Reasoning in Graph-based Clausal Form Logic

Alena Lukasova[1], Martin Zacek[2], Marek Vajgl[3]

**[1] Department of Informatics and Computers, University of Ostrava
Ostrava, Czech Republic**

**[2] Department of Informatics and Computers, University of Ostrava
Ostrava, Czech Republic**

**[3] Department of Informatics and Computers, University of Ostrava
Ostrava, Czech Republic**

Abstract

This paper follows the work of T. Richards specialization Clausal Form Logic formal system of the first order logic. The paper presents also the way of using graph-based clausal form statements in the frame of semantic (associative) networks. The goal of our research is to follow the direction towards graph-based clausal form knowledge representation shaped by Richards and build up a graph-based formal system. The new formal system Graph-based Clausal Form Logic has its own graph-based language with the expressivity similar to that one of CFL. The idea of the GCFL graph-based approach is also useful in the frame of the RDF model especially in its graph version.

Keywords: *Clausal Form Logic, formal system, graph, Graph-based Clausal Form Logic, RDF model.*

1. Introduction

T. Richards in his CFL (Clausal Form Logic) formal system of the first order logic (FOL) follows the idea of the "Horn´s clauses" and generalizes it to a concept of "conditional clauses", especially useful in declarative programming languages like PROLOG.

He defines CFL as a formal system with a special language with model-theoretic semantics and introduces a CFL modification of resolution inference rule as a tool of a formal reasoning.

He also shows the way of using graph-based clausal form statements in the frame of semantic (associative) networks. Richards [1] uses the graph-based representation especially for the illustration of meaning of clausal form expressions of CFL.

The goal of our approach is to follow the direction towards graph-based clausal form knowledge representation shaped by Richards, and build up a graph-based formal system that does not only graphically illustrate knowledge bases, but also allows users to obtain consequents of a knowledge base in a graph-based way.

The new formal system GCFL (Graph-based Clausal Form Logic) has its own graph-based language with the expressivity similar to that one of the CFL; moreover it uses the inference methods of associative networks [2] and proposes a graph-based modification of the resolution inference method of reasoning corresponding to that one of the CFL.

The idea of the GCFL graph-based approach is also useful in the frame of the RDF model especially within its graph version. To create RDF(S) knowledge base using the GCFL language completed by corresponding URIs is not difficult comparatively with an approach of OWL language representation. Finally the GCFL serves an own easy-to-understandable and usable inference mechanism.

2. Richard's clausal form logic (CFL) and its graph-based modification (GCFL)

As one of the main tools of formal reasoning, the CFL introduced by T. Richards [1] uses the conditional „if – then" statements.

A simple example illustrates the "if – then" statement structure by a "Holmes rule":

"*If one person x hates another person y, then x knows y.*"
or
If hate(x,y) then know(x,y)
or
If <antecedent> then <consequent>

Generally a conditional statement (clause) proposed by T. Richards says that some (composed) consequent statement follows from another (composed) antecedent statement.

Richards also proposed an alternative representation of the clausal form atoms (vectors) in a graph-based language well known in associative (semantic) networks (Fig. 1).

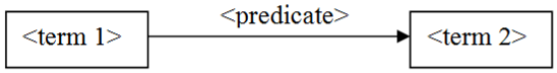


Fig. 1 The associative networks

Our idea of graph based representation of clauses connects to that one of Richards´ CFL and proposes a complete graph-based inference system GCFL. Knowledge base of the system consists of clauses represented by graphs.

To distinguish statements in the antecedent part of a graph-based clause from statements in its consequent part we introduce a convention

- to draw the arcs of antecedent vectors by dashed lines and
- to draw the arcs of consequent vectors by solid lines.

