Graph-based Editor for SWRL Rule Bases

Jaroslaw Bak, Maciej Nowak, Czeslaw Jedrzejek

Institute of Control and Information Engineering,
Poznan University of Technology,
M. Sklodowskiej-Curie Sqr. 5, 60-965 Poznan, Poland
{firstname.lastname}@put.poznan.pl

Abstract. In this paper we present a prototypical implementation of a graphical tool for creating and editing (DL-safe) SWRL rules. The tool uses a graph-based approach to model rules expressed in the SWRL language. Rules are built from concepts and roles defined in an OWL ontology. Such a knowledge base can be visualised and edited in a user-friendly interface. Moreover, the presented tool provides methods for graphical representation of data and results of reasoning performed with the Pellet engine. We present a process of creating a knowledge base of family relationships as an example case. Perspectives of our future work are also presented.

Keywords: graphical rule representation, SWRL, ontology visualisation, reasoning

1 Introduction

The Semantic Web[1], which is the extension of the World Wide Web, is still in active research and development. However, emerging technologies provide methods and standards for processing data according to the defined semantics. The semantics of data can be expressed by ontologies and rules which are of a special significance in the layered architecture of the Semantic Web. An ontology and a set of rules constitute a knowledge base of some particular domain. Using the knowledge base and data with an appropriate reasoner, we can perform reasoning tasks. Thus, additional knowledge can be inferred.

An ontology can be expressed using one of the OWL family of languages (OWL 1.1[2] and OWL 2 Profiles[3]), whereas a rule can be written in the Semantic Web Rule Language (SWRL) [1] or OWL 2 RL Profile[4].

Despite the clear advantages and availability of semantics-based technologies, there are many software application areas where they do not occur or occur in a rela-

[1] http://www.w3.org/standards/semanticweb/
[2] http://www.w3.org/Submission/owl11-overview/
[3] http://www.w3.org/TR/owl2-profiles/
[4] Other appropriate languages also exist (e.g. RDF(S) for simple ontologies, RuleML for rules etc.) but currently we do not consider them in our work.

tively simple form (e.g. ontologies as vocabularies, rules as filters). The main reason for this is because ontologies and rules are too complex to handle by an ordinary user [2]. The process of acquiring ontology- and rule-based knowledge can be simplified with the use of a graphical representation and a user-friendly interface.

Since SWRL provides more powerful reasoning capabilities than OWL and some of the ontologies can be transformed into rules (e.g. Horn-SHIQ [3]) we focus on the development of a graph-based environment which will provide an easy way of creating and managing SWRL rule bases.

The main goal of this paper is to present a graph-based tool, in which an untrained user is able to construct a set of simple (DL-safe [4]) SWRL rules and to use them in order to obtain new (inferred) information according to the semantics defined in an OWL ontology. Both rules and the ontology constitute a knowledge base of a given domain. The ontology provides necessary concepts and roles, whereas the rules constitute additional knowledge mixing concepts and roles in a way which is not allowed in OWL. Additionally, a set of facts represents data. The constructed knowledge base and facts are expressed graphically in the form of directed graphs. The knowledge base can be applied to facts using a reasoning engine. After the inference process, a user gets the result, which is also represented graphically.

The paper is organized as follows. Section 2 presents the main overview of the proposed approach, a set of employed tools and related work. Section 3 describes a prototypical implementation with a demonstration of creating a simple knowledge base of family relationships. Section 4 contains concluding remarks and our future work.

2 Graph-based Representation of an Ontology and Rules

2.1 Existing Methods and Tools

Visualising data in the form of graphs is connected to a problem of knowledge representation (KR). Many investigators have created standardized notations for KR (Unified Modeling Language/Object Constraint Language (UML/OCL) [5], UML-based Rule Modeling Language (URML) [6], Object Role Modeling (ORM) [7], or SBVR[5] to name a few), however, so far many commercial tools tend to use their own standards. Other popular KR methods include: decision tables, decision trees and eXtended Tabular Trees [8]. Most commercial applications use those representations directly, or in a form of guided textual editors. In our approach, we aim to provide similar ways of representing both knowledge bases and rules. That is why the ORM approach, combined with a graph-based representation, seems to be sufficient to start with.

