Graphol: Ontology Representation Through Diagrams

Marco Console, Domenico Lembo, Valerio Santarelli, and Domenico Fabio Savo

Dipartimento di Ing. Informatica, Automatica e Gestionale "Antonio Ruberti"
SAPIENZA Università di Roma
Via Ariosto 25, I-00186 Roma, Italy
{console,lembo,santarelli,savo}@dis.uniroma1.it

Abstract. In this paper we present Graphol, a novel language for the diagrammatic representation of Description Logic (DL) ontologies. Graphol is designed with the aim of offering a completely visual representation to the users (notably, no formulas need to be used in the diagrams), thus helping the understanding of people not skilled in logic. At the same time, it provides designers with simple mechanisms for ontology editing, which free them from having to write down complex textual syntax. Through Graphol we can specify $\mathcal{SROIQ}(D)$ ontologies, thus our language essentially captures the OWL 2 standard. In this respect, we developed a basic software tool to translate Graphol ontologies realized with the yEd graph editor into OWL 2 functional syntax specifications. We conducted some initial user evaluation tests, involving designers skilled in conceptual or ontology modeling and users without specific logic background. From these tests, we obtained promising results about the effectiveness of our language for both visualization and editing of ontologies.

1 Introduction

Ontologies are widely recognized as the best means to share domain knowledge in a formal way. This implies that the representation they provide has to be agreed upon by all their users, so that ontologies can act as reference models across groups of people, communities, institutions, and applications. The relevance of this problem is also amplified by the increased uptake of ontologies in several contexts, such as biomedicine, life sciences, e-commerce, cultural heritage, or enterprize applications [25]. Obviously, it is very likely that people operating in such contexts are not experts in logic and generally do not possess the necessary skills to interpret formulas through which ontologies are typically expressed. This turns out to be a serious problem also (and probably mainly) in the development of an ontology. Indeed, ontologists usually work together with domain experts, the first providing their knowledge about ontology modeling and languages, the latter providing their expertise on the domain of interest. During this phase, communication between these actors is fundamental for the ontology to eventually become a formal specification which faithfully represents the domain requirements.

We also notice that, despite the recent popularity of ontologies, there is still a serious lack of designers with the right skills for their modeling. In this respect, analysts and experts in conceptual modeling for software and database design might be good candidates to compensate for this lack, provided that they are trained to acquire the needed knowhow. However, they need the right tools to approach the problem, possibly close in spirit to those they usually adopt for conceptual modeling.

In the past years, some efforts have been made in this direction, and graphical syntaxes for ontologies based on the use of standard conceptual modeling languages have been devised. In particular, UML profiles for OWL DL have been proposed in [8,16,24], with the aim of making off-the-shelf UML tools usable for the purpose of ontology development. These profiles have been devised for OWL 1 [7], but did not evolve thereafter towards OWL 2 [5], and extending them to all new features of the current standard is not straightforward. An UML-based language for graphical editing of OWL 2 ontologies is instead at the basis of the more recent OWLGrEd tool [6,9]. In OWLGrEd, however, many complex OWL expressions are specified through formulas in Manchester syntax [19]. This somehow affects the graphical nature of the representation, especially for complex ontologies, where the proliferation of such formulas can compromise intuitive understanding of the final ontology. The use of (variants of) UML for ontology representation is also proposed in [17,18]. The first work, however, is focused on foundational conceptual modeling languages, rather than on DL ontologies. Instead, the tool described in [17] supports standard UML class diagrams (and also Entity-Relationship ones), where complex definitions of ontology classes and roles must be expressed through, e.g., OWL views. Thus, it does not provide extensions of standard conceptual modeling languages to capture OWL. Another graphical notation for OWL, not based on UML, is that adopted by the GrOWL ontology editor [22], which models ontologies as labeled graphs. Notably, the representation in GrOWL is completely graphical, even though it relies on a quite large set of symbols and makes use of various kinds of labeled edges. Also in this case the proposal is tailored to OWL 1, and it seems that the development of the tool has been discontinued to date. We further notice that other several formalisms for the graphical representation of knowledge have been proposed during the years (see, e.g., [10,14,1]). Such approaches are either not tailored to DLs or limited in their expressive power, i.e., they do not capture the OWL 2 standard, or large fragments thereof. We finally point out that, besides the attempts to provide languages for the graphical specification of ontologies, much attention has been dedicated to the issue of ontology visualization and various tools for it have been proposed, such as the popular Protègè plugins OntoGraf[1] and OWLViz[2]. The aim of such solutions is to ease navigation of the ontology, and thus typically they allow the user to obtain various possible views of it, according to some graphical format. Such formats are therefore not thought also for the editing task (see [21] for a survey on this matter).

