Reasoning domain ontologies at the meta-level

Francky Trichet, Frédéric Fürst

LINA - FRE CNRS 2729
Computer Science Research Institute
University of Nantes
2 rue de la Houssinière - BP 92208
44322 Nantes Cedex 3, France

{trichet,furst}@univ-nantes.fr

Abstract. In this paper, we present a new approach for reasoning domain ontologies at the meta-level. This approach is based on the application of ontology operationalization mechanisms in the context of a particular ontology of representation defined at the meta-level. The relevance of this work is illustrated in the context of two core questions for semantic interoperability: ontology evaluation and ontology matching. Our work is also compared with the OMG architecture dedicated to model engineering.

Keywords: Ontology, Meta-level, Ontology Reasoning, Ontology Matching, Conceptual Graphs

1 Introduction

Currently, ontologies are as the heart of many important Information Technology issues because they enable reasoning on domain assertions. For instance, in the context of the Semantic Web, ontologies are principally used to represent the content of web resources in order to facilitate concept-based Information Retrieval. However, managing multiple ontologies (for instance in a context of semantic interoperability) or analysing an ontology from an internal point of view (for instance for verification or validation purpose), do not require to reason on domain assertions (by using an ontology) but to reason directly on the ontologies. In other words, domain ontologies are not used as a reference system to perform reasonings on a fact base, but they are the objects on which the reasonings are performed. This approach requires the possibility to represent a domain ontology in a similar way as a knowledge base, *i.e.* a set of assertions which are defined according to a particular ontology O_1 and on which can be applied reasoning mechanisms (based on rules and constraints of O_1) for deducing news assertions.

In this paper, we claim that for reasoning on a domain ontology, it is relevant to represent it at the meta-level, in order to consider it as a knowledge base and thus to make any kind of reasoning possible, such as ontology mapping/matching, ontology merging, ontology verification, ontology validation, etc. Our approach

consists in using an ontology of representation called MetaOCGL for representing a domain ontology expressed in OCGL [1]. OCGL is a knowledge representation language based on the Conceptual Graphs formalism[1]. MetaOCGL is the ontology of representation which describes all the modeling primitives (and their relations) of OCGL and its formal semantics. This ontology is represented in OCGL and this is why we call it MetaOCGL.

The rest of the paper is structured as follows. Section 2 presents the context of our work: the OCGL ontology representation language, its corresponding representation ontology MetaOCGL, and the basic fondations of the ontology operationalization process. In section 3, the architecture we advocate for ontology reasoning at the meta-level is presented and illustrated with an example of ontology verification. An application of our meta-level reasoning approach to the problem of ontology matching is also introduced. Finally, we compare our approach to the Model Driven Architecture (MDA).

2 Context of the work

2.1 OCGL: Ontology Conceptual Graphs Language

The OCGL modeling language (*Ontology Conceptual Graphs Language* [1]) we advocate for specifying an ontology (at the conceptual level) is based on three building blocks: Concepts, Relations and Axioms. Representing an ontology in OCGL mainly consists in (1) specifying the conceptual vocabulary of the domain and (2) specifying the semantics of this conceptual vocabulary through axioms.

The conceptual vocabulary consists of a set of **Concepts** and a set of **Relations** which can be structured by using both well-known conceptual properties, called Axiom Schemata, and Domain Axioms.

The **Axiom Schemata** proposed in OCGL are:

1. the *ISA* link between two concepts or two relations (subsomption property) used to construct concept/relation taxonomies (tree or lattice);
2. the *Abstraction* of a concept (which corresponds to an *Exhaustive-Decomposition* in some works [3]);
3. the *Disjunction* between two concepts[2];

[1] The Conceptual Graphs model, first introduced by Sowa [7], is an operational knowledge representation model which belongs to the field of semantic networks. This model is mathematically founded both on logics and graph theory [7]. Two approaches for reasoning with CGs can be distinguished: (1) considered CGs as a graphical interface for logics and reasoning with logic and (2) considered CGs as a graph-based knowledge representation and reasoning formalism with its own reasoning capabilities. In our work, we adopt the second approach by using the *projection* (a graph-theoretic operation corresponding to homomorphism) as the main reasoning operator; *projection* is sound and complete w.r.t. deduction in FOL.