[5] Semantics of Business Vocabulary and Business Rules, http://www.omg.org/spec/SBVR/1.0/

There are a number of tools implementing graphical rules representations:

- Visual Rules[6] – it allows building of flow rules, decision tables and decision trees. It is focused on business logic and directs the flow of decision making by a defined life cycle. Events causing state changes are controlled by the rules. Both states and rules are converted and executed as Java code. Visual Rules lacks the ontology background, and focuses solely on business rules and decision flows.
- Drools Guvnor[7] – it provides many guided ways of creating rules: decision tables, rule flow and a single rule editor. It is a data repository for the Drools system. Guvnor offers many useful features: versioning and packaging of rules, models, functions and processes connected to knowledge bases and supervision of access to rule bases. We considered Drools as our reasoning module, but the Pellet reasoner is sufficient for the needs of SWRL rules.
- VisiRule[8] – it is an extension to Win-Prolog and it only allows creating decision flow models using a graphical paradigm. It offers a graphical representation of forward chaining rules with access to Prolog. VisiRule offers collaboration features; diagrams expressed in it may consist of nested parts. It is another platform designed for business flows rather than deductive rules.
- OntoStudio Graphical Rule Editor[9] – it is based on Object Logic [9], and operates on OL and SPARQL[10] queries. Diagrams here consist of concepts, their attributes and connections between them. It handles many known ontology formats (OWL, RDF, SPARQL, RIF) as well as UML 2.0. OntoStudio allows testing and debugging of rules. It does not allow the comparison of variables (comparisons between value and variable are allowed only). This approach is similar to ours, except that our tool visualises both ontology and reasoning results on a graph.
- CoGui[11] – it is a visualization tool for creating knowledge bases and conceptual graphs. It is based on the conceptual graph model introduced in [10]. The knowledge base of CoGui consists of hierarchies of concepts and relations, a set of individuals and a set of conceptual rules. It uses the CoGitant engine for inference tasks. The structure of graphs can be nested; relations are not restricted to unary or binary relations (n-ary relations are allowed). This tool does not support the OWL ontology format, nor does it operate on standardized rule notations.

[6] http://www.bosch-si.com/technology/business-rules-management-brm/visual-rules-suite.html
[7] http://www.jboss.org/drools/drools-guvnor.html
[8] http://www.lpa.co.uk/vsr.htm
[9] http://www.semafora-systems.com/en/products/ontostudio/
[10] http://www.w3.org/TR/rdf-sparql-query/
[11] http://www2.lirmm.fr/cogui/

- Protégé OWLViz[12] plugin – it creates a hierarchical view of the selected part of an ontology in the form of a directed graph. It does not allow manipulation of objects on the graph nor does it visualise SWRL rules.
- Protégé Axiomé[13] [11, 12] plugin – it supports visual rule base management, exploration, automated rule categorization, rule paraphrasing and rule elicitation functionality. It does not provide a way to create SWRL rules; instead it is designed to help users understand the meaning of rules. Axiomé can represent rules as a graph where each rule is represented as a node and direct edges between nodes indicate that SWRL atoms are shared by the rules.
- TopBraid Composer[14] – it is a visual modelling tool designed to create and manage ontologies in the Semantic Web standards. It is based on the Eclipse[15] platform and the Jena API[16]. TopBraid Composer offers drag-and-drop way of creating and editing OWL ontologies. It allows consistency checking and debugging of OWL Inference engine. Users are able to incorporate SPARQL rules (SPIN[17]) into the process of class definition to create some constraints.
- Snoggle[18] – a graphical SWRL-based ontology mapper. It creates directed graphs representing structures of source and destination ontologies and enables creation of mapping relations between concepts from both ontologies. Those mapping relations are then converted into SWRL rules.

