Logic Graphs: A complete visualization method for logical languages based on Ch. S. Peirce's existential graphs

Dmitry Mouromtsev and Ildar Baimuratov

ITMO University, 49 Kronverksky Pr., 197101 St. Petersburg, Russia
d.muromtsev@gmail.com, baimuratov.i@gmail.com

Abstract. Contemporary tools of graph data visualization do not allow to work with logic expressions completely. The purpose of our research is to develop a complete and convenient method for its visualization. The developed prototype is intended for realization in the Ontodia library. We analyze the tools of Ontodia with respect to description logic syntax and adopt required tools from Ch. S. Pierce's existential graphs system. In addition, we propose to use schemes of expression for prompting to a user, optimizing data structure and explicating an ontology structure. We illustrate the developed method, named logic graphs, on a simple demo-ontology.

Keywords: Visualization · Description Logic · Existential graphs.

1 Introduction

The purpose of our research is to develop a complete and convenient method for logic expressions visualization. Contemporary tools of graph data visualization do not allow to work with logic expressions completely, while necessity to work with them in visual format is obvious. Providing to a user an opportunity to observe ontologies constructed by axioms would be extremely useful. The developed prototype should be suitable for realization in the Ontodia library [1].

2 Tools overview

We present an overview of the existing ontology visualization tools.

VOWL. In the current realization of VOWL [2] it is impossible to represent a composition of two and more logic operators. For example, consider an axiom

$$Woman \equiv Person \sqcap \neg Man \tag{1}$$

which means "A woman is a person, who is not a man". Its visualization is given at the Fig. 1. In this axiom the negation of the concept Man is also a part of conjunction, while at the visualization only the negation is represented. Besides, VOWL language is not satisfactory, as logic relations are not visualized, but only symbolically depicted, this in substance does not differ from a formula.

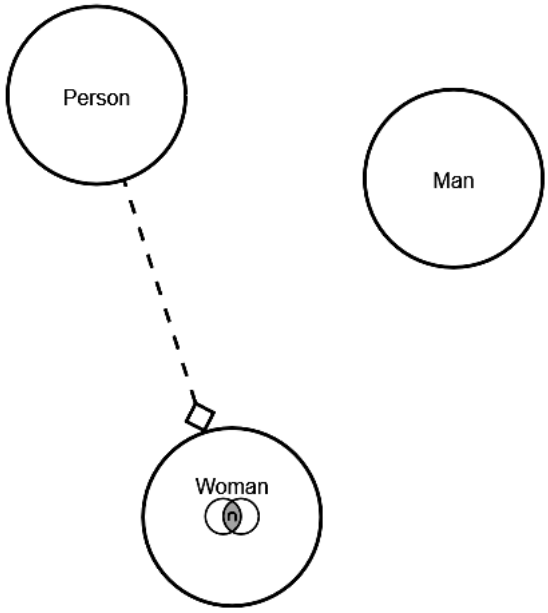

Fig. 1. "A woman is a person, who is not a man" in VOWL

Venn diagrams. First, Venn diagrams [3] do not allow to represent logic relations unambiguously. For example, a conjunction $c1 \sqcap c2$ can be represented by three different diagrams, that are given at the Fir. 2. Second, it is possible to

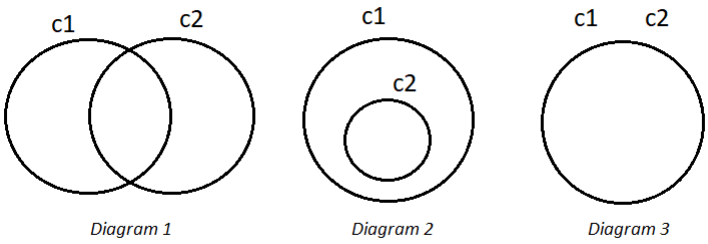

Fig. 2. $c1$ and $c2$ in Venn diagrams

visualize only concepts, but not roles with Venn diagrams.

