

Logic Graphs: Complete, Semantic-Oriented and Easy to Learn Visualization Method for OWL DL Language

Than Nguyen^a, Ildar Baimuratov^a

^aITMO University, Kronverksky Pr. 49, bldg. A, St. Petersburg, 197101, Russian Federation

Abstract

Visualization of an ontology is intended to help users to understand and analyze the knowledge it contains. There are numerous ontology visualization systems, however, they have several common drawbacks. First, most of them do not allow representing all required ontological relations. We consider the OWL DL language, as it is sufficient for most practical tasks. Second, they visualize ontological structures just labeling them, not representing their semantics. Finally, the existing ontology visualization systems mostly use arbitrary graphic primitives. Thus, instead of helping a user to understand an ontology, they just represent it with another language. In opposite, there are semantic-oriented visualization systems like Conceptual graphs, but they correspond to First-order logic, not to OWL DL. Therefore, our goal is to develop an ontology visualization method, named "Logic graphs", with three features. First, it should be complete with respect to OWL DL. Second, it should represent the semantics of ontological relation. Finally, it should apply existing graphic primitives, where it is possible. Previously, we adopted Ch. S. Peirce's Existential graphs for visualizing ALC description logic formulas. In this paper, we extend our visualization system for SHIF and SHOIN description logics, based on Venn diagrams and graph theory. The resulting system is complete with respect to the OWL DL language and allows visualizing the semantics of ontological relations almost without new graphic primitives.

Keywords

Visualization, Ontology, Description logic, Existential graph, Venn diagrams, Graph theory,

1. Introduction

In recent years, ontologies have been widely used to capture comprehensive domain knowledge in different areas. "Ontologies define the concepts and relationships used to describe and represent an area of knowledge" [1]. Ontology visualization is intended to display the concepts and structures of an ontology to help users to understand and analyze knowledge they contain.

There are numerous ontology visualization systems, the reviews can be found in [2, 3, 4]. However, these systems have several common drawbacks. First, most of them don't allow representing all required ontological relations. In particular, we consider the OWL DL language [5], as it is sufficient for most practical tasks. Thus, in other words, they are incomplete with respect to the OWL DL language.

Proceedings of the 12th Majorov International Conference on Software Engineering and Computer Systems, December 10-11, 2020, Online Saint Petersburg, Russia

✉ nguyennngocthan92@gmail.com (T. Nguyen); baimuratov.i@gmail.com (I. Baimuratov)

🆔 0000-0002-6679-7839 (T. Nguyen); 0000-0002-6573-131X (I. Baimuratov)



© 2020 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

Second, they visualize ontological structures just labeling them, not representing their semantics. For example, compare the visualization of conjunction from Graphol [6], Fig.1, with the corresponding Venn diagram, Fig. 2. Venn diagram represents by itself that these two sets have common elements, while in Graphol a user has to know that hexagon denotes conjunction.

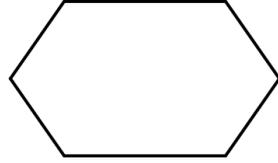


Figure 1: Conjunction in Graphol

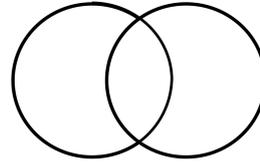


Figure 2: Conjunction in Venn diagrams

Finally, the existing ontology visualization systems mostly use arbitrary graphic primitives. Thus, instead of helping a user to understand an ontology, they just represent it with another language. Considering again the example above, in Graphol the user has to learn that hexagon denotes conjunction. It would be friendlier to use existing graphic primitives from mathematic theories as the user won't have to learn new ones.

On the other hand, there are semantic-oriented visualization systems like Conceptual graphs [7] or Concept graphs [8], but they correspond to First-order logic, not to the OWL DL language.

Therefore, our goal is to develop a complete ontology visualization method, named "Logic graphs" (LGs), with two features. First, it should represent the semantics of the ontological structures, formulated as description logic axioms. Second, it should apply existing graphic primitives from mathematical theories, where it is possible.