2.2 Overview of the Approach

The main goal of this paper is to present a graph-based environment, in which a user can: load an ontology, create and edit SWRL rules, perform reasoning and obtain results. Moreover, an ontology, rules and data are represented graphically as directed graphs. Additionally, an ontology can be represented as simple (and calculated) taxonomies of concepts and both types of roles (datatype and object properties). As a result, we obtain a graphical representation of a knowledge base constructed from concepts, roles, rules and facts (data). The knowledge base can be easily understood by an ordinary user who tries to work with ontologies and rules. Our aim is to provide an easy-to-use and easy-to-understand tool which can be used in many domains where ontologies, rules and graphs can be employed to support a user's work.

The process of rule creation consists of creating two graphs which represents two parts of a rule: the body (left hand side) and the head (right hand side). In the presented approach, rules are of the following form: *if the body then the head*. Both the body

[12] http://protegewiki.stanford.edu/wiki/OWLViz
[13] http://protegewiki.stanford.edu/wiki/Axiomé
[14] http://www.topquadrant.com/products/TB_Composer.html
[15] http://www.eclipse.org/
[16] http://jena.apache.org/
[17] http://spinrdf.org/
[18] http://snoggle.semwebcentral.org/

and the head consist of positive conjunctions of atoms that are defined in an ontology as classes (concepts), object properties (roles) and datatypes. Thus, the left hand side (LHS) of a rule should be perceived as conditional elements that need to be fulfilled in order to execute instructions written in the right hand side (RHS). The execution of a rule can add new statements to the given knowledge base in the form of new relations between objects and new classifications of them. For example, using rule (1) we can infer that a person which has a male child has a son.

$$Man(?y),\ Person(?x),\ hasChild(?x,\ ?y) \rightarrow hasSon(?x,\ ?y),\ Son(?y) \quad (1)$$

In rule (1) Man, $Person$ and Son are OWL classes, $hasChild$ and $hasSon$ are object properties and $?x,\ ?y$ are variables. By executing this rule we obtain a new relation between objects under both variables from rule (1) and a new classification of object under variable $?y$.

As mentioned before, we represent rules, an ontology and facts in a graphical form. Each of them is a different directed graph. Each graph consists of nodes and edges. The nodes are a graphical representation of OWL classes (or objects in data visualisation) whereas edges represent appropriate relations between classes (objects); or classes (objects) and datatypes. Usually, an object may belong to a number of OWL classes, for example an object of class Son belongs also to the following classes: Man and $Person$. In our method we decided to use the most detailed class, which is often represented as the most bottom concept in the taxonomy of OWL classes. The rest of the applicable classes are shown in a tooltip after moving the mouse above the object. Moreover, a user can choose which class she/he wants to see on a graph. The same approach is applied in the object and datatype property taxonomies. An example of choosing a visible class is presented in Figure 1.

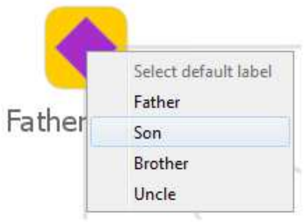


Figure 1. Selecting a visible class is done by clicking on a class name from a popup menu.

When loading an ontology, we can obtain two kinds of visualization. The first one is a Protégé-like view of taxonomies as trees. We provide three trees: the taxonomy of classes, the taxonomy of object properties and the taxonomy of datatype properties. The second visualization type is a graph-based view in which taxonomies are represented as directed graphs. Since a (rooted) tree is a special kind of directed graph, the visualization in both types is very similar. The main difference between them is that, in the graph mode, we can manipulate the graph structure by using specialised layouts or by manual rearrangement. Both types of OWL classes visualisation are presented in Figure 2.

