Easy OWL Drawing with the Graphol Visual Ontology Language

Domenico Lembo, Daniele Pantaleone, Valerio Santarelli, Domenico Fabio Savo

Dipartimento di Ingegneria Informatica, Automatica e Gestionale "A. Ruberti"
Sapienza Università di Roma
⟨lastname⟩@dis.uniroma1.it

Abstract

GRAPHOL is a visual language designed to help non-experts to understand and specify ontologies. Our language builds on the Entity-Relationship model, but has a formal semantics and higher expressiveness. Notably, OWL 2 can be completely encoded in GRAPHOL. Thanks to the novel open-source Eddy ontology editor, designers can easily draw GRAPHOL diagrams corresponding to OWL ontologies and export them into standard OWL 2 format. Both GRAPHOL and Eddy have been used in several successful industrial projects and are currently under active development. This paper reports on our more recent progresses.

Introduction

Ontologies are conceptual representation schemes suited to share domain knowledge in a structured form. By virtue of these characteristics, they have become quite popular during the years in several areas, and have recently attracted the attention of various companies and organizations, even from the industrial world (Guarino and Musen 2015; Antonioli et al. 2014; Giese et al. 2015). A recurring major issue in these contexts concerns handling communication between ontologists and domain experts, who rarely possess the skills to interpret formalisms in which ontologies are usually expressed by the former.

In recent years, we have directly experienced the above problem in various projects carried out in collaboration with industrial partners or public organizations (see, e.g, (Antonioli et al. 2014; Savo et al. 2010)). In these projects we have conducted an extensive ontology modeling activity and we have developed significantly large and complex ontologies. In doing so we were always supported by domain experts, who, in many cases, were experiencing ontologies and languages like OWL for the first time, and had little or no skills in formal languages or in logic. Not surprisingly, to carry on the modeling we initially resorted to the use of some graphical representations for the ontology given through Entity-Relationship (ER) schemas or UML class diagrams. Indeed, these notations are well-known in industrial contexts for their intensive use in database and software design, and have a well-understood correspondence with for-

mal languages like Description Logics (DLs) (Berardi, Calvanese, and De Giacomo 2005). At the same time, they are quite easy to understand, at least in broad terms, even by people without technical skills, and they are well-suited to provide a general outline of the representation.

ER schemas and UML class diagrams, however, are not able in general to capture all the representation needs, which may require the use of very expressive languages like OWL 2.[1] In our first experiences we therefore used diagrams only to provide some views of the ontology, which we instead specified in terms of OWL formulas, with the support of tools like Protégé.[2] However, maintaining the diagrams aligned with the ontology turned out to be impractical as soon as the size of ontology grew. Furthermore, the use of two kinds of representations was confusing for domain experts, which wanted to also understand those axioms that were not expressible in diagrams, to be able to take over the ontology development in the middle/long run.

To overcome the above problems, we defined and started to use GRAPHOL, a new graphical language for ontology specification. GRAPHOL is designed to offer a completely visual representation of the ontology, and so it does not make use of textual formulas to complement the graphical model. To provide the users with constructs they are familiar with, we took the basic components of GRAPHOL from the ER model. In particular, we designed our language so that simple ontologies that can be captured by ER diagrams have in GRAPHOL a representation that is specular to the ER one. The structure of graphical assertions in GRAPHOL traces that of typical DL axioms, so that our language has a natural encoding in DL syntax. Through such encoding, we assign a clear and formal semantics, borrowed from DLs, to GRAPHOL, and we show that it captures $\mathcal{SROIQ}(D)$, the logical underpinning of OWL 2. Thus, our language can be readily seen as a graphical syntax for it.