In our previous work [9], we adopted the Ch. S. Peirce's Existential graph [10, 11, 12] for description logic formulas. As the result, we developed the visualization method for ALC description logic. In this paper, we extend our visualization system for SHIF and SHOIN description logics.

The outline of the paper is as follows: Section 2 describes the OWL sub-languages and the corresponding description logics. In Section 3, we review the state-of-the-art ontology visualization tools and discuss several drawbacks they have. Section 4 gives a short description of the visualization methods used in mathematics that we apply in our system. Section 5 presents the Logic graph system in detail. In Section 6, we visualize some axioms of DoCO ontology using the developed method, before the paper is concluded in Section 7.

2. OWL sub-languages and Description logics

In this section we describe the OWL DL language [5] and its logical foundation – Description logics (DLs) [13] as the object language to be visualized.

Ontologies are denoted on the OWL language. The Web Ontology Language OWL is a semantic markup language for publishing and sharing ontologies on the World Wide Web. The OWL 1 standard provided three increasingly expressive sub-languages: OWL Lite, OWL DL, and OWL Full. OWL Lite supports those users primarily needing a classification hierarchy and simple constraints. OWL DL provides the maximum expressiveness, retaining computational

completeness (all conclusions are guaranteed to be computed) and decidability (all computations will finish in finite time). OWL DL includes all OWL language constructs, but they can be used only under certain restrictions. Finally, OWL Full is meant for users who want maximum expressiveness and the syntactic freedom of RDF with no computational guarantees. Each of these sub-languages is an extension of its simpler predecessor.

The formal foundation of OWL is description logics. Description Logics (DLs) are a family of logic languages, which can be used to represent the terminological knowledge of an application domain. In this paper, we consider the OWL DL language, as it is sufficient for most practical tasks, and the corresponding SHOIN description logic. Its syntax and semantics is represented in Table 1, where I is an interpretation function and Δ is a domain. Therefore, our goal is to develop the visualization system that would be complete with respect to the SHOIN DL and, consequently, the OWL DL language.

Table 1
SHOIN description logic.

Name	Syntax	Semantic
concept	C	$C^I \subseteq \Delta^I$
role	R	$R^I \subseteq \Delta^I \times \Delta^I$
negation	$\neg C$	$\Delta^I \setminus C^I$
conjunction	$C \sqcap D$	$C^I \cap D^I$
disjunction	$C \sqcup D$	$C^I \cup D^I$
existential restriction	$\exists R. \top$	$(\exists R. \top)^I = \{a \in \Delta^I : \exists b (a, b) \in R^I\}$
universal restriction	$\forall R. \top$	$(\forall R. \top)^I = \{a \in \Delta^I : \forall b (a, b) \in R^I \rightarrow b \in C^I\}$
inverse role	R^-	$(R^-)^I = \{(b, a) \in \Delta^I \times \Delta^I : (a, b) \in R^I\}$
transitive role	R^+	$(R^+)^I = \cup_i (R^I)_i$, where $(a, b) \in (R^I)_i \wedge (b, a) \in (R^I)_i \rightarrow (a, c) \in (R^I)_i$
functional role	$\leq 1R$	$\leq 1R \Leftrightarrow \{(a, b), (a, c)\} \subseteq R^I \Rightarrow b = c$
role inclusion	$R \sqsubseteq S$	$R^I \subseteq S^I$
nominal	$\{o\}$	$\{o^I\}$
number restrictions	$\geq nP$ $\leq nP$	$\{a \in \Delta^I \mid \{b \mid (a, b) \in R^I\} \geq n\}$ $\{a \in \Delta^I \mid \{b \mid (a, b) \in R^I\} \leq n\}$

3. Related works

3.1. Ontology Visualization Tools

According to [2, 3, 4], there are visualization tools such as TGViz, OntoTrack, KC-Viz, OntoRama, Ontodia, OntoTrix, Flexviz, OntoGraf, GraphViz, Graffoo, OWLviz, NavigOWL, Glow, SOVA, Jambalaya, Gephi, VOWL, Graphol, OntoSphere, and Tarsier. For each tool, we examined the required properties, namely its completeness with respect to the OWL DL language, its “semanticality”, the ability to represent the semantics of relations, and whether its graphic primitives are new or adopted from common visualization systems.