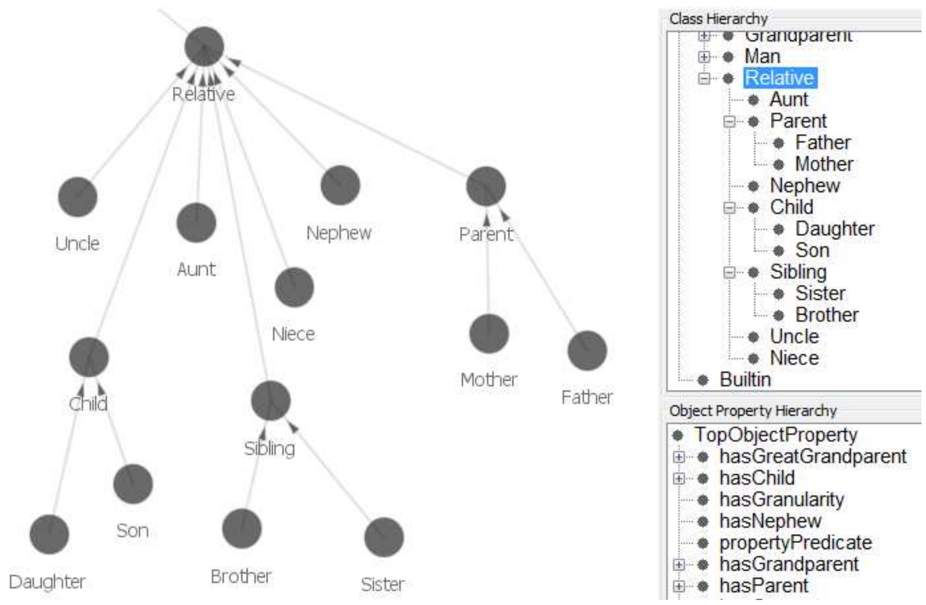


Figure 2. Visualisation methods of OWL classes.

Our graph-based editor supports the reasoning performed by the Pellet engine (see Section 2.3). Results are presented to a user as a new graph of objects or as a pair of graphs representing calculated taxonomies of classes and object properties. Moreover, a user can check the consistency of an ontology and verify results obtained from ontology- and rule-based reasoning.

2.3 Applied Tools

In the presented graph-based editor we apply the Semantic Web Rule Language with its syntax and semantics to read/write rules from/to an ontology. We employ the OWL Web Ontology Language version 1.1 as a way to express the semantics of a given domain. SWRL Built-ins [13] are used as comparison predicates between values of datatype properties or variables. Since we adapt SWRL as an OWL-based rule language we follow its semantics. As a result, negated atoms or disjunctions are not allowed. Moreover, we apply the *DL-safe rules* [4] approach, which considers decidable combinations of OWL DL and rule axioms. Decidability is preserved by forcing each rule to be DL-safe, which means that each variable is bound only to the individuals that explicitly occur in the assertional part (data) of the knowledge base. In other words, only facts that are explicitly stated can be used in the reasoning process.

We employed the OWL API[19] tool to parse and write OWL ontologies. The Pellet[20] engine is used as an OWL and SWRL reasoner. As a result of reasoning we can: check the consistency of an ontology and rules, calculate taxonomies, obtain potential

[19] http://owlapi.sourceforge.net/
[20] http://clarkparsia.com/pellet/

inconsistences and infer new facts. In the editor, we can visualise an ontology before and after the calculation of taxonomies. Additionally, we can obtain a graph of facts before and after the reasoning process.

Visualization uses two Java libraries: Gephi[21] and Processing[22]. We use Gephi to manipulate graph structures. It is also responsible for managing the layout of the nodes on the graph. Nodes can be rearranged manually or placed according to their graph-based parameters (centrality, modularity, PageRank, etc.). We use Processing as a software sketchbook to create the views of an ontology structure and a set of facts, as well as to create the rule editor.

Our graph-based rule editor for SWRL rule bases is fully implemented in the Java language.

3 Graph-based Editor

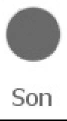
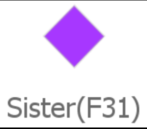
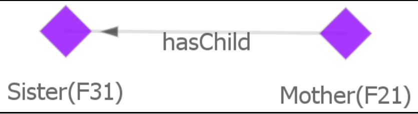
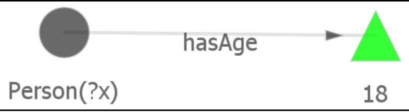
3.1 Rule Creation and Edition Method

Our graph-based editor consists of three tabs: ontology view, rule creation graph and instances view. The ontology view tab shows a visualisation of an ontology. The user can select a hierarchical structure of classes, datatypes or object properties to be visualised. Every edge in this view represents a *subClassOf* or *subPropertyOf* relation from the ontology. Our system proposes a calculated layout of classes, however this graph can be manually rearranged in order to improve the user impression and understanding. The hierarchy of classes is represented as grey circle nodes connected with edges. Structure of object properties is represented as blue square nodes connected with edges, whereas data properties are represented as green triangle nodes also connected with edges. All edges in the ontology view tab reflect the subsumption relation between two nodes.