Besides its important practical implications, our work contributes to a long-standing line of research in computer science, aimed at devising graphic notations for representing knowledge (Sowa 1984). Among many proposals, the closest to our approach are the UML-inspired graphical languages and tools for the representation of OWL
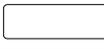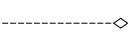
---

[1]https://www.w3.org/TR/owl2-syntax/
[2]http://protege.stanford.edu/

Symbol	Name	Symbol	Name	Symbol	Name	Symbol	Name
⬜ (rectangle)	Concept node	◇ (diamond)	Role node	◯ (circle)	Attribute node	▭ (rounded rectangle)	Value-domain node
⬡ (octagon)	Individual/ Value node	Restriction type ☐	Domain restriction node	Restriction type ■	Range restriction node	→	Inclusion edge
(and)	Intersection node	(or)	Union node	(not)	Complement node	┄┄┄◇	Input edge
(inv)	Inverse node	(oneOf)	One-of node	(chain)	Role chain node		

Figure 1: GRAPHOL predicate nodes, constructor nodes, and edges.

ontologies (Brockmans et al. 2004; Djuric et al. 2004; Guizzardi 2005; Fillottrani, Franconi, and Tessaris 2012; Liepins, Grasmanis, and Bojars 2014). Differently from GRAPHOL, none of them is completely graphical, and typically use anonymous UML classes associated with formulas to encode complex OWL expressions. Other proposals that are not based on standard conceptual modeling languages, like (Krivov, Williams, and Villa 2007; Stapleton et al. 2013; Falco et al. 2014), use structures such as graphs or concept diagrams to model ontologies. These proposals however have either been discontinued or are not able to completely capture OWL 2. Furthermore, the notation used in some of these approaches is quite distant in nature from languages like ER or UML class diagrams, and this may pose some difficulties in its understanding, especially in the enterprise context.

In the following, we first describe the GRAPHOL language, and then introduce Eddy, a new tool for the design of GRAPHOL ontologies. We then briefly discuss future work and conclude the paper.

The Graphol language

In GRAPHOL, as in DLs, an ontology is composed by inclusion assertions between expressions, whose representation is completely visual and is given in terms of a graph. Each expression is a weakly-connected acyclic directed (sub-)graph with exactly one node with no outgoing edges, called *sink*. Each such graph contains only nodes of the forms showed in Figure 1, connected with dashed directed edges terminating with a small diamond, called *input edges*. An inclusion is drawn with a solid directed arrow, called *inclusion edge*, going from the (sink node of the) subsumed expression to the (sink node of the) subsuming one. Expressions are of four types: concept, role, attribute, and value-domain. Inclusions are allowed only between expressions of the same type.

Legal GRAPHOL expressions are defined inductively. Base expressions are labeled rectangles, diamonds, circles, and rounded rectangles, which represent the atomic concepts, roles, attributes, and value-domains of the ontology, respectively. These are called *predicate nodes*. In this case, the only node in the expression is also its sink. Complex expressions are obtained through *constructor nodes* taking as input other expressions, which means that an input edge is traced from the sink node of each expression to the constructor node. There are various types of constructor nodes,

defined with only two graphical labeled shapes: box (which can be either blank or solid) and hexagon. Labels on boxes (Restriction type in Figure 1) can have one of the following values: "exists", for existential restriction, "forall", for value restriction, "self", for self restriction, and "$(x, -)$" and "$(-, y)$", with x and y positive integers, for min and max cardinality restrictions (Baader et al. 2007). The above restrictions can be specified over a role (resp. an attribute) by using a blank box that takes as input the role (resp. attribute) expression and a concept expression (missing for self-labeled boxes, and possibly missing for existential and cardinality restrictions). In this case we are constructing a concept expression. For example, to specify in GRAPHOL the DL min cardinality restriction $\geq 1 hasPassed.Exam$, denoting all individuals that have passed at least an exam, we use a blank box labeled with $(1, -)$ taking as input the role $hasPassed$ and the concept $Exam$. Analogously, with solid boxes we specify restrictions over role inverses. Furthermore, a solid box with an input attribute denotes the attribute range, which is a value-domain expression.

Hexagons can be labeled as indicated in Figure 1. These are used to represent the conjunction or disjunction of concepts or value-domains (*and* and *or*), the complement of a predicate (*not*), the inverse of a role (*inv*), the chain of roles (*chain*), or an enumerated concept or value-domain (*oneOf*). In this last case, the operator takes as input only individual or value nodes, represented by lebeled octagons (cf. Figure 1). The inverse and chain operators take as input only role expressions and construct other role expressions (for the chain operator, input edges are labeled with positive integers that indicate the order of the input expression in the chain). All the other constructor nodes define expressions of different kind, depending on the input they take. For example, the conjunction node taking as input two (or more) concept expressions returns a concept expression. We refer the reader to (Console et al. 2014) for more details.

