RDF or OWL but only exploits RDF features. In [5], the ontology is required to be well-designed and to use complex OWL constructs to describe concepts. Moreover, some of the filters used to reduce the size of a module are dependent on elements of the GALEN upper ontology.

Level of user interaction. In many systems the required user entries are limited to the inputs of the algorithm. In certain cases, some numerical parameters can be required [8] or some additional procedures can be manually (de)activated [5]. The technique in [10] has been integrated in the PROTÉGÉ ontology editor to support knowledge reuse during the building of a new ontology. In this case, modularization is an interactive process where the user has the possibility to extend the current module by choosing a new starting point for the traversal algorithm among the *boundary concepts* of the module.

Taking inferences into account. Several systems are designed to allow reasoning with the created modules and so exhibit some interesting properties concerning inferences. For example, [13] and [9] both guarantee that the entailments of the original ontology are also entailed by the set of extracted modules, and vice-versa. To our knowledge, none of the considered techniques are relying on inference mechanisms *during* the modularization process.

Use of modules. Regarding the first aspect of this criteria, that of combining the resulting modules, most approaches materialize a module as a self-contained ontology [5, 10, 12]. Several papers mention formal properties that allow the combination (e.g., union, intersection) of the modules of a particular ontology [10, 12], but none of these papers are considering the combination of modules coming from different ontologies. Regarding the aspect of actually using modules in other applications, we only know of two approaches that make their modules available to reasoners/theorem provers (but not to any other applications). The modules extracted in [13] are linked together using ε-Connections and aim at being used in a reasoner. In a similar way, the knowledge base partitions created by the approache of [9] are used in a dedicated theorem prover.

4 Modularization Requirements

While existing modularization approaches are agnostic to the needs of applications, as is shown in our survey, our goal is to define a modularization mechanism that fulfills the requirements and constraints imposed by a particular application, i.e., dynamic and automatized knowledge selection, and a particular scenario for this application, i.e., semantic browsing with Magpie (cf. Section 2). In this section we elaborate on these requirements which are established in terms of the characteristics described in Section 3.2.

R1. Modularization criteria reflect selection criteria. As explained in Section 2, ontology modularization is tightly integrated with ontology selection and its goal is to reduce the selected ontology to the relevant sub-part(s). As a result the criteria for the modularization of an ontology directly depend on the criteria that have lead to the selection of that ontology.

In our semantic browsing scenario, the knowledge selection process should retrieve ontology modules that are related to a given Web page. Keywords are extracted from this page and ontologies are selected according to their relevance to these terms. The result of the ontology selection is a set of mappings from the initial terms to elements (concepts, properties or individuals) of the selected ontology (see [7] for a detailed description of the ontology selection mechanism). Therefore, the goal of the modularization process is to extract from the ontology the smallest piece of knowledge related to and covering the mapped elements.

R2. No assumption on the ontology. The goal of our knowledge selection process is to retrieve relevant knowledge components from ontologies available on the Web. Online available ontologies are heterogeneous in several aspects: they can be expressed in different representation languages (OWL, RDFS, DAML), they can have different quality (ranging from well-designed DL-based ontologies such as Galen to simple taxonomies), and different size. In the knowledge selection scenario it is impossible to impose any assumptions on the characteristics of the input ontology.

R3. Minimal user interaction. Depending on the envisaged application scenario, the modularization mechanism may require different levels of user interaction. In our semantic browsing scenario, and in general for knowledge selection, we cannot expect the user to know anything concerning the selected ontology or about the mechanism used to extract relevant knowledge.

R4: Ensure output covers both relevant explicit and implicit knowledge. One of the advantages of semantic metadata is that it can be reasoned upon and implicit knowledge can be derived. However, a side effect of modularization is that one might exclude parts of the ontology that could lead to deriving such implicit knowledge. Ideally, the knowledge entailed by the selected ontology should also be contained in the extracted module. The method presented in [5] tries to fulfill this criterion by including all related ontology elements to the concepts that have to be covered by the module. However, this strategy leads to including a lot of extra information and obtaining large sized modules.

Summary

Table 1 summaries the correspondences between the four requirements expressed above and the systems surveyed in Section 3. Network partitioning approaches (as well as the system presented in [13]) score low on fulfilling the first requirement because they consider modularization from a general perspective and do not incorporate criteria such as those used by the selection algorithm. In [5], some strong assumptions are made about the input ontology that cannot be guaranteed in an open scenario like the one of knowledge selection. Considering that our goal is to achieve a fully *automatized* knowledge selection, interactive processes like the one detailed in [10] cannot be envisaged. Finally, unlike the system in [12], it is important to take into account the inferences that can be drawn from the considered ontology in order to extract explicit as well as implicit relevant knowledge.