All the vectors (with solid or dashed lines) represent atomic statements and have generally the structure

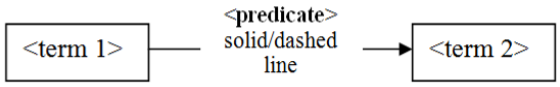


Fig. 2 Vectors

In the Richards´ clause language a general formula of CFL is of a form

$$\begin{gathered} \text{<antecedent>} \rightarrow \text{<consequent>} \\ \text{or} \\ P1\&\ldots\&\ Pm \rightarrow Q1\vee\ldots\vee Qn \\ \text{or} \\ P1,\cdots,\ Pm \rightarrow Q1,\cdots,\ Qn \end{gathered} \qquad (1)$$

where "$\rightarrow$" is a meta-symbol, the antecedent is a conjunction of some set of positive first order logic atoms (vectors) {P1,…, Pm} and the consequent is a disjunction of another set of positive first order logic atoms (vectors) {Q1,…, Qn}.

The convection of two mutually different arc lines in the GCFL does not need to separate antecedent and consequent by any meta-symbol like "$\rightarrow$", all the dashed antecedent vectors are mutually connected by & and all the solid consequent vectors are mutually connected by $\vee$.

In our Holmes example presented above we draw a vector with dashed-line arc as the antecedent part of graph representing a clause (Fig. 3)

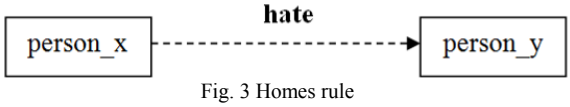


Fig. 3 Homes rule

and as the consequent part of the graph a vector with solid-line arc (Fig. 4)

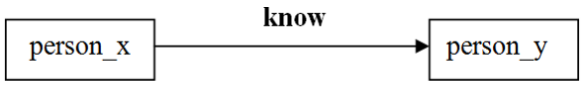


Fig. 4 Vector with solid-line arc

So the Holmes example above has in the graph language GCFL a form a clause network (Fig. 5)

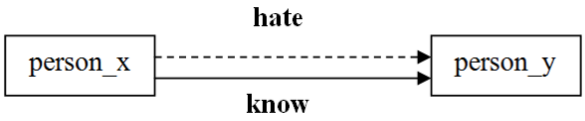


Fig. 5 Clause network

An antecedent or a consequent of the conditional clause in the GCFL can be also an empty set of atoms like in the same way as in the CFL.

In the case of an empty antecedent in the CFL for example the atom

$\rightarrow$ **know**(person_x, person_y)

has a meaning of a positive fact "A person_x knows a person_y.". To express is using the syntax of the graph language of GCFL a vector like that one at the Fig.4 suffices.

In the case of an empty consequent in the CFL the formal representation is of the form

know(person_x, person_y) $\rightarrow$

that represents a negative fact "It is not true that a person_x knows a person_y.".

To represent is using GCFL a vector with dashed line arc suffices (Fig.6).

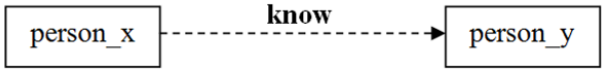


Fig. 6

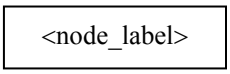
The structure of the clause allows only constructions of clauses with connections & in the antecedent and connection $\vee$ in the consequent. If necessary GCFL as well as CFL solves the problem of disjunction in the antecedent (conjunction in the consequent) by a following decomposition of the clause into m (n) separate clauses:

$$\begin{array}{ll} P_1 \rightarrow Q_1\vee\ldots\vee Q_n & P_1\&\ldots\&\ P_m \rightarrow Q_1 \\ \quad . & \quad . \\ \quad . & \quad . \\ P_m \rightarrow Q_1\vee\ldots\vee\ Q_n & P_1\&\ldots\&\ P_m \rightarrow Q_n \end{array}$$

3. Abstract syntax of the GCFL

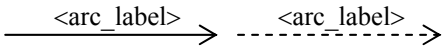
Definition (GCFL language symbols)
The language of the formal system GCFL uses the following symbols:

1) Nodes
   a) graphical symbols for nodes of networks



   b) terms in node labels are strings for representation of
      - variables – with capitals at first positions: X,Y, Man, Country,…
      - constants – with small letters or numbers at first positions: anne, cat, student,…
      - function symbols – sin(X), matka(X),...
      - existential terms – with @ as a prefix: @anybody, @known(X),...