We notice that the GRAPHOL syntax reflects that of DLs, and thus it is not difficult to transform each GRAPHOL assertion into a DL inclusion axiom. This allows us to assign GRAPHOL with a formal semantics, borrowed from DLs (Console et al. 2014). Notably, we can also express in GRAPHOL any $\mathcal{SROIQ}(D)$ (and thus OWL 2) TBox (Horrocks, Kutz, and Sattler 2006). Indeed, any such TBox can be expressed in terms of $\mathcal{SROIQ}(D)$ inclusion axioms only (thus eliminating some syntactic sugar), and then
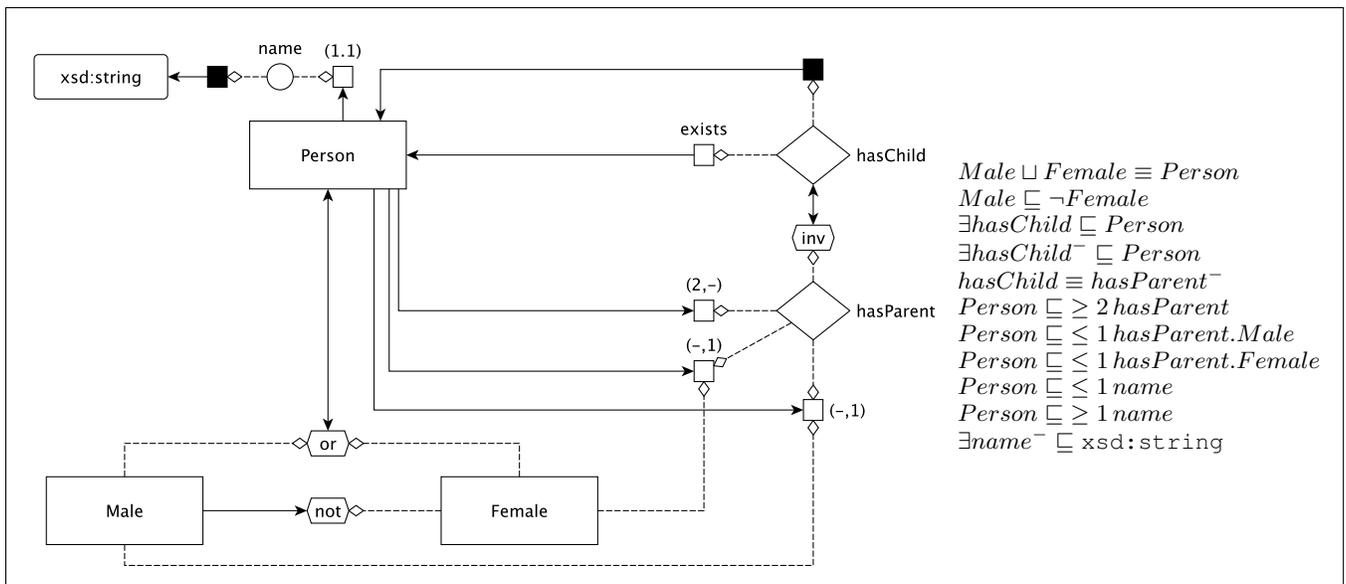
Figure 2: Example of a GRAPHOL ontology and its representation in DL.

straightforwardly translated in GRAPHOL.

An example of a GRAPHOL ontology is given in Figure 2, where we also provide its specification in terms of DL formulas. For simplicity, inclusion edges between the same expressions and with inverse directions are denoted with a single solid edge with an arrow on both ends, and exact cardinalities are specified through a single box. In words, the ontology asserts that a person is either a Male or Female and that each person as exactly one name, which is a string. Only persons can have children, which are in turn persons, and each person has one parent that is Male and one that is a Female. Furthermore, $hasParent$ is the inverse of $hasChild$.

We notice that the ontology in the figure resembles an ER diagram, thanks to the choice of using for concepts, roles, and attributes the same symbols used in ER. At the same time, however, we are able here to visually represent knowledge that cannot be expressed in ER (for example that $hasParent$ is the inverse of $hasChild$), but that can be easily specified in ontology languages.

