Dynamic Taxonomies for the Semantic Web

Pierre Allard and Sébastien Ferré

IRISA/Université de Rennes 1
Campus de Beaulieu
35042 Rennes cedex, France
piallard@irisa.fr, ferre@irisa.fr

Abstract

The semantic web aims at enabling the web to understand and answer the requests from people and machines. It relies on several standards for representing and reasoning about web contents. Among them, the Web Ontology Language (OWL) is used to define ontologies, i.e. knowledge bases, and is formalized with description logics. In this paper, we demonstrate how dynamic taxonomies and their benefits can be transposed to browse OWL DL ontologies. We only assume the ontology has an assertional part, i.e. defines objects and not only concepts. The existence of relations between objects in OWL leads us to define new navigation modes for crossing these relations. A prototype, ODALISQUE*, has been developed on top of well-known tools for the semantic web.*

1. Introduction

Dynamic taxonomies (DT) have proven their usefulness to browse medium-to-large information bases [6]. Objects can be classified under an arbitrary number of concepts; and concepts are organized in a multi-dimensional taxonomy. This taxonomy supports both expressive querying and flexible navigation. A query is any boolean combination of concepts, and defines a focus over the information base. Every query determines an *extension* as the set of answers to the query, and a *dynamic taxonomy* as the pruning of the taxonomy to those concepts that share objects with that extension. The dynamic taxonomy serves both as a summary of the current focus, and as a set of navigation links to reach other foci. A navigation link combines the current query and a selected concept to form a new query, and hence reach a new focus. DTs enable people with no *a priori* knowledge of an information base to acquire such a knowledge, and to reach relevant objects, as demonstrated by their use in e-commerce applications [7].

An interesting question is whether DTs can be adapted to more complex information bases than object collections and taxonomies. A step in this direction is made by Logical Information Systems (LIS) that use almost arbitrary logics instead of taxonomies [1, 2]. Values and patterns can be used in addition to concept names in object descriptions, queries and dynamic taxonomies; and their organization into a taxonomy is automatically derived by logical inference. However, a persisting limitation is that the information base is essentially a collection of objects that are unrelated to each other, apart from sharing common concepts. The consequence is a biased representation in many applications. For instance, in a bibliographic application, objects can be either documents or authors, but not both.

Description Logics (DL) are logical formalisms that have been adopted to represent and reason about ontologies [4] (domain-specific knowledge bases). They have points in common with DTs such as concepts, subsumption between concepts (similar to taxonomic relation), and instance relation between objects and concepts. There are however several important differences. First, concepts are not only concept names, but complex combinations of concepts by logical connectors. Second, the subsumption relations are derived automatically from a set of axioms modelling the domain. Third, relations (called *roles*) can be set between objects, so that the concepts to which an object belongs depend on other objects to which it is connected.

In this paper, we show how DTs, and LIS, can be extended so as to cope with semantic web ontologies, while retaining all their good properties. In particular, the presence of roles between objects leads us to define new navigation modes. In Section 2 we shortly introduce web ontologies and DL. In Section 3 we redefine the notions of query, extension and dynamic taxonomies that make up a focus. In Section 4 we redefine the navigation links for zoom-in and zoom-out , and we introduce navigation links for crossing roles. Section 5 presents our prototype ODALISQUE. Section 5 concludes and draws perspectives.

2. Web Ontologies

The purpose of OWL is to define ontologies for modelling and reasoning about application domains. OWL is mapped onto the Description Logics (DL). DL provide the required theoretical background, such as proving the consistency of an ontology, or classifying concepts. The OWL standard includes 3 expressivity levels: OWL Lite, OWL DL and OWL Full (in increasing expressivity). In this paper, we are interested in OWL DL, because it is decidable, a good compromise between expressivity and inference efficiency, and well supported by most existing tools (unlike OWL Full). In the following, we recall the basic definitions of OWL DL, only to the extent of what is required in this paper. Here, we adopt the terms of DL rather than those of the OWL standard, because they provide a much more compact syntax, and because our focus is more on theory than on technology.