In this paper we present our proposal for the graphical representation of DL ontologies and introduce the novel *Graphol* language. The main characteristics of Graphol can be summarized as follows:

- Similarly to previous UML-based approaches, it is rooted in a standard language for conceptual modeling. Indeed the basic components of Graphol are taken from the Entity-Relationship (ER) model. Notably, simple ontologies that correspond to classical ER diagrams (e.g., some ontologies specified in OWL 2 QL [23]) have in Graphol a representation that is isomorphic to the ER one;
- Similarly to OWLGrEd, it is tailored to OWL 2, and thus it overcomes the limits of previous UML profiles for OWL 1 and of other graphical notations for ontologies

[1] http://protegewiki.stanford.edu/wiki/OntoGraf
[2] http://protegewiki.stanford.edu/wiki/OWLViz

that capture less expressive DLs. More precisely, Graphol subsumes $\mathcal{SROIQ}(D)$, the logical underpinning of OWL 2 [13,20]. Differently from OWLGrEd, however, it offers a completely visual representation to the users (notably, no formulas need to be used in Graphol diagrams);
- As in GrOWL, which as said is limited to OWL 1, a Graphol ontology is a graph, but to draw it we resort to some standard symbols for denoting roles, concepts, and attributes (those of the ER model), and do not use labeled edges. More precisely, graph nodes represent either predicates from the ontology alphabet or constructors used to build complex expressions from named predicates. Then, two kinds of edges are adopted: Input edges, used to specify arguments of constructors, and inclusion edges, used to denote inclusion axioms between (complex) expressions.
- Graphol has a precise syntax. The structure of its graphical assertions traces that of typical DL axioms, so that our language has a natural encoding in DL. Through such encoding, we assign our language with a clear and formal semantics. Syntax and semantics of Graphol are presented in Section 2.

To support the graphical modeling of ontologies and at the same time foster the interoperation with standard OWL reasoners and development environments, we have devised a process that leads the designer from the drawing of a Graphol diagram, through the use of the open source graph editor yEd[3], to the production of the corresponding OWL 2 specification. We discuss these aspects in Section 3.

Finally, to verify the effectiveness of our language, we have conducted some user evaluation tests, where both designers skilled in conceptual or ontology modeling and users without specific logic background were involved. Our tests, though preliminary, showed that Graphol has been perceived as a valuable and easy to use tool for both ontology editing and visualization, and confirmed the importance for a visual ontology language to be completely graphical, which is one of the distinguishing feature of Graphol. A synthesis of our tests is given in Section 4, whereas for a complete description of our evaluation study, we refer the reader to the full version of the present paper [12].

2 The Graphol language

Graphol syntax. The terms that make up the alphabet of the ontology, meaning the named concepts, roles, attributes, value-domains (i.e., predetermined datatypes), and constants (both individuals and values) are modeled by labeled rectangles, diamonds, circles, rounded corner rectangles, and octagons, respectively. These symbols are called *predicate nodes*, and the associated labels refer to predicates in the ontology alphabet.

We reserve two special labels, "Top" and "Bottom", to represent, respectively, the universal concept, value-domain, role, or attribute, and the empty concept, value-domain, role, or attribute. For instance, a rectangle labeled "Top" represents the universal concept ($\top_C$), while a rectangle labeled "Bottom" represents the empty concept ($\bot_C$). Similarly for roles, attributes, and value-domains.

Other types of nodes, called *constructor nodes*, correspond to logical operators such as intersection, union, negation (i.e., complement of), or restrictions on the domain and

[3] http://www.yworks.com/en/products_yed_about.html

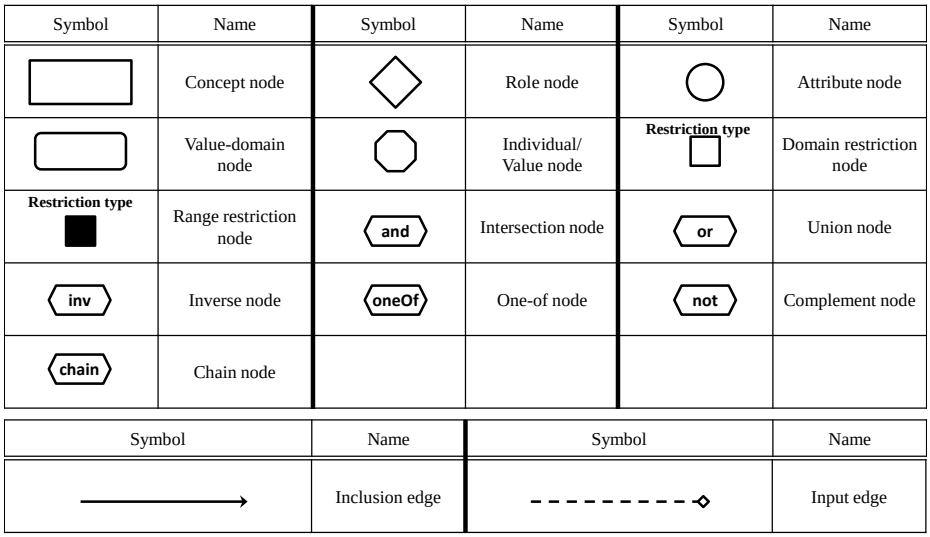


Fig. 1: Graphol predicate and constructor nodes.

range of roles and attributes, and are used to graphically construct concept, role, attribute, or value-domain expressions. Graphol nodes are shown in the top part of Figure 1, where the "Restriction type" label of the domain and range restriction nodes can be equal to: "exists", "forall", "$(x, -)$", or "$(-, x)$", where x is a positive integer.