[2] Note that it is possible to define a *Partition* [3] by using the abstraction and the disjunction. For instance, the decomposition of `Number` into (`OddNumber` and `EvenNumber`) is a partition because `Number` is an abstract concept and `OddNumber` and `EvenNumber` are disjoint.

4. the *Signature* of a relation;
5. the *Algebraic properties* of a relation (symmetry, reflexivity, transitivity, irreflexivity, antisymmetry);
6. the *Exclusivity* or the *Incompatibility* between two relations[3];
7. the *Cardinalities* (Maximal and Minimal) of a relation.

OCGL has been implemented in a tool, called TooCoM[4] (*a Tool to Operationalize an Ontology with the Conceptual Graph Model*), dedicated to the edition and operationalization of domain ontologies. Thanks to this tool, it is possible to define the conceptual primitives (concepts and relations) and to specify the axiom schemata in a graphical way. Figure 1 shows an extract of an ontology dedicated to family relationships.

Domain Axioms differ from axiom schemata in the sense that they are totally specific to the domain whereas axiom schemata represent classical properties of concepts or relations. The OCGL graphical syntax used to express such an axiom is based on the Conceptual Graphs model. Thus, an axiom is composed of an *Antecedent part* and a *Consequent part*, with a formal semantics that intuitively corresponds to: *if the Antecedent part is true, then the Consequent part is true*. Figure 1 shows the OCGL graph representing the axiom *"The enemy of my friend is my enemy"*. In the concept hierarchy, an arrow represents a subsomption link between a concept and one of its parent, a concept without surround is abstract, the crossed circles represent disjunctions between concepts. In the relation hierarchy, a crossed circle represents an incompatibility (or exclusivity) between two relations, algebraic properties and cardinalities of a relation are indicated by symbols above the name of the relation (S for symmetry, T for transitivity, C+ and C- for the cardinalities, etc.). In the axiom part, the bright nodes represent the antecedent part, the dark ones the consequent part. Each part contains concept nodes (indicated by rectangles) and relation nodes (indicated by ellipses). A concept node is described by a label and a marker that identifies the considered instance (the marker $*$ denotes an undefined instance). A relation node is only described by a label. An edge between a concept and a relation is labeled with the position of the concept in the signature of the relation. The logical expression of the graph is automatically generated.

2.2 MetaOCGL: an ontology of representation

MetaOCGL is an ontology of the OCGL language, expressed in OCGL. MetaOCGL can then be considered as an ontology at the meta-level [3]. As shown in figure 2, MetaOCGL includes all the concepts of OCGL and their relations (*isa* relation, exclusivity/incompatibility between relations, disjunction of concepts, links between relations and concepts in a graph that expresses an axiom). MetaOCGL

[3] The incompatibility between two relations R_1 and R_2 is formalized by $\neg(R_1 \wedge R_2)$, the exclusivity is formalized by $\neg R_1 \Rightarrow R_2$.

[4] TooCoM is available under GNU GPL license at `http://sourceforge.net/projects/toocom/`.