2.1 Description Logics

The definition of DL is based on three main ideas: concepts, roles and objects. Given a domain of individuals (e.g., persons in genealogy), the concepts are sets of individuals having shared characteristics (e.g. $Women$); the roles are binary relations between individuals (e.g. $sibling$); the objects are particular individuals (e.g. BOB). The syntax gathers these 3 main ideas, plus constructors for building complex concepts and roles, which determine the expressivity of the DL.

Definition 1 (Signature) *The language of formulas of a description logic is characterized by a signature $S = (O, C_a, R_a, Cstr)$, where O is a set of objects, C_a is a set of atomic concepts (denoted c_i), R_a is a set of atomic roles (denoted r_i) and $Cstr$ is a set of constructors used to create complex concepts (denoted C_i) and complex relations (denoted R_i).*

The description logic OWL DL is based on the description logic $\mathcal{SHOIQ}$, whose constructors are listed in Table 1 (the notation $\#S$ represents the cardinality of a set S).

Once the syntax is established, the second notion to define is semantics. The main difference between classical logic and description logics is the interpretation function. Where classical logic returns for each formula a truth value, DL interpretation function returns for each concept a set of individuals, its instances, and for each role a binary relation between individuals. The interpretation function for the constructors of the DL $\mathcal{SHOIQ}$ is given in Table 1.

Definition 2 (Interpretation) *Let $S = (O, C_a, R_a, Cstr)$ be a signature. An interpretation $\mathcal{I} = (\Delta_\mathcal{I}, \cdot^\mathcal{I})$ of S is a set of individuals $\Delta_\mathcal{I}$, called the interpretation domain, and a interpretation function $\cdot^\mathcal{I}$, that associates:*
to each object o_i an individual $o_i^\mathcal{I} \in \Delta_\mathcal{I}$;
to each concept c_i a set of individuals $c_i^\mathcal{I} \subseteq \Delta_\mathcal{I}$;
to each relation r_i a binary relation $r_i^\mathcal{I} \subseteq \Delta_\mathcal{I} \times \Delta_\mathcal{I}$.

The third notion to be defined for a DL is the knowledge base. A knowledge base divides the knowledge in two parts: the *terminological* part, which represents general knowledge, true for all individuals, and the *assertional* part, which represents particular knowledge, only true for particular individuals.

Definition 3 (Knowledge base) *We define a* knowledge base *as a pair $\Sigma = (T, A)$. T is the terminological part (T-Box), represented by a set of axioms like "C_i is subsumed by C_j", denoted $C_i \sqsubseteq C_j$, which means that every instance of C_i is necessarily an instance of C_j. The equivalence of 2 concepts, denoted by $C_i \doteq C_j$, is defined by the 2 axioms $C_i \sqsubseteq C_j$ and $C_j \sqsubseteq C_i$. A is the assertional part (A-Box), represented by a set of assertions like $o : C$ and $(o_i, o_j) : R$, which respectivly means that o belongs to the concept C, and a connection R exists between o_i and o_j. Moreover, the DL $\mathcal{SHOIQ}$ allows additional axioms in the T-Box: $r_i \sqsubseteq r_j$ (called role hierarchy), r_i $inverse\ of$ r_j, r $functional$ and r $transitive$.*

In the following, the notation r^{-1} is used to designate the role that is defined to be the inverse of the role r, when defined. Figure 1 presents an example knowledge base Σ_{ex}, with a T-Box and an A-Box. T-Box has 3 concepts: $Person$, $Team$, and $Bigteam$. A instance of $Team$ is defined as having a leader and at least 2 members. An instance of $Bigteam$ is defined as having a leader and at least 3 members. To help the reading of knowledge bases, concepts are written with an uppercase, relations in lowercase, and objects in uppercase letters. The A-Box of Σ_{ex} presents the description of the team LIS, leaded by OLIVIER and composed of 2 others members, PIERRE and SEBASTIEN.

Given a knowledge base Σ, some interpretations are distinguished as *models* of Σ.

Definition 4 (Model) *Let $\Sigma = (T, A)$ be a knowledge base and $\mathcal{I}$ an interpretation, for the same signature S. We call $\mathcal{I}$ model of Σ, denoted $\mathcal{I} \models \Sigma$, when:*
- for each $C_i \sqsubseteq C_j$ in T, $C_i^\mathcal{I} \subseteq C_j^\mathcal{I}$;
- for each $o : C$ in A, $o^\mathcal{I} \in C^\mathcal{I}$;
- for each $(o_i, o_j) : R$ in A, $(o_i^\mathcal{I}, o_j^\mathcal{I}) \in R^\mathcal{I}$.