In Graphol, an edge can be of two types: *input edge* or *inclusion edge*. The former is a dashed directed diamond edge, where the diamond indicates the target end, and is used to construct complex expressions in the ontology by linking constructor and predicate nodes. The latter is a solid directed arrow edge, where the arrow indicates the target end, and is used to represent an inclusion assertion between expressions. The *input edge* and the *inclusion edge* are depicted in the bottom part of Figure 1.

A *Graphol expression* is a weakly-connected acyclic directed graph, whose nodes are both predicate and constructor nodes and whose edges are only input edges, that has exactly one node with no outgoing edges. We call such node the *sink* of the expression. We recall that a directed graph is called weakly-connected if replacing all of its directed edges with undirected edges produces a connected (undirected) graph. The definition of expressions in our language is formalized below. In such definition, when we mention constructor nodes with input expressions we mean a (portion of a) graph where an input edge goes from the sink of the expression to the constructor node.

Definition 1. *A Graphol expression is built through one of the following specifications.*

1. *A concept expression is a weakly-connected acyclic graph, defined as follows.*
   - *a concept node is a concept expression, whose sink node is the concept itself;*
   - *a domain or range restriction node, labeled "exists", "forall", "$(x, -)$", or "$(-, x)$", with one input role expression and one input concept expression, is a concept expression, whose sink node is the domain or range restriction node;*

- *a domain restriction node, labeled "exists", "forall", "$(x, -)$", or "$(-, x)$", with one input attribute expression and one input value-domain expression, is a concept expression, whose sink node is the domain restriction node;*
- *an intersection or union node with $n \geq 2$ input concept expressions is a concept expression, whose sink node is the intersection node or the union node;*
- *a complement node with one input concept expression is a concept expression, whose sink node is the complement node;*
- *a one-of node with $n \geq 1$ input individual nodes is a concept expression, whose sink node is the one-of node.*

2. *A role expression is a weakly-connected acyclic graph, defined as follows.*
   - *a role node is a role expression, whose sink node is the role itself;*
   - *an intersection or union node with $n \geq 2$ input role expressions is a role expression, whose sink node is the intersection or the union node;*
   - *an inverse node with one input role expression is a role expression, whose sink node is the inverse node;*
   - *a complement node with one input role expression is a role expression, whose sink node is the complement node;*
   - *a chain node with $n \geq 2$ input role expressions, each one labeled with i such that $1 \leq i \leq n$ and $i \neq j$ for any two labels i and j, is a role expression, whose sink node is the chain node.*
3. *An attribute expression is a weakly-connected acyclic graph, defined as follows.*
   - *an attribute node is an attribute expression, whose sink node is the attribute itself;*
   - *an intersection or union node with $n \geq 2$ input attribute expressions is an attribute expression, whose sink node is the intersection or the union node;*
   - *a complement node with one input attribute expression is an attribute expression, whose sink node is the complement node.*
4. *A value-domain expression is a weakly-connected acyclic graph, defined as follows.*
   - *a value-domain node is a value-domain expression, whose sink node is the value-domain node itself;*
   - *a range restriction node, labeled "exists", with one input attribute expression is a value-domain expression, whose sink node is the range restriction node;*
   - *an intersection or union node with $n \geq 2$ input value-domain expressions is a value-domain expression, whose sink node is the intersection or the union node;*
   - *a complement node with one input value-domain expression is a value-domain expression, whose sink node is the complement node;*
   - *a one-of node with $n \geq 1$ input value nodes is a value-domain expression, whose sink node is the one-of node.*

As usual in ontology languages and DLs, intentional axioms are specified through inclusions. In Graphol, an *inclusion assertion* is expressed through an inclusion edge going from the (sink node of the) subsumed expression to the (sink node of the) subsumer. In detail, a concept (resp., role, attribute, value-domain) inclusion assertion is obtained by linking two sink nodes of concept (resp., role, attribute, value-domain) expressions. Inclusion edges between expressions of different types (e.g., a concept expression with a role expression) are not allowed. Finally, we define a Graphol ontology as a set of Graphol assertions.

Graphol semantics. We start with Graphol expressions and give their semantics by providing their representation in terms of DL expressions, which in turn have a formal semantics [4]. To this aim, we define a function Φ that takes as input a Graphol expression E_G and returns a DL expression that represents it.