2) Arcs
   a) graphical symbols for arcs of networks,



   b) binary predicate symbols as arc labels: isa(X,Y), citizen(X,Y),…

Definition (vector, ground/universal/existential vector)
Vector (atom) in the GCFL language (Fig. 2) consists of two nodes labelled by <term_1> and <term_2> symbols and their dashed/solid connecting arc labelled by a <predicate symbol>.
Vector with only constant labels of its nodes is a ground vector.
Vector with any variable symbol of a node label is a universal vector.
Vector with any existential symbol of a node label is an existential vector.

Definition (clause network, knowledge base)
Clause network in the GCFL language is a network of antecedent and/or consequent atomic vectors.
Knowledge base of GCFL is a set of GCFL clause networks.
Negative statement is in the GCFL expressed as a vector with a special symbol ⊗ (see Fig. 7). For example a statement "Anne does not know Jane." is capture in figure 7 as the vector:

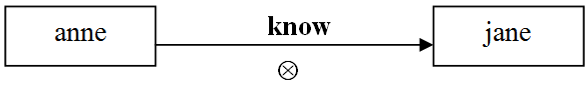


Fig. 7 Negative statement

4. Truth and consistency of a conditional clause in a given interpretation *I*

Truth value of the vector corresponds to the truth value of a corresponding predicate atom in the first order logic interpretation [1].
That means:
A ground vector is *true* iff there is a pair of elements within the relation ordered to the predicate in the *I* that is equal to the pair of ground vector constant terms, otherwise it is *false*.
For universal/existential vector holds that if all its variables are evaluated by constant terms and all of the values of the existential terms are found, then a decision of a true/false interpretation of the vector becomes the same as for the case of a ground vector.
A ground conditional clause is false if all the vectors of the antecedent are true and all the vectors of the consequent are false. Otherwise it is true.
A clause with an empty antecedent is evaluated as true; an empty consequent clause is represented as false.
A conditional clause is *consistent* in a given interpretation *I* if there is a valuation of all the variables that makes the clause true.

5. A special role of the predicate isa (ako)

Creating knowledge bases in graph-based formal systems brings (in comparison with the FOL) some new requests. For the sake of expression the clause in GCFL networks all the atoms of first order logic have to be transformed into corresponding binary versions (FOL as well as CFL does not use only binary predicates in its atoms, but also unary, ternary etc. predicates).
For the sake of creating knowledge base in the GCFL

- n-ary predicates ($n \geq 3$) have to be decomposed to binary predicates,
- unary predicates have to use some auxiliary predicates to become corresponding binary predicates with the same meaning.

Ordering of unary predicates into GCFL networks is solved by the usage of special binary predicate symbols **isa**(<term1>, <term2>)with the meaning "is a" or **ako**(<term1>, <term2>) with the meaning "a kind of".
In the case of our Holmes rule in the CFL language the solution can have a form of a clause

isa(X, person) &**isa**(Y, person) &**hate**(X,Y) → **know**(X,Y)

If the GCFL syntax is applied, then usage of the CFL principles applied on the Holmes rule cause that in the two-part graph (without symbol →) contains only dashed antecedent vectors and full consequent vectors:

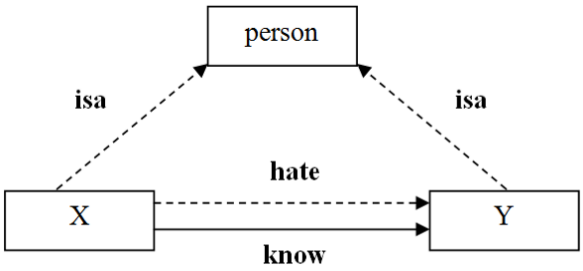


Fig. 8 Holmes rule

Capitals X and Y define variables X and Y, which are universally quantified. Fig. 8 expresses a *universal conditional* clause.
Using FCFL, the existential (skolem) constants are marked with the prefix @ at the beginning of the node name.
For example (see Fig. 9), the existential constants @person_x and @person_y introduce the individualities of the two persons.