As the result, some visualization tools like OWLViz, OntoTrack, KC-Viz, and OntoRama visualize only the class hierarchy. GLOW, OntoGraf, FlexViz, and OWLViz represent various types of property relations, but do not show property characteristics, required to fully understand the information modeled in ontologies.

Although some visualization tools, such as Graphol or SOVA, show complete ontology information, i.e. all classes and properties along with their attributes, these systems still have several drawbacks. First, they visualize ontological structures by simply labeling them, without representing their semantics, and second, existing ontology visualization systems mainly use arbitrary graphic primitives. As a result, instead of helping the user to understand the ontology, he or she has to learn one more language.

3.2. Conceptual graphs

On the other hand, there are other works on semantic-oriented visual frameworks. One of the most known examples is the Conceptual graphs. The Conceptual graphs (CGs) is the family of formalisms that originates from J. Sowa's work [8], which in turn is based on Ch. S. Peirce's existential graphs [10, 11, 12], where knowledge is represented by labeled graphs and reasoning mechanisms are based on graph operations. However, CGs don't fit our purpose, as they are found on First-order logic. According to [7], the "intersection" between Description logics (DLs) and CGs is less expressive than each formalism. For instance, DLs can't express n -ary relations for $n > 2$, while in CGs one can't represent value restrictions. The variation of CGs are F. Dau's Concept graphs [14], but they are still based on First-order logic.

In [15], the authors propose the visualization system based on the Ch. S. Peirce's existential graphs for the ALC description logic exactly, but it is not sufficient for representing the OWL DL language.

4. Visualization methods used in mathematics

Developing our visualization method, we suggest applying existing graphic primitives from mathematical theories, such as Venn diagrams [16], Ch. S. Pierce's existential graphs [10, 11, 12] and diagrams from the graph theory [17], where it is possible, in order to make the method easy to learn, as the user won't have to learn new ones.

4.1. Venn diagrams

The Venn diagrams depict elements as points in the plane and sets as regions inside closed curves. A Venn diagram consists of multiple overlapping closed curves, usually circles, each representing a set. We use Venn diagrams-based intuitions to denote concept equivalence and belonging of an individual to a concept as such, as shown in the Fig. 3, Fig. 4, Fig. 5 .

4.2. Existential graphs

An existential graph is a type of visual notation for logical expressions, proposed by Charles Sanders Peirce. Peirce presents the graphs as three general systems called "Alpha", "Beta" and

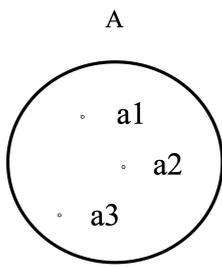


Figure 3: Membership

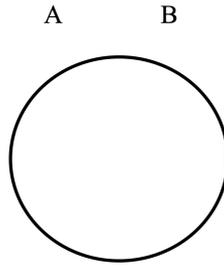


Figure 4: Equivalence

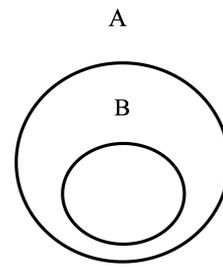


Figure 5: Inclusion

“Gamma”. This division corresponds fairly well to the contemporary proposition, predicate, and modal logic respectively. As description logic can be considered as a fragment of predicate logic, the system Beta is the most relevant. In this system blank space is considered as the “sheet of assertion”, or the space of true expressions, therefore, expressions, situated on this sheet are conjuncted. Besides, there are “cuts” on the sheet that mean inversion of the expressions, situated on these cuts, and curves, denoting predicates, existentially quantified by default. As any complex expression can be represented with only conjunction, negation, and existential quantifier, these two graphic primitives are sufficient for visualizing any expression of predicate logic. The Beta existential graph system is represented in Table 2, where a and b are propositions, P and R – predicates, and s and t – individuals. We use existential graphs for depicting logical relations: negation, conjunction, and their derivatives.