¿From this notion of model, statements and their inference from a knowledge base can be defined: classification and instantiation. The *classification* statements establish subsumption relations between concepts, and hence help to organize them. The *instantiation* statements establish the belonging of objects to concepts, and hence help to place objects in the concept classification.

Class	Description	Syntax	Semantics
$\mathcal{S}$	top (most general concept)	$\top$	$\Delta_\mathcal{I}$
	bottom (most specific concept)	$\bot$	$\emptyset$
	conjunction of two concepts	$C_1 \sqcap C_2$	$C_1^\mathcal{I} \cap C_2^\mathcal{I}$
	disjunction of two concepts	$C_1 \sqcup C_2$	$C_1^\mathcal{I} \cup C_2^\mathcal{I}$
	concept negation	$\neg C$	$\Delta_\mathcal{I} \setminus C^\mathcal{I}$
	qualified universal quantification	$\forall r.C$	$\{i \in \Delta_\mathcal{I} \mid \forall j : (i,j) \in r^\mathcal{I} \rightarrow j \in C^\mathcal{I}\}$
	qualified existential quantification	$\exists r.C$	$\{i \in \Delta_\mathcal{I} \mid \exists j : (i,j) \in r^\mathcal{I} \wedge j \in C^\mathcal{I}\}$
$\mathcal{O}$	at least one of objects	$\{o_1 \dots o_n\}$	$\{o_1^\mathcal{I}, \dots, o_n^\mathcal{I}\}$
$\mathcal{Q}$	minimal cardinality	$\geq n\, r.C$	$\{i \in \Delta_\mathcal{I} \mid \#\{j \in C^\mathcal{I} \mid (i,j) \in r^\mathcal{I}\} \geq n\}$
	maximal cardinality	$\leq n\, r.C$	$\{i \in \Delta_\mathcal{I} \mid \#\{j \in C^\mathcal{I} \mid (i,j) \in r^\mathcal{I}\} \leq n\}$
	exact cardinality	$= n\, r.C$	$\{i \in \Delta_\mathcal{I} \mid \#\{j \in C^\mathcal{I} \mid (i,j) \in r^\mathcal{I}\} = n\}$

Table 1. Set of constructors of the description logic $\mathcal{SHOIQ}$, with their syntax and semantics.

$$\begin{aligned}
Person &\sqsubseteq \top \\
Team &\doteq \exists\, hasleader.Person \\
&\ \sqcap \geq 2\ hasmember.Person \\
Bigteam &\doteq \exists\, hasleader.Person \\
&\ \sqcap \geq 3\ hasmember.Person \\
hasleader &\sqsubseteq hasmember
\end{aligned}$$

$$\begin{aligned}
\text{OLIVIER} &: Person \\
\text{SEBASTIEN} &: Person \\
\text{PIERRE} &: Person \\
\text{LIS} &: Team \\
(\text{LIS}, \text{OLIVIER}) &: hasleader \\
(\text{LIS}, \text{SEBASTIEN}) &: hasmember \\
(\text{LIS}, \text{PIERRE}) &: hasmember
\end{aligned}$$

Figure 1. Example of knowledge base

Definition 5 (Classification and Instantiation) *Two kinds of statements can be infered from a knowledge base Σ:*
Classification: $\Sigma \models C_i \sqsubseteq C_j$ iff $\forall \mathcal{I} \models \Sigma : C_i^\mathcal{I} \subseteq C_j^\mathcal{I}$;
Instantiation: $\Sigma \models o :\ C$ iff $\forall \mathcal{I} \models \Sigma : o^\mathcal{I} \in C^\mathcal{I}$.

For instance, it is easy to prove in our example that every big team is also a team: $\Sigma_{ex} \models Bigteam \sqsubseteq Team$. Also, OLIVIER being the leader of LIS, hence a member of it, and SEBASTIEN and PIERRE being members too, we can prove that LIS is a big team, i.e. $\Sigma_{ex} \models$ LIS : $Bigteam$.