The first, obvious conclusion of this analysis is that the modularization mechanism for knowledge selection should be based on a traversal approach. However, it also shows that these techniques cannot be directly reused, because they do not completely match all our requirements. Therefore, we designed an algorithm

Requirements	**Network**		**Traversal**				
	[8]	[9]	[5]	[10]	[11]	[12]	[13]
Rely on the selection criteria (R1.)	0	0	1	2	2	2	0
No assumptions on ontology(R2.)	1	0	0	3	2	3	1
Minimal user interaction (R3.)	2	3	2	0	3	3	3
Using inference (R4.)	1	3	3	0	0	0	3

Table 1. Overview of modularization approaches and the level to which they match our requirements. (0: no, 1: weak, 2: good, 3: complete).

inspired from some of these approaches (especially [5], [10] and [12]), and adapting it to the needs of dynamic knowledge selection.

5 Modularization Algorithm

In this section we propose a modularization algorithm dedicated to dynamic knowledge selection, i.e. designed to fulfill the previously defined requirements. It is a fully automatized process relying on a reasoner to extract implicit (i.e. inferred) knowledge and taking into account different levels of description, from simple taxonomies to complex knowledge representation structures. We first introduce the notations used to denote ontologies and their elements, detail the algorithm itself, and finally discuss some of its characteristics with respect to our requirements.

5.1 Notations

An ontology O contains a set $C(O)$ of concepts, a set $P(O)$ of properties, a set $I(O)$ of individuals, and a set $A(O)$ of assertions, relating elements of $C(O)$, $P(O)$ and $I(O)$. Assertions can either be axioms, introducing concepts and properties, or facts, introducing individuals. For examples, a concept inclusion axiom $\mathtt{C} \sqsubseteq \mathtt{D}$ indicates that the concept $\mathtt{C}$ is a sub-concept of $\mathtt{D}$, $\mathtt{C(a)}$ is a fact indicating that the individual $\mathtt{a}$ is an instance of a concept $\mathtt{C}$, and $\mathtt{p(a, b)}$ is a fact indicating that the property $\mathtt{p}$ relates the individuals $\mathtt{a}$ and $\mathtt{b}$.

A concept can either be described by its name or by an expression using constructs like restrictions ($\exists \mathtt{p.C}$, $\forall \mathtt{p.C}$), conjunctions ($\mathtt{C} \sqcap \mathtt{D}$), etc. In this paper, concept expressions are written using the description logic syntax.

The most common inference mechanism in OWL is the subsumption test. It consists in checking if a concept $\mathtt{C}$ is a sub-concept of another concept $\mathtt{D}$. We use $O \models \mathtt{C} \sqsubseteq \mathtt{D}$ to denote that it can be inferred from the ontology O that $\mathtt{C}$ is subsumed by $\mathtt{D}$, i.e. that $\mathtt{C}$ is a sub-concept of $\mathtt{D}$. In general, this notation can be used with any assertion. For example, $O \models \mathtt{C(a)}$ means that it can be inferred that $\mathtt{a}$ is an instance of $\mathtt{C}$.

The modularization mechanism presented here is designed to be integrated in a knowledge selection process, as described in Section 2. Its goal is to extract the part of the ontology O that is related to the terms that have lead to the selection of O. Therefore, the input of this algorithm is the ontology O, and a sub-vocabulary SV of O that the extracted module should cover. SV is described by a set $C(SV) \subseteq C(O)$ of concept names, a set $P(SV) \subseteq P(O)$ of property names, and a set $I(SV) \subseteq C(O)$ of individual names.

The following description of the modularization algorithm is illustrated on a toy example using the ontology presented in Figure 2(a). In this ontology, Samantha is described as being a woman (i.e., a person of the female sex) who has a child named Tabatha. A parent is a person having at least a child. A mother is a parent and a woman. We are interested in extracting from this ontology the knowledge concerning Samantha and the concept of mother. Therefore, the sub-vocabulary SV used as an input for the algorithm is: $C(SV) = \{\texttt{Mother}\}$, $P(SV) = \emptyset$, and $I(SV) = \{\texttt{samantha}\}$.