4.3. Graph theory

A graph is graphically represented by drawing a point for each vertex and drawing an arc between two vertices if they are joined by an edge. If the graph is oriented (digraph), then the direction is indicated by an arrow. We use graph-based intuitions to represent inverse and transitive roles.

In graph theory, an inverse relation is depicted with an arrow between two vertices A and B , such that it is oriented from the vertex A to the vertex B . See Fig. 6. Transitivity is the same as saying there must be a direct arrow from vertex A to another vertex C , if one can walk from that vertex to the last one through a list of arrows, travelling always along the direction of the arrows. See Fig. 7.



Figure 6: Inversion

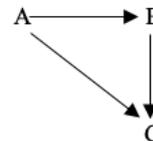
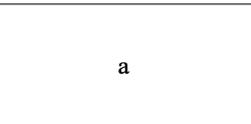
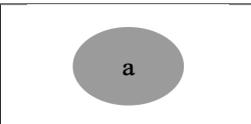
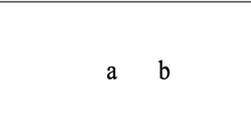
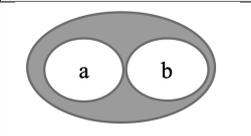
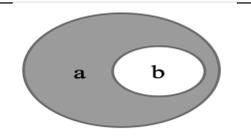
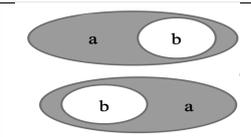
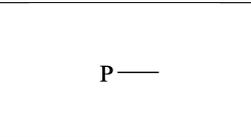
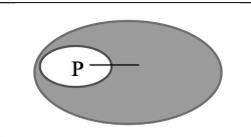
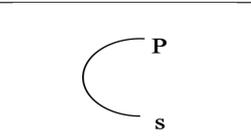
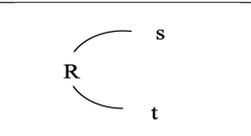


Figure 7: Transitivity

Table 2
Existential graphs.

1.	a		2.	$\neg a$	
3.	$a \wedge b$		4.	$a \vee b$	
5.	$a \supset b$		6.	$a \equiv b$	
7.	$\exists xP(x)$		8.	$\forall xP(x)$	
9.	$P(s)$		10.	$R(s, t)$	

5. Logic graphs

We propose the complete ontology visualization method, named “Logic graphs”, aimed to represent the semantics of description logic axioms and to be easy to learn, due to using existing graphic primitives.

5.1. ALC

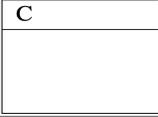
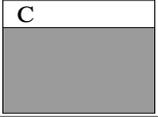
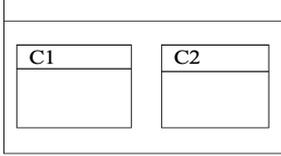
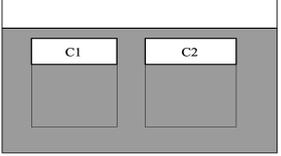
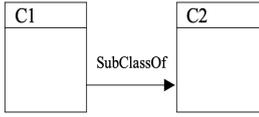
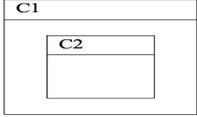
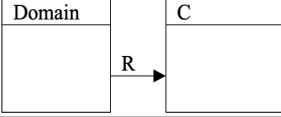
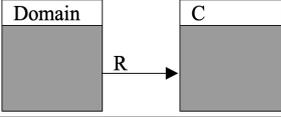
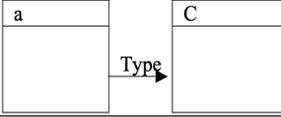
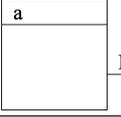
In our previous work [9], we adopted the existential graph visualization method for description logic formulas. As the result, we developed the visualization method for ALC description logic. It is represented in Table 3, where C , $C1$, and $C2$ are concepts, R – role and a and b – objects. *Domain* – the technical concept, introduced to represent domain concepts of roles. The exact form of the figures corresponding to concepts is not crucial, here we use rectangles for practical reasons