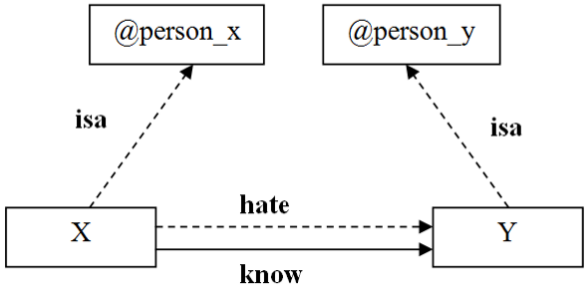


Fig. 9 Networks with existential constants

Predicates **isa** (**ako**) brings some elements of inheritance into a formal system.
If a statement **r**(X, Z) with a predicate **r** holds and at the same time isa(Y, X) holds, then also **r**(Y, Z) must hold. So it means in the language of GCFL the following auxiliary rule (Fig. 10) also has to be valid.

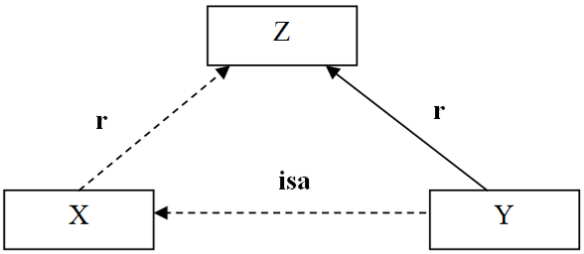


Fig. 10 Auxiliary rule

The clause at the Fig. 11 expresses the statement "If Anne does not know Jane, Jane is a stranger for Anne"
GCFL as well as CFL solves a problem of a base negative vector within a clause by means of "transfer" from antecedent to consequent and vice versa.

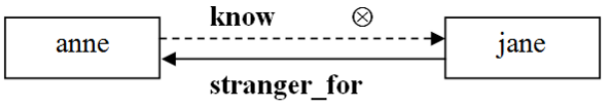


Fig. 11 Statement

For the ground clause holds a following *"rule of the transfer of negative ground atoms"*.
If a negative ground atom (vector) ought to be ordered into the antecedent set of atoms, transfer it as a positive atom into the consequent set of atoms.
If a negative ground atom (vector) ought to be ordered into the consequent set of atoms, transfer it as a positive atom into the antecedent set of atoms.
After a transfer of an antecedent vector into the consequent part the graph (Fig.12) becomes a graph of a clause without antecedent with a meaning "Anne knows Jane or Jane is a stranger to Anne."

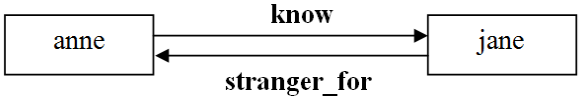


Fig. 12

Similarly GCFL as well as CFL also applies a special transfer rule in the case of universal/existential clause (for example this one of the Fig. 13) in the same way as in the extended RDF graph model, which works with quantifiers [4].
The clause "It is not true that Jane knows everybody, then somebody is a stranger for Jane." changes after the transfer into a clause "Jane knows everybody or somebody is a stranger for Jane."(Fig. 14).

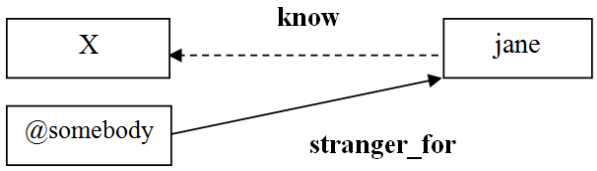


Fig. 13 Universal/existential clause

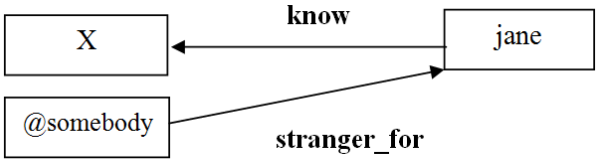


Fig. 14 Result clause

6. Resolution reasoning on GCFL knowledge base

GCFL works by the help of two rules - the substitution rule and the cut rule, both are well known in the first order logic. Together both rules form *the resolution rule* as well as in the CFL. In the frame of associative networks [2, 3] the resolution rule has been slightly modified and it is known as the *transfer rule*.