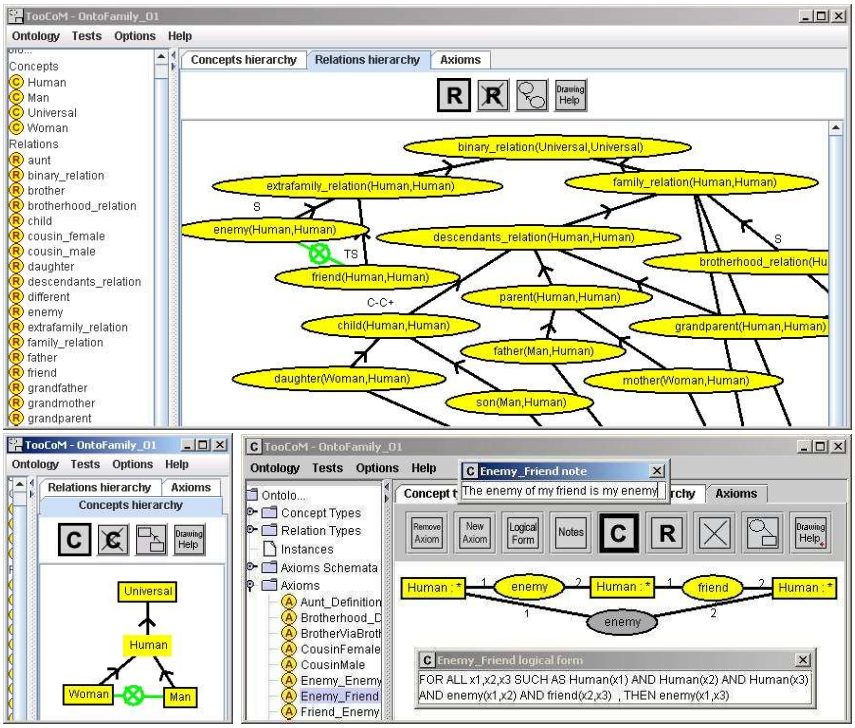

Fig. 1. Representation of OntoFamily O_1 in TooCoM. The upper part presents the relation hierarchy, the bottom left part presents the concept hierarchy, and the bottom right part presents an axiom.

also includes axiom schemata and domain axioms which express the formal semantics of OCGL.

A domain ontology can be represented as a MetaOCGL instance (*i.e.* a MetaOCGL graph), as domain facts can be represented by OCGL graphs. The MetaOCGL graph that represents an ontology contains a part which is dedicated to the representation of the concept hierarchy, a part which is dedicated to the representation of the relation hierarchy, and as many part as axioms in the ontology. Figure 3 shows the MetaOCGL graphs dedicated to the representation of the two axioms of OntoFamily O_1 *"the enemy of my enemy is my friend"* and *"the enemy of my friend is my friend"*, and their corresponding meta-graphs in MetaOCGL.

2.3 Operationalization: basic fondations

Ontologies can be used to shared knowledge between systems or between systems and humans, to reason on knowledge bases or to search in knowledge bases. Thus, they have to integrate all the knowledge of a given domain, and not only

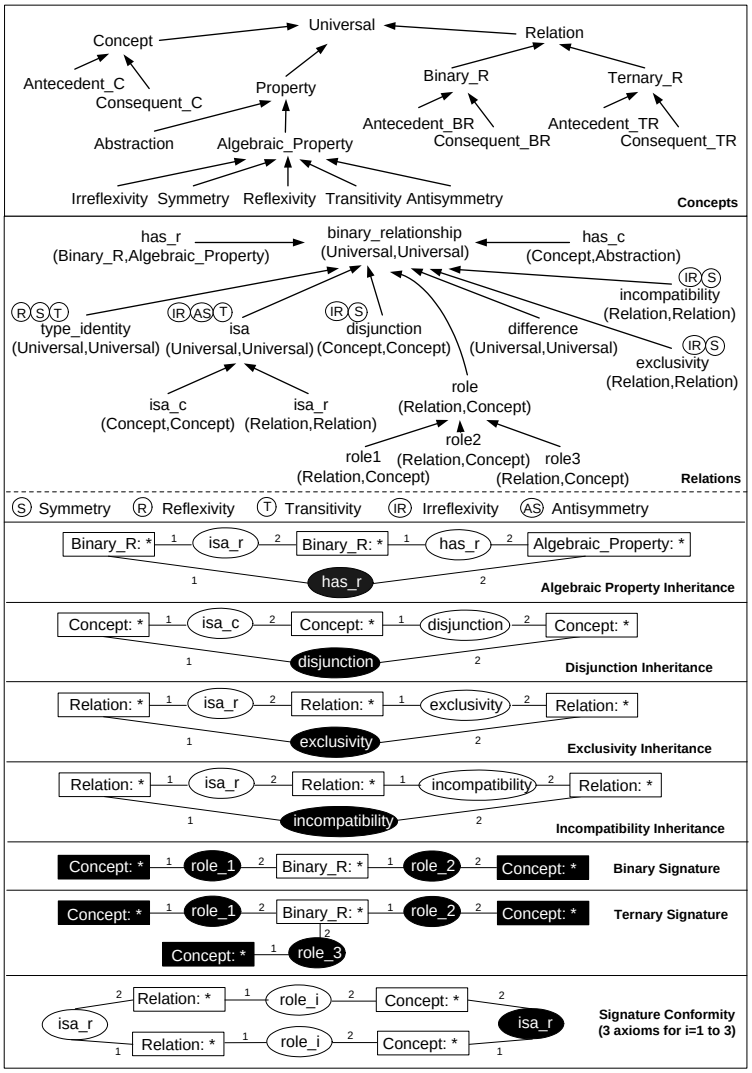

Fig. 2. Concepts, relations and axioms of MetaOCGL.

the terminological knowledge, as in a thesaurus. Ontologies have to evolve from *lightweight ontologies*, which only include some well-known properties such as subsomptions or algebraic properties, to *heavyweight ontologies*, which include all axioms that are needed to represent the semantics of the domain [8].