The rule creation tab consists of 3 parts: conditions side, which represents the body of a rule; conclusions side, which reflects the head; and the class hierarchy presented in a tree structure. In order to create a rule, a user drags a class from the Class Hierarchy tree and drops it onto one of the rule sides (conditions or conclusions side). She/he is asked for a variable name or a value, which indicates the added object. Class concepts are presented as circles on the graph, with their class name and variable (value) as their labels. Both datatype properties and comparisons of variables can be added by right clicking on an object on the graph and selecting an appropriate option. The system limits datatype properties to those which can be linked with the selected object type (the selected class is in the domain of that datatype property or the domain constraint does not occur). Datatype properties are displayed in the form of a triangle connected by an edge with the corresponding object. The name of datatype is shown on the edge, and its value as a label of the triangle.

[21] https://gephi.org/
[22] http://processing.org/

Table 1. Representations of main elements in our Graph-based Editor for SWRL rules.

Element	**Graph-based representation**
OWL Class	
OWL Class instance	
Object property between two OWL instances	
Datatype property between an object and a value	

Relation between objects (object properties) can be added in a similar manner. After selecting a node and right clicking the other node, a list of available object properties is presented. After selecting one of them, it is displayed as an edge between selected nodes. In order to save a rule, the user needs to select an option from the top menu (*File*, then *Save as…*). A user can choose to save the ontology combined with the created rules.

The instances tab visualizes individuals (facts) stored in a knowledge base. They are represented as purple rhombs connected with each other by edges (roles from the ontology). Individuals can have datatype properties, which are visualized in the same way as in the rule creation panel, by triangles. After the reasoning, objects can belong to many OWL classes. This fact is impossible to represent on a static graph, however we present a method to solve this problem. After moving the mouse above an individual, a tooltip with all inferred classes appears. User can select which class should be visible on the graph as a default one.

The graphical representation of particular elements, which is applied in our editor, is presented in Table 1.

3.2 An Example Case

An example application of our tool is performed with an ontology describing family relationships. We slightly modified an ontology developed by Christine Golbreich presented in [14]. Her ontology is publicly available[23]. It contains the usual classes, e.g. $Person$, Man, $Woman$, $Child$, $Parent$, etc., and relationships within a family, e.g. $hasConsort$, $hasChild$, $hasParent$, etc.

[23] http://protege.cim3.net/file/pub/ontologies/family.swrl.owl/family.swrl.owl

The main difference between the original family ontology and our version of it, is the addition of:

- Classes: $Grandparent$, $Grandfather$, $Grandmother$, $GreatGrandparent$, $GreatGrandfather$, $GreatGrandmother$.
- Object properties: $hasCousin$, $hasGrandparent$, $hasGrandfather$, $hasGrandmother$, $hasGreatGrandparent$, $hasGreatGrandfather$, $hasGreatGrandmother$.
- Datatype property: $hasAge$.

Since the aforementioned new elements of the ontology are self-explanatory we do not provide more detailed descriptions. The modified version of the family ontology was then loaded into our editor. In the tool we created rules which are responsible to obtain instances of the following:

- Classes: $Grandfather$, $Grandmother$, $GreatGrandfather$, $GreatGrandmother$.
- Object properties: $hasCousin$, $hasGrandfather$, $hasGrandmother$, $hasGreatGrandfather$, $hasGreatGrandmother$.