The *transfer rule*
Conditional clause is a tool of transferring its consequent into another clause unfixable with its antecedent. It is possible (for example) to use the isa rule (Fig. 8) for transferring its consequent (after proper substitution into another network).
For example, if we apply the vector **have**(rex, fell) to the semantic network N (see Fig. 10), then we get an instance (Fig. 15) describing properties of animals and the position of Rex.

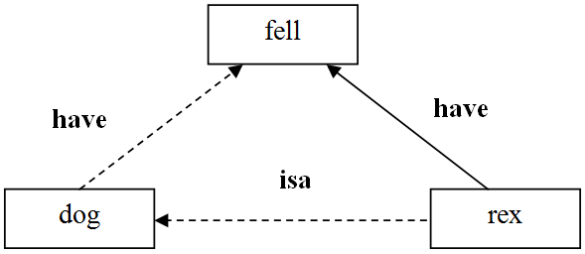


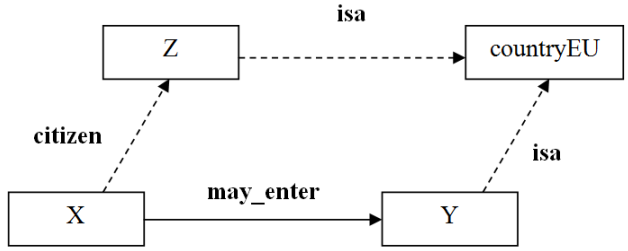
Fig. 15 The transfer rule

The *substitution rule*:
A new clause can be obtained from a clause with variables by a uniform substitution of a term for some of the variables.

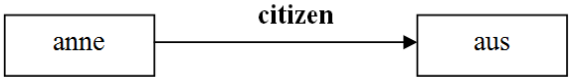
The *cut rule*:
If there are two clauses sharing the same atom in the knowledge base for reasoning, one in the antecedent of the first clause and one in the consequent of the second clause, then we can obtain a new clause by cutting out the same atoms at both sides and create a new clause with antecedent (consequent) that contains all the atoms of the original clauses antecedents (consequents).

7. Examples of the resolution reasoning in GCFL

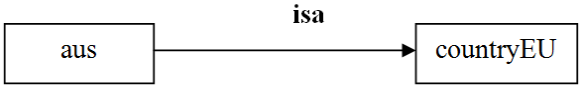
Example 1
Immigration rules of an EU country for citizens of EU countries in the language of GCFL forms a knowledge base in the following set of graphs {1, 2, 3, 4}.
1.



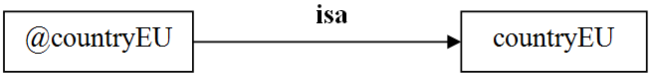
2.



3.

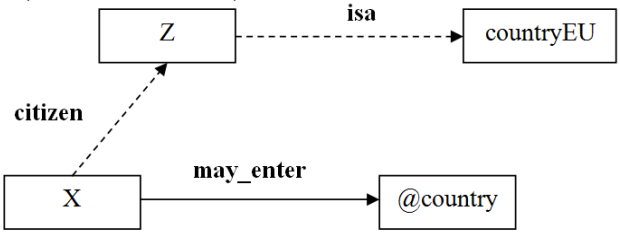


4.