Graphol. Graphol language [4] is not satisfactory as well, because, as in VOWL, logic relations are not interpreted. Besides, Graphol language is overcomplicated and hard to perceive. As an example, visualization of an axiom

$$CheezeyPizza \equiv Pizza \sqcap \exists hasTopping.CheezeyTopping \tag{2}$$

which means "CheezeyPizza is a pizza, which has cheezy topping", from the Pizza ontology [5] is given at the Fig 3.

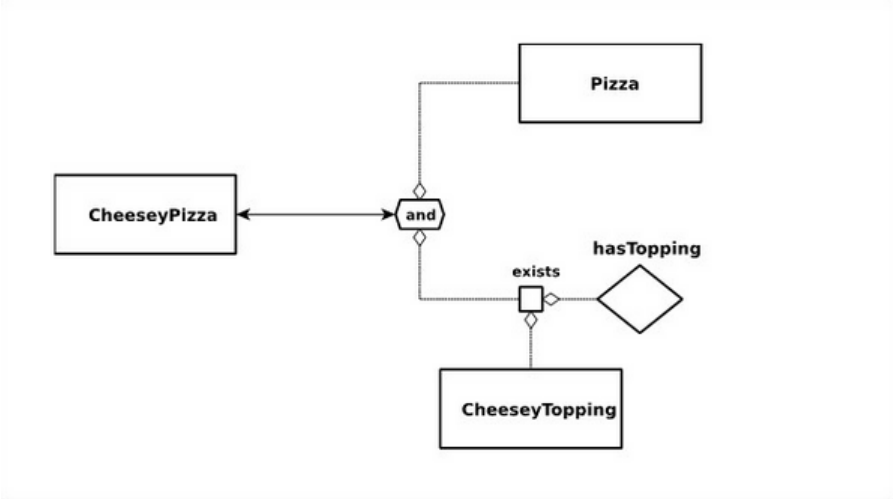

Fig. 3. "CheezeyPizza is a pizza, which has cheezy topping" in Graphol

Therefore, it is necessary to develop a complete, explicit, but simple visualization method for OWL language and description logic.

3 Logic Graphs

We consider visualization method completeness with respect to syntax of description logic, which is used to formulate ontology axioms. Basic description logic syntax [6] consists of notations given at the Table 1.
Ontodia already has visualization tools for concepts, roles, inclusion, equivalence and assertion. Developing our method, we intend to save all features presented in Ontodia and, therefore, we use all existing visualization tools without changes. In result, it is sufficient to find a way to visualize conjunction, disjunction, negation and universal and existential restrictions.

The proposed method is based on the existential graphs, suggested by Ch. S. Pierce [7]. In the existential graphs system sheets of assertion are considered. A sheet of assertion denotes a space of true expressions, therefore, expressions situated on the sheet are conjuncted. Besides, there are cuts, that means inversion

Table 1. Basic description logic syntax

Symbol	Description	Example	Reading
C	Concept	$c1$	Concept $c1$
R	Role	$op1$	Role $op1$
$\sqcap$	Conjunction	$c1 \sqcap c2$	$c1$ and $c2$
$\sqcup$	Disjunction	$c1 \sqcup c2$	$c1$ or $c2$
$\neg$	Negation	$\neg c1$	Not $c1$
$\forall$	Universal restriction	$\forall op1.c1$	All $op1$-successors are in $c1$
$\exists$	Existential restriction	$\exists op1.c1$	An $op1$-successors exists in $c1$
$\sqsubseteq$	Inclusion	$c1 \sqsubseteq c2$	All $c1$ are $c2$
$\equiv$	Equivalence	$c1 \equiv c2$	$c1$ is equivalent to $c2$
:	Assertion	$a : c1$, $(a, b) : op1$	a is a $c1$, a is $op1$-related to b

of the expression space truth-value, therefore, if an expression is situated in a cut, it is negated. Finally, there are curves to denote relations. Visualization of description logic expressions with existential graphs is illustrated in the Fig. 4.

Though existential graphs are sufficient for visualizing description logic, practically it is not convenient as existential graphs for complex expression are overcomplicated and hardly interpretable. It can be seen on the graph 6 already. Therefore, we propose to develop a new system, which inherit current visualization tools of Ontodia and extends it with required elements of the existential graphs. The resulting system, named logic graphs, adjusted to Ontodia design, is given at the Table 5.