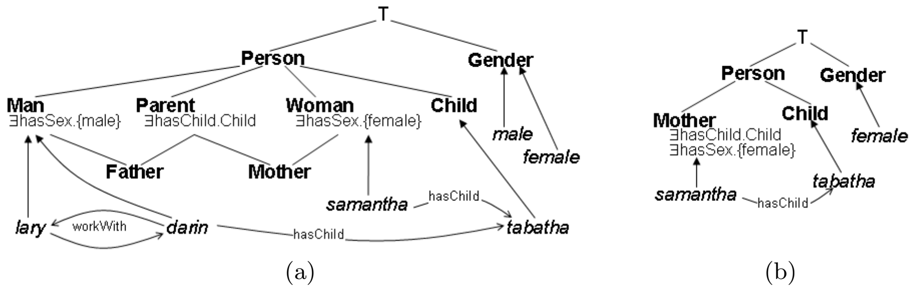

Fig. 2. The original ontology (a) and the resulting module (b).

The result of the modularization process is a module M, supposed to be a reusable and self-contained piece of knowledge related to SV. M is then described in a similar way as the ontology O, by a set $C(M) \subseteq C(O)$ of concepts, a set $P(M) \subseteq P(O)$ of properties, a set $I(M) \subseteq I(O)$ of individuals, and a set $A(M) \subseteq A(O)$ of assertions.

5.2 Details of the Algorithm

The actual modularization algorithm consists in computing $C(M)$, $P(M)$, $I(M)$ and $A(M)$ recursively, in a fix-point like algorithm. The following definitions clarify which concepts, properties, instances and assertions should be included. The overall idea is to include in the module the elements that SV relates, either directly or indirectly, explicitly or implicitly.

$C(M)$, the set of concepts to be contained by the module, corresponds to the set of concepts in O that are used to define elements of the sub-vocabulary SV, or that relate other concepts of $C(M)$ in the ontology hierarchy (i.e. they are the most specific common super-concepts of other concepts in $C(M)$).

Definition 1. *$C(M)$ contains a concept $\mathtt{C}$ iff:*

c1. *$\mathtt{C}$ is a concept name and $\mathtt{C} \in C(SV)$ or*
c2. *$\mathtt{C}$ is a concept name and it can be inferred that $\mathtt{C}$ is the most specific common super-concept of two concepts in C(M)*[1]
c3. *$\mathtt{C}$ is a concept name and it can be inferred that $\mathtt{C}$ is the most specific concept of an individual in I(M) or*
c4. *$\mathtt{C}$ is a concept expression and there is a concept $\mathtt{D} \in C(M)$ such that $O \models \mathtt{D} \sqsubseteq \mathtt{C}$ or*
c5. *there exists a concept $\mathtt{D} \in C(M)$ such that $\mathtt{D}$ is a concept expression of one of the following forms: $\exists \mathtt{p.C}$, $\forall \mathtt{p.C}$, $\mathtt{C} \sqcap \mathtt{E}$, $\mathtt{C} \sqcup \mathtt{E}$ or $\neg \mathtt{C}$*

In our example, the `Mother` concept is included because it is in SV (*c1.*), but also because it can be inferred that `samantha` is an instance of `Mother` and so that `Mother` is its most specific concept (*c3.*). The *c4.* part of the definition leads to the inclusion of the expressions $\exists$`hasChild.Child` and $\exists$`hasSex.{female}`, inherited by `Mother` from the definitions of `Woman` and `Parent`. Note that, to avoid obtaining a large module, the `Woman` and `Parent` concepts themselves are not included explicitly, though their definitions are inherited by `Mother`. Then, because of the *c5.* part, `Child` is included in the module, as well as `Person` that is the most specific super-concept of `Child` and `Mother` (*c2.*).

Similarly, a property is included in $P(M)$ if it is used to define an element of SV, or if it is the most specific super-property of other properties in $P(M)$.

Definition 2. *$P(M)$ contains a property $\mathtt{p}$ iff:*

p1. *$\mathtt{p} \in P(SV)$ or*
p2. *it can be inferred that $\mathtt{p}$ is the most specific common super-property of two properties in $P(M)$ or*
p3. *$O \models \mathtt{p}(\mathtt{a}, \mathtt{b})$ and $\mathtt{a} \in I(M)$ or*
p4. *there exists a concept $\mathtt{C} \in C(M)$ such that $\mathtt{C}$ is a concept expression of one of the following forms: $\exists \mathtt{p.D}$, $\forall \mathtt{p.D}$, $\exists \mathtt{p}.\{a\}$, $\leq_n \mathtt{p}$, $\geq_n \mathtt{p}$ or $=_n \mathtt{p}$*

In our example ontology, only the `hasChild` and `hasSex` properties are kept for the module: they are used in the previously included concept expressions (*p4.*) and `samantha` relates to other individuals using these properties (*p3.*).