5.2. Logic SHIF

In this paper, we extend our visualization system for SHIF and SHOIN description logics. SHIF extends ALC with inverse, transitive, and functional roles and role inclusion. In order to depict inverse and transitive roles, we suggest applying the graph-theoretic intuitions.

An inverse role R^- corresponds to an inversely directed arrow. See Fig. 8. If a role R^+ is

Table 3
Logic graphs for ALC.

1.	C		2.	$\neg C$	
3.	$C1 \sqcap C2$		4.	$C1 \sqcup C2$	
5.	$C1 \sqsubseteq C2$		6.	$C1 \equiv C2$	
7.	$\exists R.C$		8.	$\forall R.C$	
9.	$a : C$		10.	$(a, b) : R$	

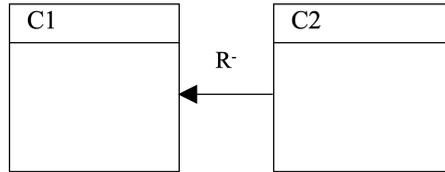


Figure 8: Inverse roles

transitive and a corresponding edge connects a node A with a node B and a node B with a node C , then it connects the nodes A and C . See Fig. 9.

Unfortunately, we suppose there is no simpler way to represent functional roles, than labeling the corresponding property arrow with “ \leq ” notation. See Fig. 10.

For role inclusion, we suggest not to introduce a new graphic primitive, but to represent it through its semantics. The semantics of role inclusion is following

$$R \sqsubseteq S \Leftrightarrow R^I \subseteq S^I \quad (1)$$

According to the definition of a role,

$$R^I \subseteq S^I \Leftrightarrow (a, b) \in R^I \Rightarrow (a, b) \in S^I \quad (2)$$

Let us define the set X such that

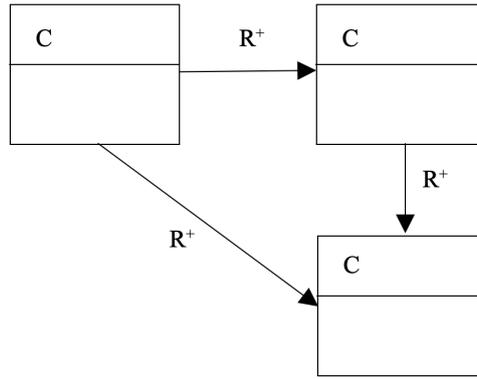


Figure 9: Transitive roles

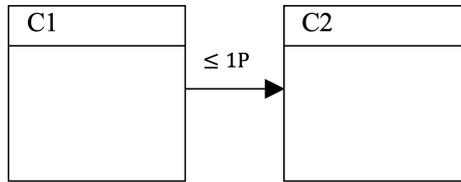


Figure 10: Functional roles

$$X = \{a : \exists b (a, b) \in R^I\} \quad (3)$$

and the set Y such that

$$Y = \{a : \exists b (a, b) \in S^I\} \quad (4)$$

Theorem 1. $R \sqsubseteq S \Rightarrow X \sqsubseteq Y$

Proof. According to the definition of inclusion, the theorem can be represented as follows: $R^I \subseteq S^I \Rightarrow X \sqsubseteq Y$. According to the definition of X, for every a if $a \in X \Rightarrow (a, b) \in R^I$, then, according to the property (2), $(a, b) \in S^I$ and, finally, according to the definition of Y, $a \in Y$. Therefore, $a \in X \Rightarrow a \in Y$ for every a, and $X \sqsubseteq Y$. \square

As the result, we propose to depict role inclusion $R \sqsubseteq S$ with concept inclusion $X \sqsubseteq Y$, as presented in the Fig. 11. Thus we extended Logic graphs for SHIF description logic.