In formalizing Φ, we denote with $\mathsf{sink}(E_G)$ the sink node of E_G, and with $\mathsf{arg}(E_G)$ the possibly empty set of Graphol expressions that are linked to $\mathsf{sink}(E_G)$ by means of input edges. Then, we define Φ as follows:

- if $\mathsf{sink}(E_G)$ is a concept, role, attribute, value-domain, or individual/value node labeled S, then $\Phi(E_G) = S$[4];
- if $\mathsf{sink}(E_G)$ is a domain restriction node labeled "exists" (resp., "forall", "$(x, -)$", "$(-, x)$"), and $\mathsf{arg}(E_G) = \{\epsilon_{RA}, \epsilon_{CV}\}$, then $\Phi(E_G) = \exists\Phi(\epsilon_{RA}).\Phi(\epsilon_{CV})$ (resp., $\Phi(E_G) = \forall\Phi(\epsilon_{RA}).\Phi(\epsilon_{CV})$, $\Phi(E_G) =\geq x\ \Phi(\epsilon_{RA}).\Phi(\epsilon_{CV})$, $\Phi(E_G) =\leq y\ \Phi(\epsilon_{RA}).\Phi(\epsilon_{CV})$);
- if $\mathsf{sink}(E_G)$ is a range restriction node labeled "exists" (resp. "forall", "$(x, -)$", "$(-, x)$"), and $\mathsf{arg}(E_G) = \{\epsilon_R, \epsilon_C\}$, then $\Phi(E_G) = \exists(\Phi(\epsilon_R))^-.\Phi(\epsilon_C)$ (resp. $\Phi(E_G) = \forall(\Phi(\epsilon_R))^-.\Phi(\epsilon_C)$, $\Phi(E_G) =\geq x\ (\Phi(\epsilon_R))^-.\Phi(\epsilon_C)$, $\Phi(E_G) =\leq y\ (\Phi(\epsilon_R))^-.\Phi(\epsilon_C)$);
- if $\mathsf{sink}(E_G)$ is a range restriction node labeled "exists" and $\mathsf{arg}(E_G) = \{\epsilon_A\}$, then $\Phi(E_G) = \exists(\Phi(\epsilon_A))^-$;
- if $\mathsf{sink}(E_G)$ is a complement node and $\mathsf{arg}(E_G) = \{\epsilon\}$, then $\Phi(E_G) = \neg\Phi(\epsilon)$;
- if $\mathsf{sink}(E_G)$ is an intersection (resp. a union or a one-of node) and $\mathsf{arg}(E_G) = \bigcup_{i=1}^{n} \epsilon^i$, then $\Phi(E_G) = \bigsqcap_{i=1}^{n} \Phi(\epsilon^i)$ (resp. $\Phi(E_G) = \bigsqcup_{i=1}^{n} \Phi(\epsilon^i)$, $\Phi(E_G) = \{\Phi(\epsilon^1), \ldots, \Phi(\epsilon^n)\}$);
- if $\mathsf{sink}(E_G)$ is an inverse node and $\mathsf{arg}(E_G) = \{\epsilon_R\}$, then $\Phi(E_G) = (\Phi(\epsilon_R))^-$;
- if $\mathsf{sink}(E_G)$ is a chain node and $\mathsf{arg}(E_G) = \{\epsilon_R^1, \ldots \epsilon_R^n\}$, where each ϵ_R^i, with $1 \geq i \geq n$, is a Graphol role expression linked to $\mathsf{sink}(E_G)$ by an input edge labeled i, then $\Phi(E_G) = \Phi(\epsilon_R^1) \circ \Phi(\epsilon_R^2) \circ \cdots \circ \Phi(\epsilon_R^n)$.

Similarly, in order to provide the semantics of Graphol inclusion assertions, we define a function Θ that takes as input one such inclusion A_G and returns its DL representation. We denote with $\mathsf{source}(A_G)$ the Graphol expression whose sink node is the source of the inclusion edge in A_G, and with $\mathsf{target}(A_G)$ the Graphol expression whose sink node is the target of the inclusion edge in A_G. The function Θ is thus defined as $\Theta(A_G) = \Phi(\mathsf{source}(A_G)) \sqsubseteq \Phi(\mathsf{target}(A_G))$. Given a Graphol ontology O_G, we are able to translate it into a DL ontology O_{DL} by applying Θ to each assertion in O_G. The semantics of O_G coincides with the semantics of O_{DL}, for which we refer to [4].

In Tables 1 and 2 we provide an example of the application of the function Φ to Graphol expressions of "depth" 0 or 1, i.e., expressions formed only by predicate nodes or by a constructor node taking as input only predicate nodes.

It is easy to see that any DL ontology produced by the function Θ subsumes a $\mathcal{SROIQ}(D)$ ontology. Furthermore, the inverse of the functions Φ and Θ can be defined analogously, thus showing that $\mathcal{SROIQ}(D)$ is indeed subsumed by Graphol.