Introducing Eddy

Eddy is a novel graphical editor specifically developed to support the design of GRAPHOL ontologies through ad-hoc functionalities.[3] In the design environment it offers, drawing features allow designers to comfortably edit ontologies in a central viewport area, while two lateral docking areas contain specifically-tailored widgets for editing, navigation, and inspection of the diagram, as shown in Figure 3.

The GRAPHOL palette in the left-hand side docking area provides all GRAPHOL nodes and edges. Users can select a node and insert it into the diagram through a point-and-click mechanism, or choose an edge and anchor it between two

nodes by drag and drop. To improve the layout of the diagram, Eddy allows to bend edges by adding breakpoints, to resize predicates nodes, and to move the labels of predicate nodes from their default central position inside the node. More advanced drawing functionalities, available through contextual menu and keyboard shortcuts, provide support for rapid creation of commonly recurring GRAPHOL expressions, e.g., restrictions on roles or attributes through domain or range restriction nodes, and for easily specifying properties on roles, such as symmetry, reflexivity, transitivity.

Our tool is also equipped with refactoring functionalities for predicate renaming. Indeed, in order to ease the task of ontology design and reading, we allow to use multiple predicate nodes with the same label, obviously all representing the same atomic predicate. Eddy thus offers the possibility of automatically applying the renaming of a predicate to all the nodes representing such predicate in the ontology.

In Eddy, syntactic validation functionalities do not allow to build expressions and assertions outside the GRAPHOL syntax. We envision to extend these features by allowing the user to restrict the expressiveness of her ontology to a language subsumed by OWL 2, for example to one of its profiles.[4]

GRAPHOL ontologies designed in Eddy can be automatically exported into OWL 2 ontologies expressed in the standard functional-style syntax.

Future Work and Conclusion

GRAPHOL and Eddy have been and are currently being used in various industrial and academic projects by teams of ontology designers for the specification and maintenance of domain ontologies (see, e.g., (Antonioli et al. 2014)). As already discussed, these projects have actually been a source
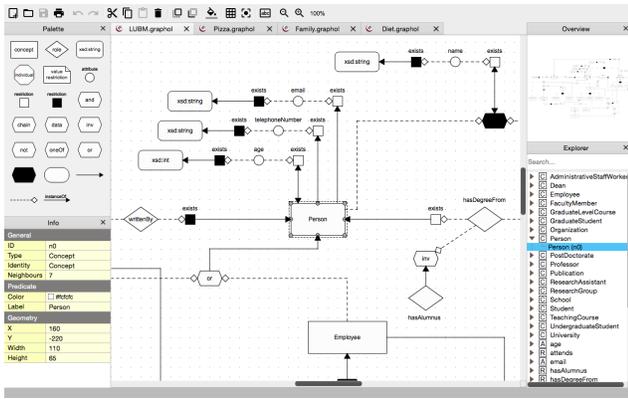
---

[3]Eddy is distributed under the GPL v3 license at http://www.dis.uniroma1.it/~graphol/download.html.

[4]https://www.w3.org/TR/owl2-profiles/

Figure 3: Eddy viewport and docking areas.

of inspiration for GRAPHOL and have provided useful insights for the development of our tool. At the same time, they have shown GRAPHOL's validity as a language for ontology design, as it has played a crucial role in their successful accomplishment.

We also point out that many of the choices we adopted for both GRAPHOL and Eddy are based on the feedback deriving from usability tests conducted on the language and on various versions of the tool (Console et al. 2014).

Our current work is mainly focused on the tool Eddy, for which there are various major upgrades in the pipeline.

The first is to equip Eddy with semantic reasoning capabilities through integration of external OWL 2 reasoners. This will allow to validate the ontology, for instance by checking its consistency, identifying unsatisfiable predicates, and retrieving explanations of such malformations, possibly in a graphical form. Support to highlight GRAPHOL axioms that are logically implied by other portions of the ontology will be also provided. This is, for instance, useful to identify redundancies in the specification.