In this paper, we present two rules created with our editor. Rule (2) asserts an instance of relation $hasCousin$ which reflects that children of siblings are cousins of each other ($hasCousin$ is defined as a symmetric property). Rule (2) is presented in Figure 3.

$$\begin{aligned}&Person(?x),\ \ Person(?y),\ \ Person(?w),\ \ Person(?z),\\ &hasParent(?w,?z),\ \ hasParent(?x,?y),\ \ hasSibling(?y,?z)\\ &\rightarrow hasCousin\ (?x,?w)\end{aligned} \tag{2}$$

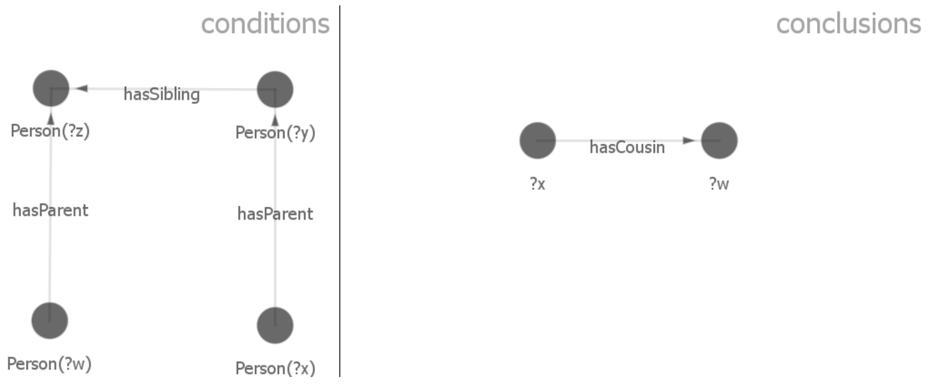


Figure 3. Creation of rule (2).

Rule (3) asserts an instance of class $GreatGrandfather$ and an instance of a role $hasGreatGrandfather$. The rule expresses that a father of our grandparent is our great grandfather. Rule (3) is presented in Figure 4.

$$\begin{aligned}&Person(?x),\ \ Person(?y),\ \ Person(?w),\\ &hasGrandparent(?x,?y),\ \ hasFather(?y,?w)\\ &\rightarrow hasGreatGrandfather\ (?x,?w),\ \ GreatGrandfather(?w)\end{aligned} \tag{3}$$

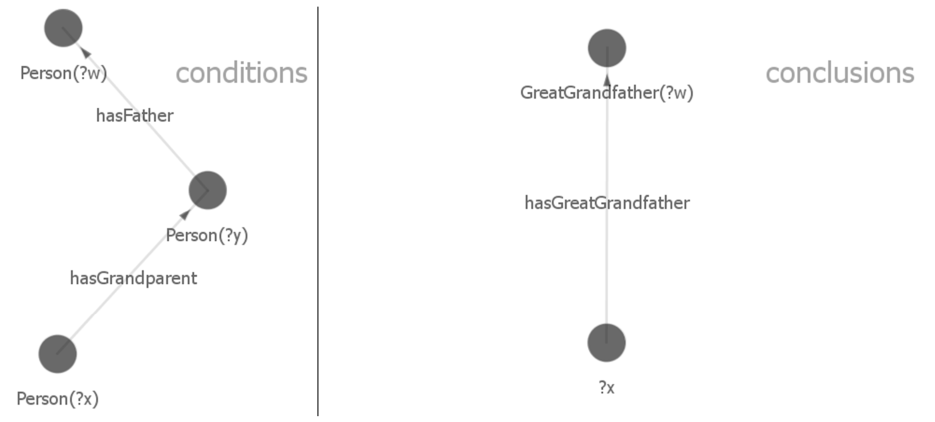


Figure 4. Creation of rule (3).

Created rules need to be applied to the set of facts in the ontology. After the reasoning process, executed by Pellet, a user obtains results presented in a new graph (in contrast to the graph before execution). Thus, new relations between objects and the classification of them are obtained. Figures 5 and 6 present two graphs: before reasoning (Figure 5) and after reasoning (Figure 6). These figures represent a part of the knowledge base to which rules (2) and (3) can be applied. Instances preceded by the letter 'M' represent men and instances preceded by 'F' represent women.