2.2 Browsing Ontologies

There exists a number of tools for browsing ontologies. Some of them are simply viewers that make no or little inference. They present an ontology in the form of a tree or a graph. The tree represents the classification of all named concepts, and each concept can also be decorated by its instances (e.g., Protégé). The graph represents a knowledge base by representing concepts and objects by nodes, and subsumption, equivalence, and assertions by edges (e.g., RDF-Graph-Viewer). On one hand, some viewers impose to display the full ontology at once, making it quickly unreadable. On the other hand, some viewers display only predefined subsets of information, e.g. the A-Box or the T-Box (e.g., SVG-OWL-Viewer). In summary, these viewers do not provide navigation facilities that would allow users to focus on various subsets of information.

Other tools provide querying facilities, by adapting the SQL language to web ontologies. RQL [5] is designed to query RDF ontologies; OWL-QL [3] is a language to query OWL ontologies. They heavily rely on inference mechanisms in the course of computing answers to queries. However, like with most querying systems, the user is not guided in his search, and must have preliminary knowledge about the ontology structure, and the query language, in order to express useful queries.

3. Query, Extension, and Dynamic Taxonomy

The objective of this section is to show how a web ontology can fit into the framework of dynamic taxonomies. The key notions are the classification and instantiation statements that can be inferred from a knowledge base. Classification statements define a subsumption ordering over concepts, which can play the role of a taxonomy. Instantiation statements determine whether an object is an instance of a concept, and hence define the deep extension of concepts. Therefore, all ingredients are present to apply the framework of dynamic taxonomies on web ontologies. The important differences with the usual application of DTs are that (1) the concepts may be complex and in infinite number, and (2) the taxonomy shape and extensions are defined through logical inference. The infinite number of concepts is coped with by selecting which concepts should appear in taxonomies, and also by computation-on-demand when expanding concepts. Roles seem absent from this picture, but in fact, they appear as parts of complex concepts and are exploited by new kinds of navigation links (see Section 4).

At each focus, we define the *local view* as the combination of a query, an extension, and a dynamic taxonomy. The *query* specifies the current focus, i.e. the user location in the vast navigation space. The *extension* is the set of answers of the query, i.e. a set of objects. The *dynamic taxonomy* gives feedback about this extension, and it is the support of most navigation links.

Definition 6 (Query) *Let S be a signature. A query is a (complex) concept, which can be written with all OWL DL constructors. In particular, the 3 boolean constructors are available (conjunction, disjunction, negation).*

Sometimes, it is useful to see the query as a conjunctive set of simpler concepts, which can be obtained from any query by putting it in conjunctive normal form. In this paper, we use alternately the 2 forms. For instance, the query $Team \sqcap \neg Person$ is identical to $\{Team, \neg Person\}$.

Next, given a query q, the extension is defined as the set of objects that can be proved to be instances of q. This definition explains why we assume the knowledge base has an A-Box (assertional box). Otherwise, no instantiation statement could be infered, and so all extents would be empty.

Definition 7 (Extension) *Let Σ be a knowledge base and q be a query. The* extension *of q is the set of objects which are instances of this concept: $ext(q) = \{o \in O \mid \Sigma \models o : q\}$.*

We have to define the taxonomy before being able to associate it to a query. The taxonomy is made of a subset of the concept language, ordered by subsumption.

Definition 8 (Taxonomy) *Let Σ be a knowledge base. The* taxonomy *derived from Σ is the partially ordered set $T_\Sigma = (X_\Sigma, \sqsubseteq_\Sigma)$. The set of concepts X_Σ is the smallest set that contains the concept $\top$, all concept names in C_a, and for every role name $r \in R_a$, every concept name $c \in C_a$, and every natural number $n \in \mathbb{N}$, the concepts $\exists r.\top$, $\exists r.c$, $\geq n\ r.\top$ and $\geq n\ r.c$. Any 2 concepts $C, D \in X_\Sigma$ are ordered by $\sqsubseteq_\Sigma$ (i.e. $C \sqsubseteq_\Sigma D$) iff $\Sigma \models C \sqsubseteq D$.*

The 3 boolean constructors ($\sqcap$, $\sqcup$, and $\neg$) are excluded from the taxonomy because they are easily introduced into queries through the navigation process (see Section 4). The "at least one of objects" can also be introduced through navigation by a direct selection of objects in the extension. Other constructors are excluded from the taxonomy because they are redundant according to equivalences (e.g. $\forall r.C \doteq \neg\exists r.\neg C$).