But, for keeping the independence of an ontology from the applications where it is used, in order to ensure its portability, the representation of the axioms must only precise their *formal semantics*, which constraint the interpretation of the conceptual primitives, without forcing their *operational semantics*, which fix the

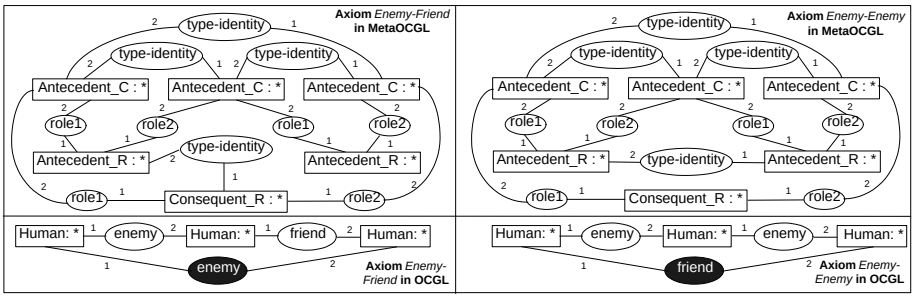

Fig. 3. Two axioms of OntoFamily O_1 represented in MetaOCGL. The *type_identity* links denote that the nodes are of the same type in the axiom. The two graphs are the same without considering type_identity links, but they differ when considering these links, because relations in the hypothesis part of the right axiom have the same type, but not those of the left one.

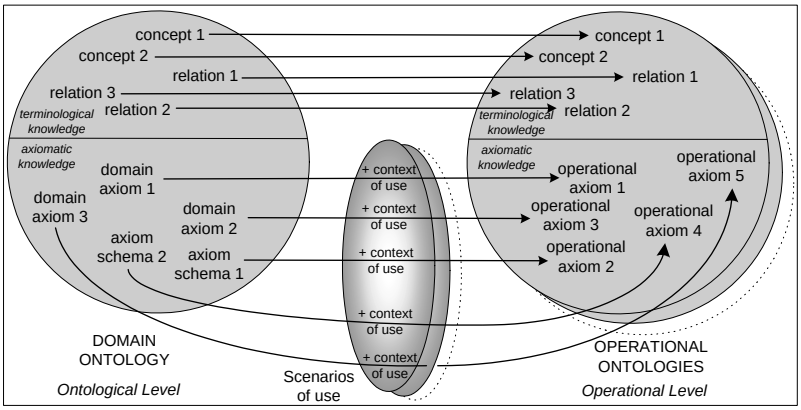

Fig. 4. The operationalization process of an heavyweight ontology. Terminological knowledge is represented in the same way at the ontological level and the operational level. The representation of axiomatic knowledge at the operational level depends on a scenario of use that describes the way the axioms are used to reason in the operational ontology. The operational representation of an axiom is a set of rules and/or constraints.

way the axioms are used in a KBS to reason [1]. For example, the axiom *"the enemy of my enemy is my friend"* can be used to produce knowledge (*i.e.* to deduce, when there exists an enemy of one of my enemies, that he is my friend) or to check assertions (*i.e.* to check that any enemy of one of my enemies is my enemy). An axiom can also be automatically applied by the system or explicitly applied by the user. The combination of these two criteria produces four different contexts of use for an axiom: (i) *inferential implicit* to automatically produce new knowledge from given facts, (ii) *inferential explicit* to allow the user to produce new assertions from given facts, (iii) *validation implicit* to automatically check

a fact base, and (iv) *validation explicit* to allow the user to check a fact base at his request[5].

Using heavyweight ontologies in applications requires their *operationalization*, which consists in (1) specifying the way the axioms will be used to reason through a *scenario of use* and (2) transcribing the axioms in operational forms (rules and/or constraints) according to the adopted scenario of use. Because the operational semantics of each axiom have to be specified, building a scenario of use consists in choosing, for each axiom, its *context of use* which defines the way the axiom will be used[6]. For a given ontology, each scenario of use (*i.e.* a set of contexts of use) leads to a different operational ontology, as shown in figure 4.