Following the definition of $C(M)$ and $P(M)$, $I(M)$ should include any individual of O that is used to describe elements of SV, even if indirectly.

Definition 3. *$I(M)$ contains an individual $\mathtt{a}$ iff:*

i1. *$\mathtt{a} \in I(SV)$ or*
i2. *$O \models \mathtt{p}(\mathtt{b}, \mathtt{a})$ and $\mathtt{b} \in I(M)$ or*

[1] More precisely, among the concept names that are super-concepts of both of these two concepts in the ontology hierarchy, `C` is one of the most specific. This can be easily computed on the basis of the subsumption test provided by a reasoner.

i3. there exists a concept $\texttt{C} \in C(M)$ *such that* $\texttt{C}$ *is a concept expression of the form* $\{\texttt{a}, \ldots\}$

Three individuals are included in the result of our example modularization: `samantha` because it is in SV (*i1.*), `tabatha` because it is related to `samantha` by the `hasChild` property (*i2.*), and `female` because it is used in an included concept expression (*i3.*).

Finally, the relations between elements of the module that are known in the ontology must be kept in the module. This means that $A(M)$ contains the assertions that can be inferred from O and that link elements of M. For example, $\texttt{C} \sqsubseteq \texttt{D} \in A(M)$ if and only if $O \models \texttt{C} \sqsubseteq \texttt{D}$, $\texttt{C} \in C(M)$, and $\texttt{D} \in C(M)$.

Definition 4. *$A(M)$ contains an assertion A iff $O \models A$ and A relates elements of $C(M)$, $P(M)$ or $I(M)$.*

The assertions of $A(M)$ then re-build all the links between concepts, properties and individuals of the module, as shown in Figure 2(b). Note that $A(M)$ can be computed at the end since it does not influence any other definition.

5.3 Characteristics of the Algorithm and of the Resulting Modules

The first requirement in Section 4 states that the modularization criterion is based on the criterion for selection, meaning that the resulting module should be the smallest part of the ontology that covers the terms used during the ontology selection. Following the traversal approach, our algorithm recursively inspects the ontology expressions to include the elements that participate to the definition of these terms. Unlike the algorithm in [5], all the super-concepts of a selected concept are not necessarily included (only the ones that directly relate concepts of the module, i.e. the most specific common concepts). We only keep the required structure and remove from the hierarchy the intermediary concepts that do not participate to the semantic definition of the considered terms.

The second requirement emphasizes that no assumptions should be made regarding ontology types. For this reason, the definitions on which the algorithm relies are made of different parts, taking into account different levels of description: the algorithm can work with the simplest ontologies but can also explore complex structures in expressive representation languages when they are available. In the current implementation, we consider only the RDFS and OWL formalisms but since these are the most common ontology languages for the Semantic Web, this choice is not too restrictive. Moreover, this modularization process is in itself modular in the sense that parts of the definitions can easily be added, removed or replaced to optimize it for a particular kind of ontology, or to match the requirements of other scenarios.

To fulfill the requirement of having no user interaction, we base our modularization solely on inputs that can be provided by the selection mechanism that proceeds it. Finally, according to our fourth requirement, the extracted module should not only include the elements that explicitly (i.e. syntactically) participate to the definition of the input terms, but also elements that are implicitly related by the mean of inferences. We fulfill this requirement by relying on inferences *during* the modularization process. For example, in Definition 2, the *p3.* part may rely on declared as well as inferred relations between individuals. Accordingly, an interesting property of our algorithm is that anything that can be entailed from the original ontology concerning the relations between elements of the input sub-vocabulary can still be entailed from the module. This property may be particularly interesting in applications where modularization is used to reason about the terms of the sub-vocabulary.

6 Discussion

The proposed algorithm has been implemented in Java, using the JENA API[2]. This program can be tuned for different levels of inference support, from the simplest rule based mechanism provided by JENA, to complete description logic reasoning using the PELLET OWL reasoner[3]. During implementation we have monitored the performance of the algorithm on a large medical ontology (about 4000 concepts) from the OpenGalen project[4]. For example, when interested in describing a piece of text about lung cancer, we observed that 1) only a part of this big ontology is concerned with cancer and 2) the `Cancer` concept and the `Lung` concept are described in different and distant parts of the concept hierarchy, making the visualization and interpretation of this knowledge difficult. Using our algorithm with a sub-vocabulary containing the terms “cancer”, “lung”, “adenocarcinoma” and “breath”[5], we obtained a concise, focused and self-contained ontology module of about 50 named concepts. This encouraging result demonstrated that the modularization can be interfaced with the selection algorithm and use the same selection criteria.