Now, given a query q, this taxonomy is pruned to retain only concepts that are extensionally related to the query, which results in the *dynamic taxonomy*.

Definition 9 (Dynamic taxonomy) *Let Σ be a knowledge base, and q be a query. A concept $x \in T_\Sigma$ is extensionally related to the query q iff $ext(x) \cap ext(q) \neq \emptyset$. The* dynamic taxonomy *$DT_\Sigma(q)$ is the pruning of T_Σ to the concepts that are extensionally related to the query q. For historical reasons [1], we name those concepts* increments.

Because of concepts $\geq n\ r.C$, there is an infinite number of concepts in the taxonomy T_Σ. However, every dynamic taxonomy is finite because a knowledge base is finite, and therefore for every role r and every concept C, there is necessarily a number k such that for every $n \geq k$, the extension of $\geq n\ r.C$ is empty. Still, dynamic taxonomies are often too large to be computed and displayed entirely at once. This is why users are initially presented with a fully collapsed dynamic taxonomy, showing only the most general concept $\top$, and are allowed to expand concepts on demand, i.e. computing and displaying children increments.

4. Navigation Links

A focus is a particular point of view over a knowledge base, and we need navigation links to move from one focus to another. Every navigation link is anchored on some element of the local view, and defined as a function that maps the current query to a new query.

4.1 Zoom-in and Zoom-out

The most common navigation link is the zoom-in link. A zoom-in link, aka specialization, reaches a query, whose extension is smaller than the previous extension, but not empty. A concept from the dynamic taxonomy $DT_\Sigma(q)$, an increment of q, can be used as a zoom-in link iff $ext(x) \cap ext(q) \subsetneq ext(q)$, in order to ensure that a strictly smaller extension is reached. Let x be such an increment, the new query is obtained by replacing by x the elements in q that subsume x (specialization): $q \leftarrow (q \setminus \{C \in q \mid x \sqsubseteq C\}) \cup \{x\}$.

A zoom-out link, aka generalization, is the inverse of the zoom-in. A increment x can be used as a zoom-out link iff it subsumes some query element: $\exists C \in q : C \sqsubseteq x$. There are 2 cases when following a zoom-out link, respectively the removal and the generalization of query elements. In the first case, when $x \in q$, the concept x is simply removed from the query: $q \leftarrow q \setminus \{x\}$. In the second case, when $x \notin q$, the query elements that are strictly subsumed by x are replaced by x (generalization): $q \leftarrow (q \setminus \{C \in q \mid C \sqsubseteq x\}) \cup \{x\}$. Hence, every zoom-in link can be undone by a zoom-out link, and conversely.

To increase the expressivity of navigation links, we allow the combination of increments into complex increments. First, when several increments $\{x_i\}_{i \leq n}$ are selected in the dynamic taxonomy, they are disjuncted to form a complex increment $x = x_1 \sqcup \ldots \sqcup x_n$. Second, a negated zoom-in is provided by replacing x by $\neg x$ in the above definition. Finally, the OWL DL logic also provides the constructor $\{o_1 \ldots o_n\}$. These concepts can be used in zoom links exactly like other increments by picking objects in the extension. In this way, the extension can be restricted to a subset of the current extension or a set of objects can be excluded from the current extension.