Operationalizing a domain ontology corresponds to building a KBS which can be used to reason on facts on the domain. For example, operationalizing OntoFamily O_1 in an inferential scenario of use produces an operational ontology appropriated to deduction: given few persons, linked only by descendants links, the KBS can automatically deduce parents links, brotherhood links and other family links. Another operationalization, in a validation scenario of use, can be used to check a fact base of family relationships, in order to evaluate its consistency, its completeness and its conciseness.

The OntoFamily O_1, represented in OCGL, includes 3 concepts, 31 binary relations, 11 axiom schemata and 18 axioms. Operationalizing O_1 for automatically completing a set of facts, *i.e.* in an inferential implicit scenario of use, leads to an operational ontology which includes the same terminological knowledge (3 concepts and 31 relations) but 39 rules and 7 constraints. The operational ontology automatically generated can be used, for example, to deduce, from a graph that represents facts which only deals with direct parent relations (father and mother), all the family relationships such as brotherhood relations, grandfather and grandmother relation, niece, nephew, uncle, aunt, etc. (*cf.* [2] for more detail).

3 Reasoning domain ontologies at the meta-level

3.1 Operationalization at the meta-level

Since domain ontologies are conceptual representations of a domain, their operationalization produces operational ontologies that enable reasoning on domain facts. In the same way, reasoning on ontologies themself can be done by operationalizing the representation ontology on which they are based, *i.e.* a

[5] Note that "deduction vs validation" and "implicit vs explicit" contexts of use are fine-grained examples of knowledge uses. At a more general level of granularity, a scenario of use can specify the reasoning mechanism used in a KBS (deduction, abduction, induction), or the goal of the KBS (*e.g.* teaching system or corporate memory management).

[6] According to the structure of the axiom and to the context of use, the operational form can be a rule, a constraint, or a set of rules and constraints (*cf.* [1] for more detail about this transformation process).

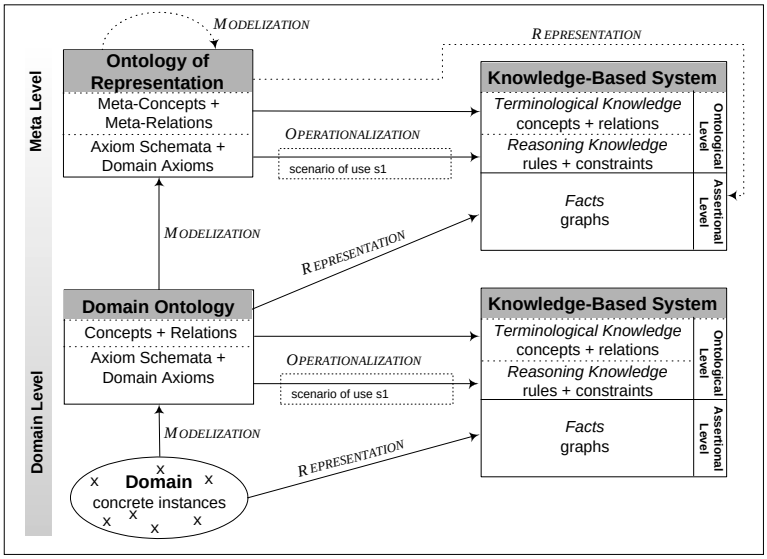

Fig. 5. Overview of the interactions between Domain Ontology, Ontology of Representation and KBS.

meta-ontology. More precisely, if we consider ontologies expressed in the OCGL language (for example), operationalizing the ontology of the OCGL language produces operational ontologies that permit to reason about the first ontologies. The ontology of OCGL, called MetaOCGL, represents knowledge about the OCGL language, the primitives of the language, and their semantics expressed through axioms. Operationalizing MetaOCGL consists in choosing the way the axioms will be used to reason about an ontology expressed in OCGL, for example OntoFamily O_1. To complete O_1, by automatically adding subsomption links, or by propagating inherited properties, for example, MetaOCGL have to be operationalized in an inferential scenario of use. To validate OntoFamily according to the OCGL formal semantics, MetaOCGL have to be operationalized in a validation scenario of use.