Summing up, the proposal extends visualization tools of Ontodia with only two elements: filling for negation and embedded concepts for conjunction. All roles are considered as existentially restricted by default. Other required operations are expressed by means of these two.

Logic graphs are applicable for visualizing other logical systems. The original existential graphs system, described in the Fig. 4, consists of *Alpha* and *Beta* subsystems. The existential graphs 1-6 relate to *Alpha* subsystem, which is isomorphic to propositional logic. As logic graphs system inherit existential graphs 1-6 of *Alpha* as graphs 1-3 and its derivatives, logic graphs are applicable for visualizing propositional logic as well. However, they are more convenient as logic graphs inherit simpler inclusion and equivalence representation from Ontodia.

Schemes. For interactive constructing of ontologies from axioms we propose to use axiom schemes – logic structure of a formula regardless of situated concepts. Axiom schemes serve to several purposes:

- Prompting to user, in which node of an ontology he should place a selected concept. Elements of a schema are not clickable, after constructing is finished, all additional elements are hidden.
- Decreasing the amount of data displayed at the data panel. One schema could be associated with different axioms. In this case it is sufficient to display at

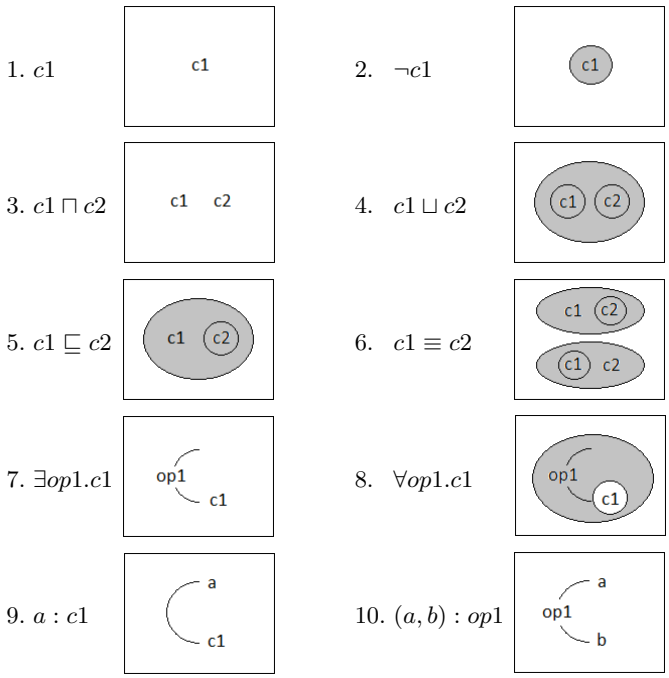

Fig. 4. Existential graphs

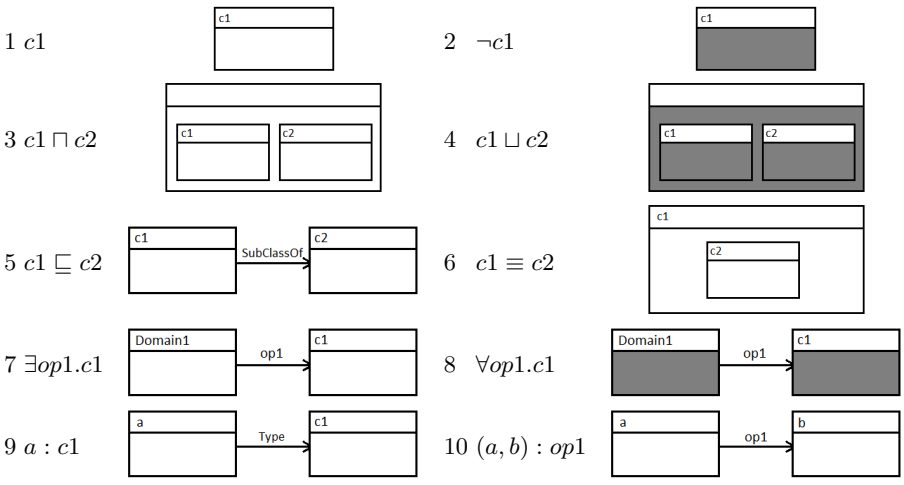

Fig. 5. Logic graphs

the data panel only scheme and different axioms would be formed in result of filling it with concepts.
– Explication of ontology structure and the process of its construction. As user would have to construct an ontology by filling schemes, during this process he would get an understanding of what is structural elements of an ontology and how they are connected.