A further issue we intend to face is the import of ontologies expressed in OWL 2 into Eddy. In this case the problem is devising effective techniques for the automatic positioning of the elements of the ontology starting from a representation composed merely by a collection of formulas without any information regarding size and placement. Achieving this result is particularly challenging because the difficulty is not only in coming up with a placement without overlapping elements, but also with a visualization that reflects the semantic connections between them.

We also plan to equip the tool with more advanced functions for ontology inspection. In particular, we would like to have mechanisms that allow the user to access the ontology at various levels of detail, and to extract and visualize only partial views of the ontology. These features are particularly complex to achieve, since they require investigation on both logical and visualization aspects.

References

Antonioli, N.; Castanò, F.; Coletta, S.; Grossi, S.; Lembo, D.; Lenzerini, M.; Poggi, A.; Virardi, E.; and Castracane, P. 2014. Ontology-based data management for the italian public debt. In *Proc. of FOIS*, 372–385.

Baader, F.; Calvanese, D.; McGuinness, D.; Nardi, D.; and Patel-Schneider, P. F., eds. 2007. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2nd edition.

Berardi, D.; Calvanese, D.; and De Giacomo, G. 2005. Reasoning on UML class diagrams. *AIJ* 168(1–2):70–118.

Brockmans, S.; Volz, R.; Eberhart, A.; and Löffler, P. 2004. Visual modeling of OWL DL ontologies using UML. In *Proc. of ISWC*, volume 3298 of *LNCS*, 198–213. Springer.

Console, M.; Lembo, D.; Santarelli, V.; and Savo, D. F. 2014. Graphol: Ontology representation through diagrams. In *Proc. of DL*, volume 1193 of *CEUR,* ceur-ws.org, 483–495.

Djuric, D.; Gasevic, D.; Devedzic, V.; and Damjanovic, V. 2004. A UML profile for OWL ontologies. In *Revised Selected Papers of the European Workshop on Model Driven Architecture*, 204–219.

Falco, R.; Gangemi, A.; Peroni, S.; Shotton, D.; and Vitali, F. 2014. Modelling OWL ontologies with Graffoo. In *ESWC 2014 Satellite Events*, volume 8798 of *LNCS*, 320–325.

Fillottrani, P. R.; Franconi, E.; and Tessaris, S. 2012. The ICOM 3.0 intelligent conceptual modelling tool and methodology. *Semantic Web* 3(3):293–306.

Giese, M.; Soylu, A.; Vega-Gorgojo, G.; Waaler, A.; Haase, P.; Jiménez-Ruiz, E.; Lanti, D.; Rezk, M.; Xiao, G.; Özçep, Ö. L.; and Rosati, R. 2015. Optique: Zooming in on Big Data. *IEEE Computer* 48(3):60–67.

Guarino, N., and Musen, M. A. 2015. Applied Ontology: The next decade begins. *Applied Ontology* 10(1):1–4.

Guizzardi, G. 2005. *Ontological Foundations for Structural Conceptual Models*. Ph.D. Dissertation, University of Twente, The Netherlands.

Horrocks, I.; Kutz, O.; and Sattler, U. 2006. The even more irresistible $\mathcal{SROIQ}$. In *Proc. of KR*, 57–67.

Krivov, S.; Williams, R.; and Villa, F. 2007. GrOWL: A tool for visualization and editing of OWL ontologies. *J. of Web Semantics* 5(2):54–57.

Liepins, R.; Grasmanis, M.; and Bojars, U. 2014. OWLGrEd ontology visualizer. In *ISWC Developers Workshop 2014*, volume 1268 of *CEUR,* ceur-ws.org, 37–42.

Savo, D. F.; Lembo, D.; Lenzerini, M.; Poggi, A.; Rodríguez-Muro, M.; Romagnoli, V.; Ruzzi, M.; and Stella, G. 2010. MASTRO at work: Experiences on ontology-based data access. In *Proc. of DL*, volume 573 of *CEUR,* ceur-ws.org, 20–31.

Sowa, J. F. 1984. *Conceptual Structures: Information Processing in Mind and Machine*. Addison Wesley Publ. Co.

Stapleton, G.; Howse, J.; Taylor, K.; Delaney, A.; Burton, J.; and Chapman, P. 2013. Towards diagrammatic ontology patterns. In *Proc. of WOP*.