Figure 5 shows the interactions that exist between Domain Ontology, Ontology of Representation and KBS. It also underlines the three main activities related to the integration of ontologies into KBS: *Modelling*, *Operationalization* and *Representation*.

At the domain level, an ontology (called Domain Ontology in figure 5) of a particular domain (called Domain in the figure) is built via a *modelling* process. Reasoning about facts on this domain in a KBS is allowed by operationalizing the ontology according to a particular scenario of use which describes the way the axiomatic part of the ontology is used in the KBS. Then, the generated operational ontology can be used to reason about facts which are representations of instances of the domain. To sum-up, the *modelling* of a domain leads to a domain

ontology including *Concepts*, *Relations* and *Axioms* (both axiom schemata and domain axioms). The *Operationalization* of a domain ontology leads to the development of the ontological level of a KBS, including *Terminological Knowledge* (concepts and relations) and *Reasoning Knowledge*, *i.e.* rules and constraints corresponding to the operational forms of the axioms in the context of use which has been choosen. Finally, the *Representation* of a domain leads to the construction of the *Assertional Level* of the KBS, *i.e.* facts which are defined according to the *Terminological Knowledge*, and which are manipulated by the *Reasoning Knowledge*.

This three-step process (*Modelling, Operationalization, Representation*) can also be applied at the meta-level (*cf.* figure 5). The Ontology of Representation modelizes the language used to express the Domain Ontology. This ontology of representation is also expressed with the considered language. It can be operationalized in a KBS, and the generated operational ontology enables reasoning on the Domain Ontology. In this KBS defined at the meta-level, a fact is the representation of a particular domain ontology, for example a graph which represents OntoFamily O_1 in MetaOCGL. Because the ontology of representation is a meta-representation, modelizing this ontology in the same language produces the same ontology of representation. But this ontology can be represented as a fact in a KBS which implements an operational version of it, in order to reason on the ontology of representation itself.

3.2 Operationalization of MetaOCGL: an application to ontology evaluation

In order to use the MetaOCGL ontology for ontology evaluation (which includes verification, validation and assessment activities [3]), it is necessary to operationalize it in a validation and explicit scenario of use, *i.e.* all the axioms are used to validate a fact base. In the example of the figure 6, this fact base is the graph which represents an extract of the OntoFamily O_1. An error has been voluntarily introduced in the signature of the *"aunt(Woman,Universal)"* binary relation: this relation is a sub-relation of the *"relation_involving_a_Man(Woman,Human)"* relation. So, the signature of *"aunt"* is not in conformity with those of *"relation_involving_a_Man"*. The application of the signature conformity axiom (*cf.* figure 2), in a validation context of use, reveals the problem: the dark part of the graph is those which corresponds to the breaking of the axiom.

Note that our approach allows the knowledge engineer to explicitly define, through the definition of axioms at the meta-level, the criteria used to evaluate the content of ontologies in terms of consistency, completeness and conciseness. This declarative definition of criteria at the conceptual level increases both the portability and the modularity of the evaluation criteria, which, in most of the similar works, are directly hard-coded in the tools.

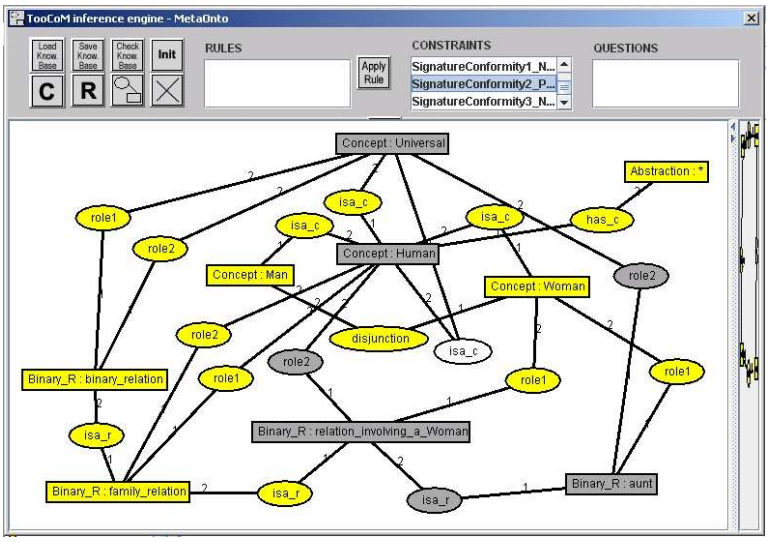

Fig. 6. Operationalization of MetaOCGL for ontology verification.