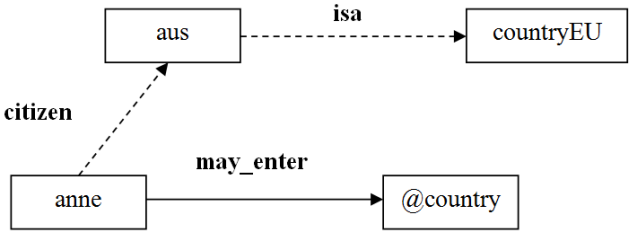


Proof of the statement „Anne may enter.“ in the GCFL on the knowledge base {1, 2, 3, 4}. Graphs 1 – 4 are prerequisites of the proof.

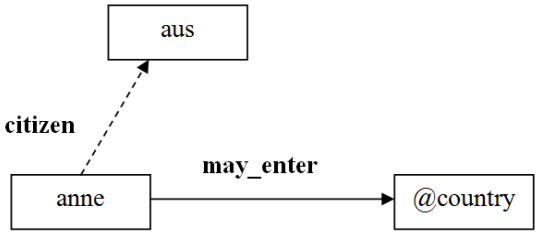
5. (1 after a cut with 4)



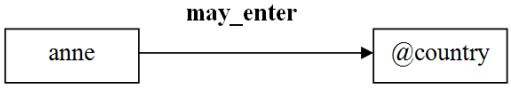
6. (5 after the substitution [anne/X, aus/Z]



7. Cut on the clauses 3 and 5



8. Cut on the clauses 2 and 5

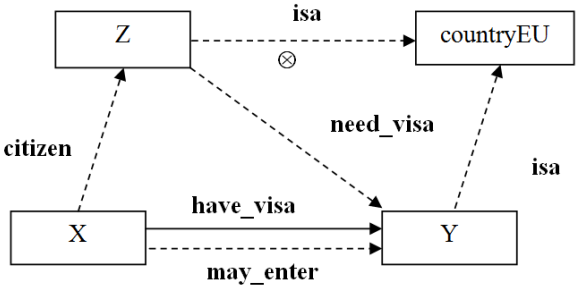


Example 2

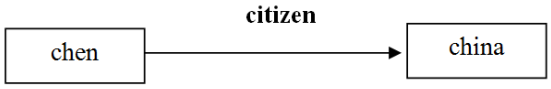
Indirect proof that a citizen of China Chen cannot enter an EU country without a visa:

Graphs 1 – 8 are prerequisites of the proof, graph 9 is a conclusion to be proved. Graphs 10 – 12 are (shortened) steps of the proof.

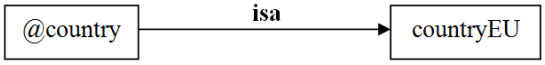
1.



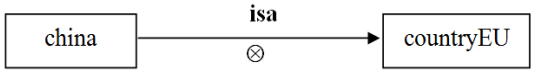
2.



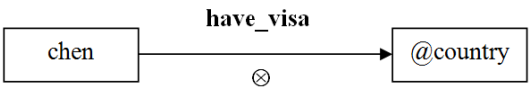
3.



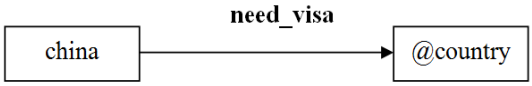
4.



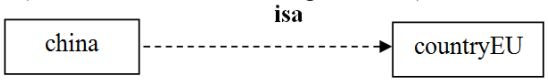
5.



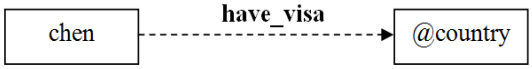
6.



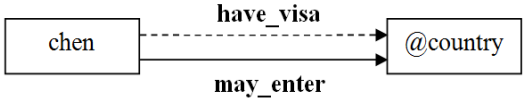
7. (4 after a transfer into a negative fact)



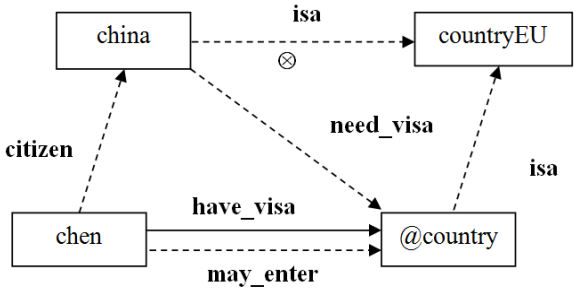
8. (5 after a transfer into a negative fact)