Concepts on the canvas has two states: "out of context" and "in context". A concept is out of context, if it is out of a scheme, and it is "in context" otherwise. Concepts, which are out of context, can have only one instance at the canvas, and multiple instances otherwise.

In addition, we propose two interactivity modes, that differ in organization of the data panel. In the Mode 1 concepts from expressions that have identical structure are grouped by its schemes. This approach allows to compress data at the data panel, but still allows duplication. In the Mode 2 all concepts are listed in the separate list. In this case concept duplication is excluded.

4 Examples

To illustrate the method we made a list of axioms:

1. $SPbWoman \equiv Person \sqcap \neg Man \sqcap \exists LiveIn.SPb$ ("Saint-Petersburg woman is a person, who is not a man and lives in Saint-Petersburg");
2. $MskWoman \equiv Person \sqcap \neg Man \sqcap \exists LivsIn.Msk$ ("Moscow woman is a person, who is not a man and lives in Moscow").
3. $Person \equiv Man \sqcup Woman$ ("Person is a man or a woman").
4. $Woman \sqsubseteq \neg Man$ ("All women are not men").
5. $NativeSPbWoman2ndGen \sqsubseteq \forall HasParent.BornInSPb$ ("All native Saint-Petersburg women in second generation has only those parents, who were born in Saint-Petersburg").
6. $ParentOfSPbPerson \equiv \exists HasChild.\exists LiveIn.SPb$ ("Parent of an Saint-Petersburg person is a person, who has a child, who lives in Saint-Petersburg").

Their visualization is schematically represented below.

Modes. First, consider difference between the Modes 1 and 2. Expressions about $SPbWoman$ and $MskWoman$ have identical structure, therefore, in the Mode 1 concepts from these expressions are grouped into one schema. Other expressions have different schemes and there is a concept group for each axiom in the class tree. This approach allows to reduce concept duplication in the tree. For example, concepts Man and $Person$ from the two axioms with identical structure (about $SPbWoman$ and $MskWoman$) are not duplicated. It is notable, that it is one and the same concept Man in two different groups and it has one and the same IRI.

In the Mode 2 the user has two lists in the concept tree: a list of unique concepts and a list of schemes. The difference is illustrated at the Fig. 6. Further only Mode 1 is considered.

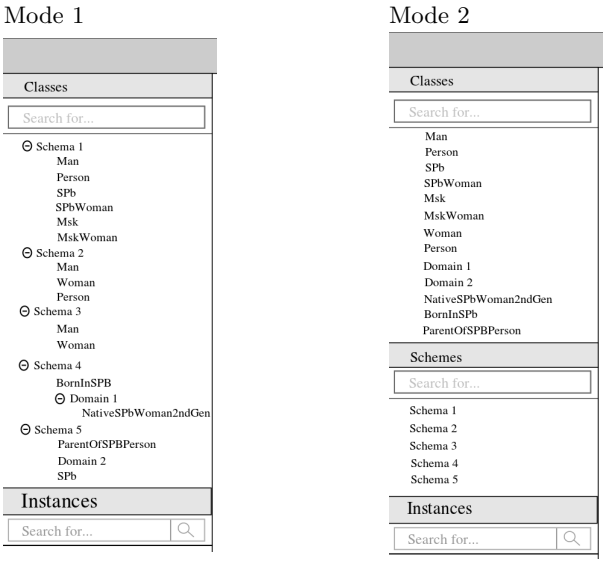

Fig. 6. Modes

Step 1. After data initilized, a user has a list of accessible schemes in the class tree. The concepts *Domain* 1 and 2 are auxiliary and appears after axioms being parsed. They are necessary as domains of the corresponding roles are not specified in the formula. At this step all elements are clickable, it is possible to drag them on the canvas. See Fig. 7.