[4] We assume the alphabets for concepts, roles, attributes, value-domains, individuals, and values to be pairwise disjoint, so that it is clear which kind of predicate S represents. Also, with a little abuse of notation, if $S =$"Top" (resp. $S =$"Bottom"), we assume the Φ returns the correct DL universal (resp. empty) predicate, depending on whether $\mathsf{sink}(E_G)$ is a concept, role, attribute, or value-domain.

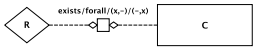
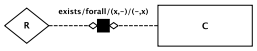
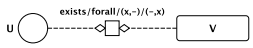
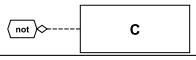
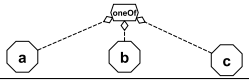
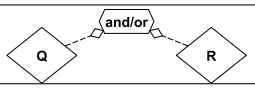
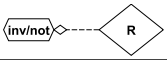
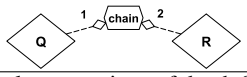
Expression	DL-syntax	Graphol syntax
Atomic Concept	A	
Domain restriction on role	$\exists R.C \;\; \forall R.C$ $\geq xR.C \;\; \leq xR.C$	
Range restriction on role	$\exists R^-.C \;\; \forall R^-.C$ $\geq xR^-.C \;\; \leq xR^-.C$	
Domain restriction on attribute	$\exists U.V \;\; \forall U.V$ $\geq xU.V \;\; \leq xU.V$	
Concept Intersection Concept Union	$C \sqcap D$ $C \sqcup D$	
Concept Complement	$\neg C$	
Concept One-of	$\{a, b, c\}$	
Atomic Role	R	
Role Intersection Role Union	$Q \sqcap R$ $Q \sqcup R$	
Role Inverse Role Complement	R^- $\neg R$	
Role Chain	$Q \circ R$	

Table 1: DL-Graphol correspondence for concept and role expressions of depth 0 or 1.

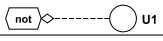
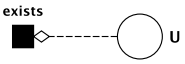
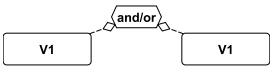
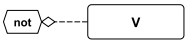
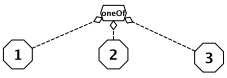
Expression	DL-syntax	Graphol syntax
Attribute	U	
Attribute Intersection Attribute Union	$U_1 \sqcap U_2$ $U_1 \sqcup U_2$	
Attribute Complement	$\neg U_1$	
Value-domain	V	
Range existential restriction on attribute	$\exists U^-$	
Value-domain Intersection Value-domain Union	$V_1 \sqcap V_2$ $V_1 \sqcup V_2$	
Value-domain Complement	$\neg V$	
Value-domain One-of	$\{1, 2, 3\}$	

Table 2: DL-Graphol correspondence for attribute and value-domain expressions of depth 0 or 1.

Shortcuts. In order to aid the user in the design of a Graphol ontology, we have defined some shortcuts that allow for a more compact representation of frequently used expressions and assertions. Here we introduce two of them.

The first shortcut is the *disjoint union node*, which is shaped as a black hexagon. This special node is used to represent a union expression, and at the same time it imposes the disjointness between its arguments. Therefore, it also represents negative inclusion assertions involving its arguments. An example of its use is given in Figure 2.

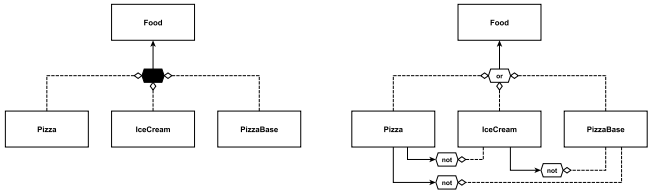


Fig. 2: Example of a disjoint concept hierarchy represented in Graphol with (left-hand side figure) and without (right-hand side figure) the disjoint union node.

We also define a new edge, called *global functionality edge*. which is an input edge with an additional vertical line on its source end. It is used to define the functionality of a role (resp. an inverse role), by linking it to a domain (resp. range) "exists" restriction node. Similarly for the functionality of an attribute. An example is given in Figure 3.

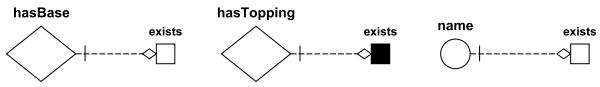


Fig. 3: Example of Graphol global functionality edge: from left to right we specify role functionality, inverse role functionality, and attribute functionality.