5.3. Logic SHOIN

SHOIN extends SHIF with nominals and number restrictions. To represent a nominal, we apply Venn diagrams intuitions. We suggest representing each individual, forming the nominal, as a point. For example, see Fig. 12.

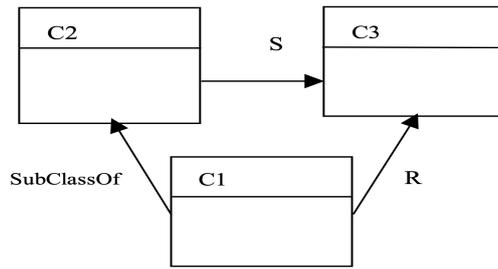


Figure 11: Logic graph for role inclusion

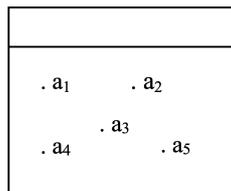


Figure 12: Nominal

Again, we suppose there is no simpler way to represent number restrictions, than labeling them with the exact, minimum, or maximum cardinality restrictions written as numbers above the property arrow. See Fig. 13. Thus, we extended Logic graphs for SHOIN description logic and, therefore, the method is complete with respect to the OWL DL language.

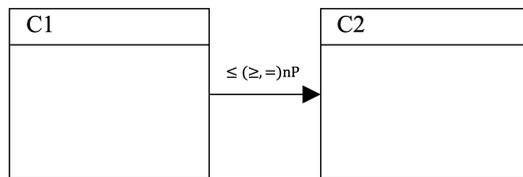


Figure 13: Number restriction

6. Logic graphs for the DoCO ontology

The Document Components Ontology (DoCO) [18] is an ontology that provides a structured vocabulary for document components. We use the DoCO ontology as an example for visualization, as it is a real ontology, used in different applications and it contains nontrivial axioms. We visualized some axioms using the developed method. See Table 4.

Table 4
Logic graphs for the DoCO ontology.

	$\text{abstract} \sqsubseteq (\text{chapter} \sqcup \text{section})$ $\sqcap (\exists \text{ispartof}.\text{bodymatter} \sqcup \text{frontmatter})$
	$\text{afterword} \sqsubseteq \text{section} \sqcap$ $\exists \text{ispartof}.\text{backmatter}$
	$\text{appendix} \sqsubseteq (\text{section} \sqcap$ $\text{headedcontainer}) \sqcap$ $(\exists \text{ispartof}.\text{backmatter})$
	$\text{backmatter} \sqsubseteq \text{discourseelement}$ $\sqcap \text{container}$
	$\text{backmatter} \sqsubseteq \forall \text{iscontainedby}$ $(\neg \text{backmatter} \sqcup \text{bodymatter} \sqcup \text{frontmatter})$
	$\text{Blockquotation} \sqsubseteq \text{container}$
	$\text{chapter} \sqsubseteq \exists \text{contains}.\text{paragraph}$ $\sqcup \text{section}$
	$\text{chapter} \sqsubseteq \exists \text{contains}.\neg(\text{chapter})$
	$\text{chapterlabel} \sqsubseteq \neg \text{sectionlabel}$
	$\text{chaptersubtitle} \sqsubseteq \exists \text{ispartof}.\text{chapter}$
	$\text{figure} \sqsubseteq \text{meta} \sqcup \text{milestone}$
	$\text{header} \equiv \text{frontmatter}$
	$\text{glossary} \sqsubseteq \text{section} \sqcap$ $(\exists \text{ispartof}.\text{backmatter} \sqcup \text{frontmatter})$
	$\text{list} \sqsubseteq \forall \text{contains}.\text{block} \sqcup \text{field} \sqcup$ $(\text{container} \sqcap (\neg(\text{headedcontained} \sqcup \text{table})))$
	$\text{tableofcontents} \sqsubseteq$ $\exists \text{haspart}.\text{listofreferences} \sqcap$ $(\forall \text{contains}.\exists \text{relation}.\text{section})$
	$\text{table} \sqsubseteq \text{table}$