4.2 Reversal and Traversal

We now introduce and motivate new navigation links that make use of roles. First, roles are already present in dynamic taxonomies through quantified concepts (e.g., $\exists r.C$ and $\geq n\, r.C$). So, they can be introduced in queries by zoom-in. For instance, the following query can be reached in 2 zoom-in steps: $q = \{Bigteam, \exists hasleader.Person\}$, i.e. "the big teams with a leader". A useful navigation link would be to reach the related query: $\{Person, \exists isleaderof.Bigteam\}$, i.e. "the leaders of big teams", where $isleaderof$ is defined as the inverse of $hasleader$. This navigation link, called *reversal* link, changes the point of view by crossing a role in the query, and turning upside-down the query accordingly. In the above example, the point of view has been changed from teams to persons, through the role $hasleader$. A query element x can be used as a reversal link if it has the form $x = \exists r.C$, and then the new query is defined as: $q \leftarrow C \cup \{\exists\, r^{-1}.(q \setminus \{\exists\, r.C\})\}$. It can be verified that, if we follow again the reversal link on $\exists r^{-1}.(q \setminus \{\exists\, r.C\})$, we come back to the initial query q.

The second type of navigation link using roles is a composition of two navigation links defined above and is called *traversal*. Indeed, it often happens that the user wants to cross a role that is not yet present in the query, but already visible in the dynamic taxonomy. Hence, for an increment $x = \exists r.C$, x is successively used for zoom-in and reversal. These two steps can be simplified by the following definition of the traversal link: $q \leftarrow C \cup \{\exists\, r^{-1}.q\}$. For instance, we can move in one step from the query $Bigteam$ to the query $\{Person, \exists isleaderof.Bigteam\}$ by traversing the increment $\exists hasleader.Person$.

In the definitions of reversal and traversal, we assume that the role r has an inverse that is defined in the knowledge base. If a role has no inverse, we can either prevent it to be reversed or traversed, or we can automatically define an inverse role in the knowledge base.

5 Conclusion

We developed a prototype, ODALISQUE (Ontology Description and Logical Information System Querying) to experiment DTs-like browsing of OWL ontologies. It shares a similar user interface with an existing LIS implementation [2], but is based on a DL reasoner (with help of Jena API[1]) to infer classification and instantiation statements. It also adds the navigation links related to roles. An example of a complete navigation scenario of an OWL ontology about out research laboratory is available[2].

In this paper, we have demonstrated how dynamic taxonomies can be adapted and applied for browsing web ontologies, using the OWL DL language, which is the most expressive among decidable ontology languages. ODALISQUE, a fully OWL standard compliant prototype which only assumes a non-empty A-Box, was developed. It relies on existing OWL reasoners, hence ensuring the correctness of its answers. It includes all features of dynamic taxonomies, and introduces new possibilities. First, complex concepts can be used not only in queries, but also in dynamic taxonomies, and classical navigation modes can be applied to them. Second, relations can be defined between objects through role assertions, and this entails new navigation links for reversing a query (change of point of view), and traversing a role from a set of objects to a set of related objects. Third, because of the increased diversity of navigation links, it appears that navigation links can be defined not only on increments, i.e. concepts in dynamic taxonomies, but also on objects in extensions, and on some query parts.

References

[1] S. Ferré and O. Ridoux. An introduction to logical information systems. *Information Processing & Management*, 40(3):383–419, 2004.

[2] S. Ferré and O. Ridoux. Logical information systems: from taxonomies to logics. In *Int. Work. Dynamic Taxonomies and Faceted Search (FIND)*, pages 212–216. IEEE Computer Society, 2007.

[3] R. Fikes, P. Hayes, and I. Horrocks. OWL-QL: A language for deductive query answering on the semantic web. Technical Report KSL 03-14, Stanford University, Stanford, CA, 2003.

[4] I. Horrocks and U. Sattler. Ontology reasoning in the SHOQ(D) description logic. In B. Nebel, editor, *Int. Joint Conf. Artificial Intelligence*. Morgan Kaufmann, 2001.

[5] G. Karvounarakis, A. Magkanaraki, S. Alexaki, V. Christophides, D. Plexousakis, M. Scholl, and K. Tolle. Querying the semantic web with RQL. *Computer Networks*, 42(5):617–640, 2003.

[6] G. Sacco. Dynamic taxonomies: a model for large information bases. *Knowledge and Data Engineering, IEEE Transactions on*, 12(3):468–479, 2000.

[7] G. M. Sacco. The intelligent e-sales clerk: the basic ideas. In M. R. et al, editor, *Int. Conf. Human-Computer Interaction (INTERACT)*, pages 876–879. IOS Press, 2003.

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

[2]http://hal.archives-ouvertes.fr/hal-00284992/en/