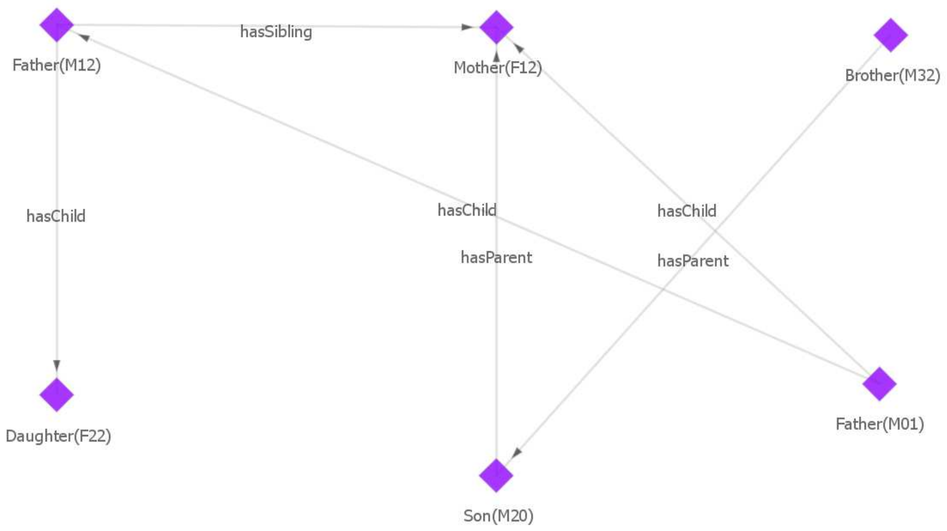


Figure 5. Graph of instances before reasoning.

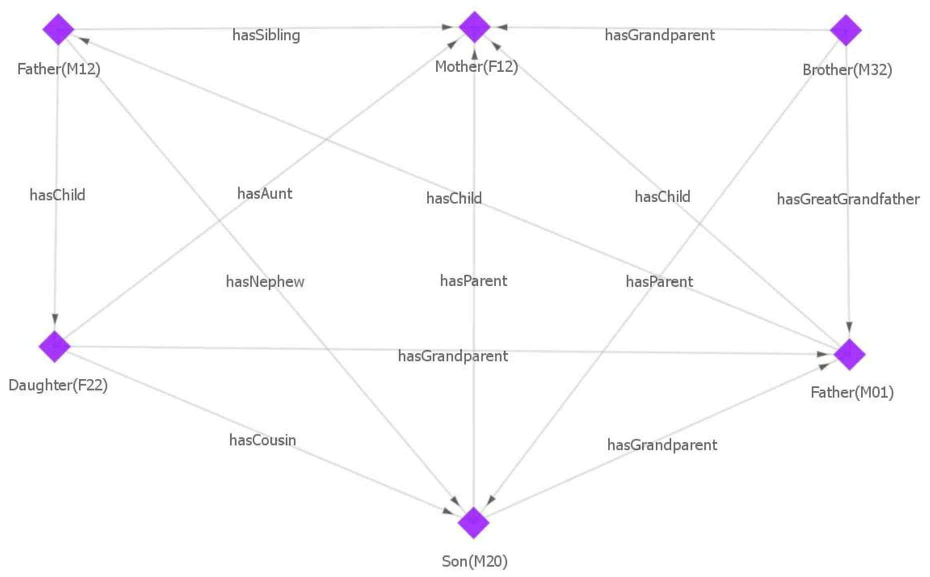


Figure 6. Graph of instances after reasoning.

4 Conclusions and Future Work

In this paper we have demonstrated a tool which supports graph-based creation and edition of SWRL rules. The tool provides a visualisation of an OWL ontology, SWRL rules and data. The graph-based representation is very convenient and intuitive. It is an initial implementation which supports the creation of SWRL rules in a graphical manner. The work presented in this paper is based on our previous experiences with graph-based representation of rules [15].

The developed graph-based editor can be used in many domains where ontologies, rules and graphs can be employed to support users in their work. Moreover, changes in a SWRL rule base can be made by business specialists without engaging an experienced programmer. As a result, the usual process of consultation between them is omitted or shortened in time. Thus, the tool can significantly increase their work's efficiency.