Step 2. The user can drag all concepts from the tree. At this step all of them are out of context. After the user dragged *Schema* 1 on the canvas, the structure of the schema appears. Elements of a schema are not clickable, they prompt to the user, which concepts it is able to drag in the context. As the schema embraces *Axiom* 1 and 2, all their concepts are specified. It is notable that though the concept Man is negated in the *Schema* 1, it is not filled being out of context, but it is into the schema, where the concept is negated. See Fig. 8.

Step 3. The user fills the *Schema* 1 with accessible concepts in arbitrary order. As soon as required domain and range for the role $LiveIn$ are added, the connection automatically appears. After the user dragged all required concepts into the schema, it transforms into a particular axiom, depending on the concepts situated, schema borders disappear. In this case, the *Schema* 1 resulted in the *Axiom* 2. After that, the user can drag the concept Man on the canvas out of context from another schema. See Fig. 9.

Step 4. All schemes are illustrated. See Fig. 10.

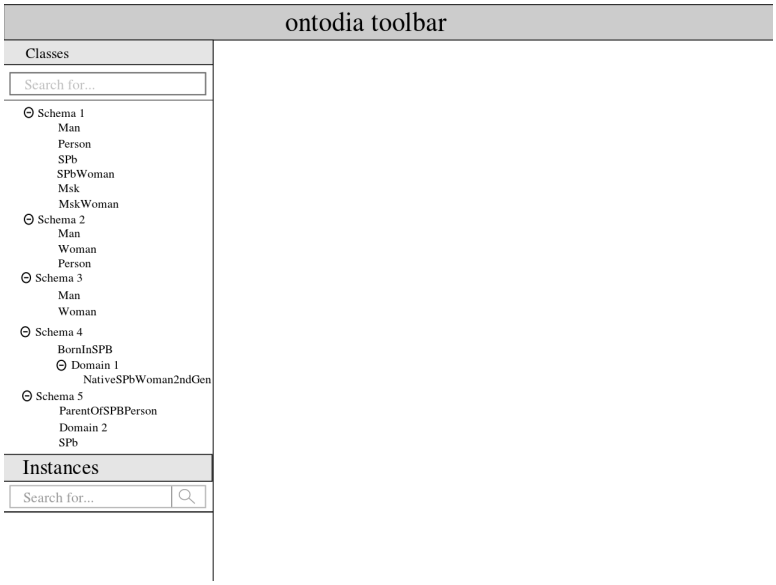

Fig. 7. Step 1

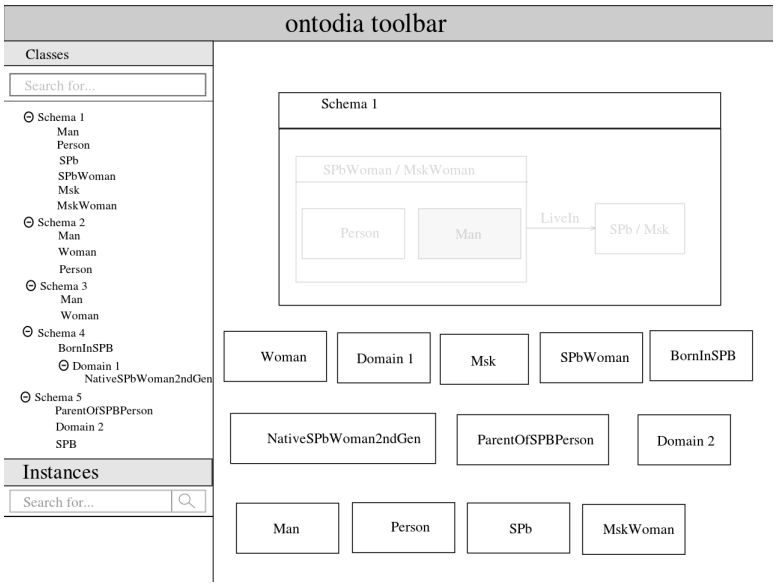

Fig. 8. Step 2

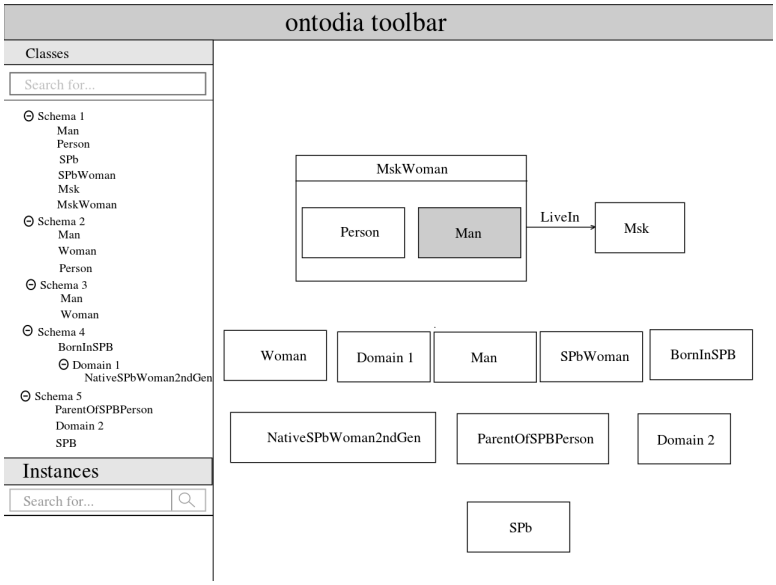