7. Conclusion and Future works

The developed visualization method, named “Logic graphs”, is complete with respect to the OWL DL language. It allows visualizing the semantics of logical expressions almost without new graphic primitives, except for functional roles and number restrictions. As the result, a user familiar with the set theory, graph theory and description logic will be able to use Logic graphs without additional instructions. Several examples were provided considering the DoCO ontology.

In further research, we are going, first, to perform formal quantitative evaluation of Logic graphs and compare them with other visualization systems. Second, we are going to develop an ontology visualization application, implementing LGs, based on the Ontodia library [19]. Finally, we are going to perform a usability study to assess if LGs are more convenient for ontology practioners.

References

- [1] W3C, W3c semantic web faq, 2009. URL: <https://www.w3.org/RDF/FAQ>.
- [2] F. Antoniazzi, F. Viola, Rdf graph visualization tools: A survey, in: 2018 23rd Conference of Open Innovations Association (FRUCT), IEEE, 2018, pp. 25–36.
- [3] M. Dudáš, S. Lohmann, V. Svátek, D. Pavlov, Ontology visualization methods and tools: a survey of the state of the art, *The Knowledge Engineering Review* 33 (2018).
- [4] S. Lohmann, S. Negru, F. Haag, T. Ertl, Visualizing ontologies with vowl, *Semantic Web* 7 (2016) 399–419. doi:10.3233/SW-150200.
- [5] D. L. McGuinness, F. Van Harmelen, et al., Owl web ontology language overview, W3C recommendation 10 (2004) 2004.
- [6] Graphol, Graphol official website, 2014. URL: <Http://www.obdasystems.com/graphol>.
- [7] M. Chein, M.-L. Mugnier, Graph-based knowledge representation: computational foundations of conceptual graphs, Springer Science & Business Media, 2008.
- [8] J. F. Sowa, Conceptual graphs for a data base interface, *IBM Journal of Research and Development* 20 (1976) 336–357.
- [9] D. Mouromtsev, I. Baimuratov, Logic graphs: A complete visualization method for logical languages based on ch. s. peirce’s existential graphs, in: *CEUR Workshop Proceedings*, volume 2344, 2019, pp. 1–10.
- [10] C. Hartshorne, P. Weiss, A. W. Burks, et al., *Collected Papers of Charles Sanders Peirce*, volume 8, Belknap Press of Harvard University Press, 1958.
- [11] R. S. ROBIN, *Annotated catalogue of the papers of charlew s. peirce*, 1967.
- [12] J. F. Sowa, Peirce’s tutorial on existential graphs, *Semiotica* 2011 (2011) 347–394.
- [13] F. Baader, D. Calvanese, D. McGuinness, P. Patel-Schneider, D. Nardi, et al., *The description logic handbook: Theory, implementation and applications*, Cambridge university press, 2003.
- [14] F. Dau, *The logic system of concept graphs with negation: And its relationship to predicate logic*, volume 2892, Springer Science & Business Media, 2003.

- [15] F. Dau, P. Eklund, A peirce-style calculus for alc, VISUAL LANGUAGES AND LOGIC (2007) 55.
- [16] M. W. Frank Ruskey, A survey of venn diagrams, The Electronic Journal of Combinatorics 1000 (2005).
- [17] R. Trudeau, Introduction to graph theory. Dover Pubns, 1994.
- [18] Ontology DoCo, The document components ontology (doco), 2016. URL: <https://sparontologies.github.io/doco/current/doco.html>.
- [19] Ontodia, Ontodia homepage, 2015. URL: <http://ontodia.org/>.