Example. In Figure 4, we provide the Graphol specification of a portion of the Pizza ontology. We also present below a logically equivalent representation of such ontology given in terms of DL assertions.

$$
\begin{array}{lcl@{\qquad}lcl}
CheeseTopping & \sqsubseteq & PizzaTopping & Food & \sqsubseteq & \exists calories \\
CheeseTopping & \sqsubseteq & \neg MeatTopping & Pizza & \sqsubseteq & Food \\
MeatTopping & \sqsubseteq & PizzaTopping & Pizza & \sqsubseteq & \neg PizzaTopping \\
MeatTopping & \sqsubseteq & \neg CheeseTopping & CheeseyPizza & \sqsubseteq & Pizza \\
PizzaTopping & \sqsubseteq & Food & MeatyPizza & \sqsubseteq & Pizza \\
PizzaTopping & \sqsubseteq & \neg Pizza & MeatyPizza & \sqsubseteq & \neg VegetarianPizza \\
VegetarianPizza & \sqsubseteq & Pizza & \exists hasIngredient & \sqsubseteq & Food \\
\exists hasIngredient^{-} & \sqsubseteq & Food & hasTopping & \sqsubseteq & hasIngredient \\
\exists hasTopping & \sqsubseteq & Food & \exists hasTopping & \sqsubseteq & PizzaTopping \\
\exists calories & \sqsubseteq & Food & \exists calories^{-} & \sqsubseteq & xsd : integer \\
VegetarianPizza & \equiv & \neg hasTopping.MeatTopping & & &
\end{array}
$$

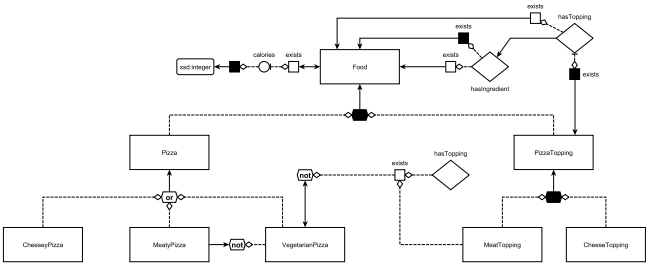


Fig. 4: Graphol ontology example: an excerpt of the Pizza ontology.

3 Tools supporting Graphol

As a must-have for Graphol, we singled out the importance of having an automated translation tool to produce OWL 2 specifications from Graphol ones that are compatible with this standard. We also tried to provide this new language with an easy way to design, edit, and debug the graphical specifications. To these aims, we devised a process for ontology designers that relies on both existing open source tools and originally developed software components.

The process starts specifying, by means of the yEd editor, a Graphol diagram, encoded in the graphml format[5]. The use of a third party software provides the developers with a large set of features, professionally devised for supporting the editing of diagrams of all sorts. To further support editors, we packed up a yEd palette containing all and only the symbols needed for editing a Graphol ontology. This comes in handy in trying to overcome the lack of specificity of the editor.

We then implemented a tool capable of translating a set of yEd diagrams specifying a Graphol ontology into an equivalent OWL 2 encoding, if any. More precisely, the tool is devised to be both a support for the editing tasks and an automatic translator that produces an OWL 2 functional syntax specification. The latter is achieved by parsing the yEd files and translating each portion with the corresponding OWL statement, straightforwardly applying the rules sketched in Section 2. As for the editing aids, the tool provides a syntactic validation of a given diagram. While parsing, if a portion of the graph is found such that either it is non well-formed, i.e., it does not respect the Graphol syntax, or does not correspond to any OWL 2 statement, the process terminates. If the translation was not successful, the tool shows, by means of an external yEd viewer, one of the nodes in the identified untranslatable portion and describes the recognized error in a pop-up window. Both the translator tool and the yEd palette for Graphol can be downloaded from the web site `http://www.dis.uniroma1.it/~graphol/`.

4 User Evaluation

In this section we briefly discuss the user evaluation tests we have conducted for the Graphol language. The goal of these tests is to evaluate the effectiveness of Graphol

[5] `http://graphml.graphdrawing.org/index.html`

for ontology comprehension and design by users with different backgrounds and varying levels of expertise. To achieve this goal, we have conducted two separate studies, designed around a series of model comprehension and model editing tasks which were performed individually by the participants.

- The first is a comparative study in which participants were asked to perform two series of eight comprehension tasks on two ontologies modeled respectively in Graphol and in a different graphical ontology language. For this study we chose a group of ten computer science Ph.D. and Master's students with only basic skills in conceptual modeling, and whose experience with ontologies (if any) is limited to working with ontologies defined by others. We refer to this group as *ontology consumers*.
- The second is a non-comparative user study in which participants were asked to answer ten questions and to perform ten editing tasks on two Graphol ontologies. For this study we selected a group of twelve computer science Ph.D. students, postdocs, and researchers with advanced knowledge in conceptual modeling and some basic skills in logic (but only a small subset of them with some experience in ontology design). We refer to this group as *ontology designers*.

We point out that the number of participants to both our evaluation study is in line with guidelines on usability testing [15].