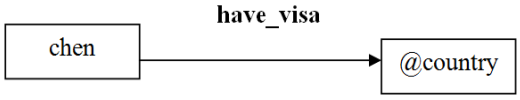
9. A conclusion to be proven



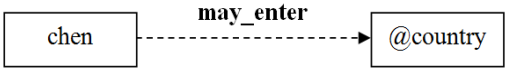
10. (1 after the substitution [chen/X, china/Z, @country/Y] and after the substitution the transfer of negated ground vector isa(china, countryEU) into the clause consequent)



11. (9 after cuts with positive facts citizen(chen, china), need_visa(china, @country), isa(@country, countryEU) and negative fact isa(china, countryEU))



12. Negation set of clause 9.



13. After two cuts on the graphs (clauses) 11 and 12 the resulting graph becomes empty, it is an inconsistent clause – a contradiction has been obtained.

8. Conclusions

As every formal system ought to operate on knowledge bases because of obtaining new consequents and interrelations between them it is natural to develop knowledge systems that can manipulate with structured data in a straightforward graph/based way without a necessity of rewriting knowledge into a language like OWL. Moreover, at present a concept of Linked Data built on RDF model lies at the proper heart of what Semantic

Web is all about. The idea of the RDF knowledge representation in its graph/based modification has its predecessor in a concept of associative (semantic) networks as a simple and understandable tool to represent of knowledge bases.

Our system GCFL gives a possibility to deduce new knowledge or to create new interesting interrelations in a straightforward way within graph representation. This fact offers us a possibility to use notifications and inference rules of GCFL also in the frame of RDF modeling. Reasoning, possibly supported by the graphical version of representation, then becomes more understandable and easier to use than that one of rewriting RDF models by description logic tools (OWL language).

References

[1] Richards, T.: Clausal Form Logic. An Introduction to the Logic of Computer Reasoning. Addison-Wesley, 1989.

[2] Lukasová, A.: Knowledge representation in associative networks (in Czech) Proceedings of Znalosti 2001. Praha, 2001.

[3] Lukasová, A., Telnarová, Z., Habiballa, H., Vajgl, M.: Formal knowledge representation (in Czech).Universum, 2010. 345 p.

[4] Lukasová, A., Vajgl, M., Žáček, M.: Reasoning in RDF graphic formal system with quantifiers. Proceedings of the International Multiconference on Computer Science and Information Technology. 2010. pp. 67-72.

Alena Lukasova has been working as a professor at the Department of Informatics and Computers at the University of Ostrava (Czech Republic). Her interests include theoretical principles of information and knowledge systems and tools of representation of semantically structured knowledge (database systems with knowledge bases components and their semantics), formal deduction in concept oriented languages, formal ontology for information systems, principles of ontology driven (based) information systems, and sharable knowledge patterns. She is author more than 50 scientific publications.

Martin Zacek graduated at the University of Ostrava (Czech Republic). The focus of his dissertation is a formal deduction in graph knowledge representation systems. The PhD thesis includes four graph systems: semantic (associative) networks, conceptual graphs of Sowa, RDF model and Topic Maps. The topic of the PhD thesis corresponds to the content of this article. He is author more than 10 scientific publications. In 2010 he won award “Young Scientist” at the International Multiconference on Computer Science and Information Technology awarded by the International Fuzzy Systems Association Distinction for the presentation of article Reasoning in RDFgraphic formal system with quantifier.

Marek Vajgl graduated at the University of Ostrava (Czech Republic) and defended PhD. thesis with title “A proposal of tool for creating and updating knowledge bases for semantic web”. He has been working as a lecturer at the Department of Informatics and Computers, University of Ostrava. His interests include descriptive logic formalism which is frequently used as semantic web formalism. He is author more than 20 scientific publications.