In further work, we will implement a query method for searching a knowledge base in a graphical manner. Moreover, we will provide a relational database interface. As a result, a semantic query will be posed into an integrated environment which will include a relational database, a set of rules and an ontology. In this case any graph containing nodes and edges could be entered as a search phrase. The reasoning engine will search the whole knowledge base for a given set of conditions, and return all objects that meet the specified requirements.

Another desired feature is to support OWL 2, which contains profiles designed for reasoning with rules and query answering, the RL and QL profiles respectively. A method of comparison between inferred and non-inferred knowledge bases is also planned.

Finally, we are going to make our tool available online for download and use with a free academic license (for non-commercial users) [16].

Acknowledgement. This work was supported by DS-MK 45-102/13 and 45-085/12 DS-PB grants.

References

1. Horrocks, I., Patel-Schneider, P.F., Boley, H., Tabet, S., Grosof, B., Dean, M.: Swrl: A semantic web rule language combining owl and ruleml. W3C Member Submission (May 21 2004), http://www.w3.org/Submission/SWRL/
2. Shotton D., Catton C., Klyne G., Ontologies for Sharing, Ontologies for Use, http://ontogenesis.knowledgeblog.org/312?kblog-transclude=2
3. Hustadt U., Motik B., Sattler U., *Data Complexity of Reasoning in Very Expressive Description Logics*, In IN PROC. IJCAI 2005, pages 466–471. Professional Book Center, 2005. (Cited on pages 5and 37.)
4. Motik B., Sattler U., Studer R., *Query Answering for OWL-DL with Rules*. In Journal of Web Semantics, pages 549–563. Springer, 2004.
5. Object Constraint Language (OCL), v2.0. http://www.omg.org/spec/OCL/2.0/
6. UML-based Rule Modelling Language, http://oxygen.informatik.tu-cottbus.de/rewerse-i1/?q=URML
7. Lukichev S., Jarrar M.: Graphical Notations for Rule Modeling. In: A. Giurca, D. Gasevic, and K. Taveter (Eds), Handbook of Research on Emerging Rule-based Languages and Technologies: Open Solutions and Approaches, IGI Publishing, 2009
8. Grzegorz J. Nalepa, Antoni Ligęza, and Krzysztof Kaczor. 2011. Overview of knowledge formalization with XTT2 rules. In Proceedings of the 5th international conference on Rule-based reasoning, programming, and applications (RuleML'2011), Nick Bassiliades, Guido Governatori, and Adrian Paschke (Eds.). Springer-Verlag, Berlin, Heidelberg, 329-336.
9. Michael Kifer, Georg Lausen, and James Wu. Logical foundations of object oriented and frame-based languages. J. ACM, 42(4):741–843, 1995
10. Sowa J. F., *Conceptual Structures: Information Processing in Mind and Machine*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1984.
11. Hassanpour S., O'Connor M. J., Das A. K., *A Rule Management and Elicitation Tool for SWRL Rule Bases*, 3rd International Rule Challenge at RuleML 2009, Las Vegas, NV.
12. Hassanpour S., O'Connor M. J., Das A. K., *Exploration of SWRL Rule Bases through Visualization, Paraphrasing, and Categorization of Rules*, International RuleML Symposium on Rule Interchange and Applications, Las Vegas, NV, 5858, 246-261, 2009.
13. SWRL Built-ins, http://www.w3.org/Submission/2004/SUBM-SWRL-20040521/
14. Golbreich C., "Combining rule and ontology reasoners for the semantic web.", Rules and Rule Markup Languages for the Semantic Web. Springer Berlin Heidelberg, 2004. 6-22.
15. Nowak M., Bak J., Jedrzejek C., *Graph-based Rule Editor*, in Hassan Aït-Kaci, Yuh-Jong Hu, Grzegorz J. Nalepa, Monica Palmirani and Dumitru Roman, editors, RuleML2012@ECAI Challenge and Doctoral Consortium at the 6th International Symposium on Rules, Montpellier, France, August 27th-29th, 2012, volume 874 of CEUR Workshop Proceedings. CEUR-WS.org, 2012.
16. Demo site: http://draco.kari.put.poznan.pl/ruleml2013_SWRLEditor/