The language we chose for the comparative study was OWLGrEd [6,9]. OWLGrEd was chosen because its UML-based design principles and visual representation are quite different from Graphol, but are, at least in principle, easily accessible to users who are familiar with UML class diagrams. We point out that this requirement was fulfilled by all participants to this test, though at different expertise levels, and that such participants only had limited knowledge on ontology languages (or none).

We avoided extending the second study with comparative test between Graphol and OWLGrEd for ontology editing purposes because this test would have been strongly influenced by the tools used for the task, and an evaluation of these tools was outside the scope of the experiments (in particular because OWLGrEd comes with its own editor while Graphol currently does not). Furthermore, on a more practical note, this would have excessively drawn out the duration of the test (as it is, it exceeded two hours). We also did not conduct a comparative evaluation of Graphol against ontology editors such as Protègè, or simply against modeling through DL axioms, because these approaches obviously do not offer graphical editing solutions, whereas our main aim has been to test the evaluation of consumers and designers not necessary skilled in ontologies.

In Figure 5 we provide some of the results we obtained through our tests. From left to right, the figure shows the average results in terms of correctness (percentage of correct answers) for the comprehension tasks in Graphol and OWLGrEd conducted by ontology consumers, the average correctness results for the comprehension and editing tasks in Graphol conducted by ontology designers, and the overall average times for the Graphol comprehension and editing tasks (in minutes, where 2.5 minutes for comprehension questions and 4 minutes for editing questions were the predetermined benchmark values). Clearly, the very high average correctness scores and low average times produced by the ontology designers in both tasks show an excellent comprehension of the Graphol language by these users, and a good ability in performing the required

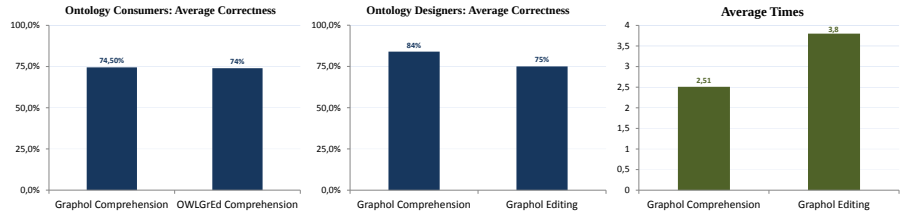


Fig. 5: Average results for percentage of correct answers and for overall times.

Graphol ontology modeling tasks. Furthermore, the ontology consumers, who did not have advanced knowledge on conceptual modeling, nevertheless showed a good level of comprehension of the Graphol language, comparable to, and in some cases better than, the one showed for OWLGrEd, which is strongly based on a formalism which they were familiar with. According to our results and to the feedback we obtained by users from several open questions that were presented in a post-test questionnaire and from a brief discussion that concluded the tests, we also observed that Graphol has been perceived as easier than OWLGrEd for more complex expressions, which in OWLGrED require the use of Manchester syntax formulas.

5 Conclusion

Besides the user evaluation tests we have described in Section 4, we could assess the effectiveness of Graphol also through its use in some industrial projects in which we have been recently involved (see, e.g., [2,3]). In these projects, we developed quite large ontologies, comprising hundreds of predicates and axioms, and have been in touch with domain experts without any background on ontologies or formal languages. This has also been a valuable test bed for the design process we have described in Section 3, which allowed us to automatically obtain processable OWL encodings from our Graphol specification, which instead turned out to be crucial for communication with domain experts.

Both user evaluation tests and practical experience however have highlighted the need of a dedicated editing tool specifically built to support the Graphol ontology specification. The development of such tool is our main future work on Graphol. At the same time, we are working to improve Graphol visualization. A first step on this direction has been the use of a specific component for visualizing Graphol ontologies in Mastro Studio [11], a tool for the management of ontology-based data access applications. In this tool, Graphol ontologies are suitable connected to wiki-like ontology documentation, through hyperlinks from the elements of the diagrams to wiki pages associated to such elements. For the future, we plan to enrich such features, e.g., by enabling construction of on the fly Graphol excerpts of the ontology, based on specific user requests.

Acknowledgments. This research has been partially supported by the EU under FP7 project Optique (grant n. FP7-318338).