Fig. 9. Step 3

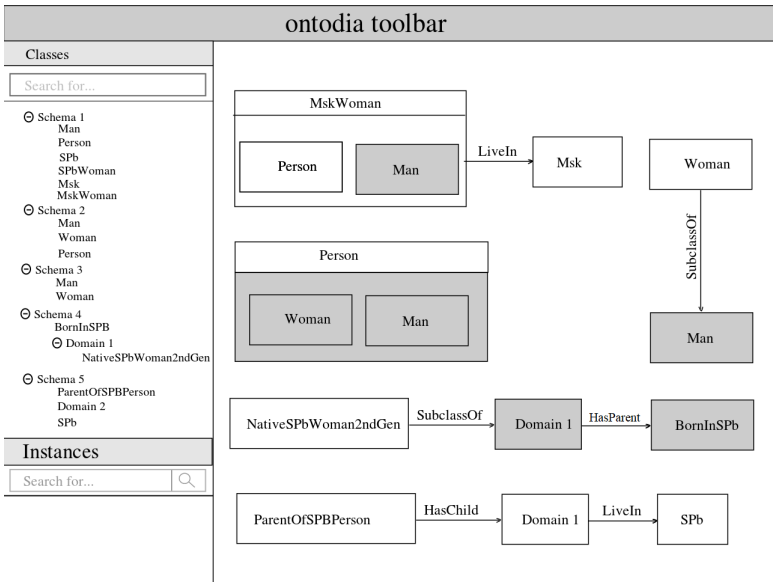

Fig. 10. Step 4

5 Conclusion

Summing up, the developed method allows to visualize semantics of logical expressions completely, unambiguously and in a simple way. It was illustratd on a demo ontology. The future work will consider the possibility of visualizing expressions of different logic languages.

References

1. Ontodia Homepage, http://ontodia.org/. Last accessed 29 Jan 2019
2. VOWL Homepage, http://vowl.visualdataweb.org/. Last accessed 29 Jan 2019
3. Venn J.: I. On the diagrammatic and mechanical representation of propositions and reasonings. The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science **10**(59), 1–18 (1880)
4. Graphol Homepage, http://www.obdasystems.com/graphol. Last accessed 13 Mar 2019
5. Pizza Ontology, https://protege.stanford.edu/ontologies/pizza/pizza.owl. Last accessed 29 Jan 2019
6. Baader F., Calvanese D., McGuinness D., Nardi D., Patel-Schneider P. (eds.): The Description Logic Handbook: Theory, Implementation, Applications. Cambridge University Press, Cambridge, UK (2003)
7. Peirce Ch.: Manuscript 514. 1909. Transcribed by Michel Balat with commentary by J. F. Sowa. http://www.jfsowa.com/peirce/ms514.htm. Last accessed 29 Jan 2019