The aim of this paper was to identified the requirements of dynamic knowledge selection regarding modularization, to analyze how existing modularization approaches can contribute in fulfilling these requirements, and to derive an algorithm dedicated to this particular task. This has led to the design and implementation of a modularization tool inspired from state-of-the-art traversal approaches. As a next step, this tool will be integrated with an ontology selection mechanism and a similar study will be realized concerning the merging part of our process. Moreover, a proper evaluation should be conducted to estimate the quality of our modularization with respect to knowledge selection, as well as in other applications. We believe that the quality of a module, its usefulness, is highly dependent on the scenario in which the modularization is integrated. Existing modularization techniques are based on diverse principles and rely on

[2] http://jena.sourceforge.net/

[3] http://www.mindswap.org/2003/pellet/

[4] http://www.co-ode.org/ontologies/test/breaksmetrics.owl

[5] extracted from the keywords of a web page concerning lung cancer

different assumptions. Therefore, the evaluation and comparison of modularization techniques are rather difficult tasks that require a proper framework, in the form of a benchmark. The design of such an evaluation process, including criteria for evaluating the relevance of diverse modularization techniques with respect to different usage scenarios will be considered as future work.

Acknowledgements

This work was funded by the Advanced Knowledge Technologies (AKT) Interdisciplinary Research Collaboration (IRC), sponsored by the UK Engineering and Physical Sciences Research Council and the Open Knowledge and NeOn projects sponsored by the European Commission as part of the Information Society Technologies (IST) programme.

References

1. Motta, E., Sabou, M.: Next Generation Semantic Web Applications. In: Proc. of the 1st Asian Semantic Web Conference (ASWC). (2006)
2. Motta, E., Sabou, M.: Language Technologies and the Evolution of the Semantic Web. In: Proc. of the 5th International Conference on Language Resources and Evaluation (LREC). (2006)
3. Sabou, M., Lopez, V., Motta, E., Uren, V.: Ontology Selection: Ontology Evaluation on the Real Semantic Web. In: Proc. of the 4th International EON Workshop. (2006)
4. (coordinator) Spaccapieta, S.: Report on Modularization of Ontologies. Knowledge Web Deliverable 2.1.3.1 (2005)
5. Seidenberg, J., Rector, A.: Web Ontology Segmentation: Analysis, Classification and Use. In: Proc. of the World Wide Web Conference (WWW). (2006)
6. Dzbor, M., Domingue, J., Motta, E.: Magpie - towards a semantic web browser. In: Proc. of the Second International Semantic Web Conference (ISWC). (2003)
7. M. Sabou, V.L., Motta, E.: Ontology Selection on the Real Semantic Web: How to Cover the Queens Birthday Dinner? In: Proc. of the European Knowledge Acquisition Workshop (EKAW), Podebrady, Czech Republic (2006)
8. Stuckenschmidt, J., Klein, M.: Structure-Based Partitioning of of Large Concept Hierarchies. In: Proc. of the International Semantic Web Conference (ISWC). (2004)
9. MacCartney, B., McIlraith, S., Amir, E., Uribe, T.: Practical Partition-Based Theorem Proving for Large Knowledge Bases. In: Proc. of the International Joint Conference on Artificial Intelligence (IJCAI). (2003)
10. Noy, N., Musen, M.: Specifying Ontology Views by Traversal. In: Proc. of the International Semantic Web Conference (ISWC). (2004)
11. Bhatt, M., Flahive, A., Wouters, C., Rahayu, W., Taniar, D., Dillon, T.: A Distributed Approach to Sub-Ontology Extraction. In: Proc. of the 18th International Conference on Advanced Information Networking and Applications (AINA). (2004)
12. Stuckenschmidt, H.: Toward Multi-Viewpoint Reasoning with OWL Ontologies. In: Proc. of the European Semantic Web Conference (ESWC). (2006)
13. Grau, B., Parsia, B., Sirin, E., Kalyanpur, A.: Automatic Partitioning of OWL Ontologies Using ε-Connections. In: Proc. of Description Logic Workshop (DL). (2005)