References

1. do Amaral, F.N.: Model outlines: A visual language for DL concept descriptions. Semantic Web J. 4(4), 429–455 (2013)
2. Antonioli, N., Castanò, F., Civili, C., Coletta, S., Grossi, S., Lembo, D., Lenzerini, M., Poggi, A., Savo, D.F., Virardi, E.: Ontology-based data access: the experience at the Italian Department of Treasury. In: Proc. of the Industrial Track of the 25th Int. Conf. on Advanced Information Systems Engineering (CAiSE 2013). CEUR Electronic Workshop Proceedings, http://ceur-ws.org/, vol. 1017, pp. 9–16 (2013)
3. Antonioli, N., Castanò, F., Coletta, S., Grossi, S., Lembo, D., Lenzerini, M., Poggi, A., Virardi, E., Castracane, P.: Ontology-based data management for the italian public debt. In: Proc. of the 8th International Conference on Formal Ontology in Information Systems (FOIS 2014) (2014), to Appear
4. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press, 2nd edn. (2007)
5. Bao, J., et al.: OWL 2 Web Ontology Language document overview (second edition). W3C Recommendation, World Wide Web Consortium (Dec 2012), available at http://www.w3.org/TR/owl2-overview/
6. Barzdins, J., Cerans, K., Liepins, R., Sprogis, A.: Advanced ontology visualization with OWLGrEd. In: Proc. of the 8th Int. Workshop on OWL: Experiences and Directions (OWLED 2011). CEUR Electronic Workshop Proceedings, http://ceur-ws.org/, vol. 796 (2011)
7. Bechhofer, S., van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D.L., Patel-Schneider, P.F., Stein, L.A.: OWL Web Ontology Language reference. W3C Recommendation, World Wide Web Consortium (Feb 2004), available at http://www.w3.org/TR/owl-ref/
8. Brockmans, S., Volz, R., Eberhart, A., Löffler, P.: Visual modeling of OWL DL ontologies using UML. In: Proc. of the 3rd Int. Semantic Web Conf. (ISWC 2004). pp. 198–213. Springer (2004)
9. Cerans, K., Ovcinnikova, J., Liepins, R., Sprogis, A.: Advanced OWL 2.0 ontology visualization in OWLGrEd. In: Databases and Information Systems VII – Selected Papers from the 10th Int. Baltic Conference (DB&IS). pp. 41–54 (2012)
10. Chein, M., Mugnier, M.L.: Graph-based Knowledge Representation: Computational Foundations of Conceptual Graphs. Springer (2009)
11. Civili, C., Console, M., De Giacomo, G., Lembo, D., Lenzerini, M., Lepore, L., Mancini, R., Poggi, A., Rosati, R., Ruzzi, M., Santarelli, V., Savo, D.F.: MASTRO STUDIO: Managing ontology-based data access applications. Proc. of the VLDB Endowment 6, 1314–1317 (2013)
12. Console, M., Lembo, D., Santarelli, V., Savo, D.F.: The Graphol language for ontology specification, manuscript, available from the authors. 2012
13. Cuenca Grau, B., Horrocks, I., Motik, B., Parsia, B., Patel-Schneider, P., Sattler, U.: OWL 2: The next step for OWL. J. of Web Semantics 6(4), 309–322 (2008)
14. Dau, F., Eklund, P.W.: A diagrammatic reasoning system for the description logic $\mathcal{ALC}$. J. Vis. Lang. Comput. 19(5), 539–573 (2008)
15. Dix, A., Finlay, J.E., Abowd, G.D., Beale, R.: Human-Computer Interaction (3rd Edition). Prentice Hall, 3 edn. (Dec 2003), http://www.worldcat.org/isbn/0130461091
16. Djuric, D., Gasevic, D., Devedzic, V., Damjanovic, V.: A UML profile for OWL ontologies. In: Revised Selected Papers of the European Workshop on Model Driven Architecture. pp. 204–219 (2004)

17. Fillottrani, P.R., Franconi, E., Tessaris, S.: The ICOM 3.0 intelligent conceptual modelling tool and methodology. Semantic Web J. 3(3), 293–306 (2012)
18. Guizzardi, G.: Ontological Foundations for Structural Conceptual Models. Ph.D. thesis, University of Twente, The Netherlands (2005)
19. Horridge, M., Patel-Schneider, P.F. (eds.): OWL 2 Web Ontology Language Manchester Syntax (Second Edition). W3C Working Group Note (11 december 2012), available at http://www.w3.org/TR/owl2-manchester-syntax/
20. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible $\mathcal{SROIQ}$. In: Proc. of the 10th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2006). pp. 57–67 (2006)
21. Katifori, A., Halatsis, C., Lepouras, G., Vassilakis, C., Giannopoulou, E.G.: Ontology visualization methods - a survey. ACM Comput. Surv. 39(4) (2007)
22. Krivov, S., Williams, R., Villa, F.: GrOWL: A tool for visualization and editing of OWL ontologies. J. of Web Semantics 5(2), 54–57 (2007)
23. Motik, B., Cuenca Grau, B., Horrocks, I., Wu, Z., Fokoue, A., Lutz, C.: OWL 2 Web Ontology Language profiles (second edition). W3C Recommendation, World Wide Web Consortium (Dec 2012), available at http://www.w3.org/TR/owl2-profiles/
24. Object Management Group: Ontology definition metamodel. Tech. Rep. formal/2009-05-01, OMG (2009), available at http://www.omg.org/spec/ODM/1.0
25. Staab, S., Studer, R. (eds.): Handbook on Ontologies. International Handbooks on Information Systems, Springer, 2nd edn. (2009)