3.3 Operationalization of MetaOCGL: an application to ontology matching

The objective of ontology matching is to discover and evaluate identity links between conceptual primitives (concepts and relations) of two given ontologies supposed to be built on connected domains. Our approach relies on the use of the axiomatic level of the ontologies to discover semantic analogies between primitives, in order to reveal identities between them and to calculate the similarity coefficient of these identities, *i.e.* a coefficient that indicates how closely two concepts or relations are related. The comparison of axioms is based on their representation at the meta-level, in order to preserve their formal semantics but to erase their syntactical differences, while existing matching algorithms are essentially based on syntactical comparison [5].

Our algorithm takes as input two ontologies O_1 and O_2 (represented in OCGL) and provides as output potential similarities between two concepts or two relations: the result is a set of matchings (P_i, P'_j, C), where P_i and P'_j are respectively conceptual primitives (concepts and relations) of O_1 and O_2, and C the similarity coefficient between P_i and P'_j. Both axiom schemata and domain axioms are used to evaluate or discover primitive matchings. Of course, the weight of each OCGL property is used to modulate its influence on the evaluation of the matching.

As introduced in figure 3, domain axioms are represented in MetaOCGL, in order to compare their structures independently of their syntax. For each axiom couple (a_1, a_2), where $a_1 \in O_1$ and $a_2 \in O_2$, the representations of a_1 and a_2 in MetaOCGL, $meta(a_1)$ and $meta(a_2)$, are built. These representations can be enriched by adding information about the nodes: for instance, in figure 3, the

two relations *enemy* of the axiom in OCGL are represented in MetaOCGL by the two concepts *Antecedent_R* which are linked by the meta-relation called *type_identity*.

Two types of topological equivalence are then considered: the **equivalence**, that occurs when projections (in the context of the CG model) exist from $meta(a_1)$ to $meta(a_2)$ and from $meta(a_2)$ to $meta(a_1)$, without considering the *type_identity* relations, and the **typed equivalence** that occurs when the two projections exist with the *type_identity* relations. Of course, the weight of a typed equivalence is higher than those of an equivalence. A typed equivalence (resp. equivalence) between two axioms increases the coefficient of nodes linked by projection by the weight of the axiom typed equivalence (resp. equivalence). For example, the two axioms of figure 3 are equivalent because two projections exist between their meta-graphs without considering the *type-identity* relations. When considering the *type-identity* relations, there exists no projection, so they are not typed equivalent.

We have applied these principles to the matching of two ontologies related to the family domain [2]. This experiment has shown the relevance of the comparison of two ontologies at the meta-level, even if the algorithm has to be improved, in particular by taking the subsumption links into account.

4 Conclusion

In this paper, we have presented a new approach for reasoning domain ontologies at the meta-level, approach which mainly relies on the application of ontology operationalization mechanisms to an ontology of representation. We have also illustrated the relevance of our work in the context of two core questions for semantic interoperability: ontology evaluation and ontology matching.

In the field of model engineering, the OMG has defined a complete architecture called MDA (Model Driven Architecture) [6]. In the MDA, a specific model (level M1) captures each aspect of a system, a meta-model (level M2) captures a model and a meta-meta-model (level M3) captures a meta-model (*cf.* figure 7). For example, an UML model modelizes an application, UML modelizes this UML model and the MOF modelizes UML. The M0 level is the "real world", that is the applications in software engineering. A similar architecture can be considered in the field of ontology engineering: an ontology (level M1) is a model of a knowledge domain, a meta-ontology (level M2) is a model of ontology and a meta-meta-ontology (level M3) is a model of meta-ontology. For example, Onto-Family O_1 is a model of the family relationship domain, OCGL is a model of O_1, and Meta-OCGL is a model of OCGL.

What we claim is that reasoning the M1 level (for ontology validation or verification, ontology querying, ontology mapping, etc.) can be done by using terminological and axiomatic representations at the M3 level. This approach, based on a specific operationalization process (which can be compared to a transformation in model engineering), provides a more portable and modular method for reasoning domain ontology than methods which are only defined at the M2

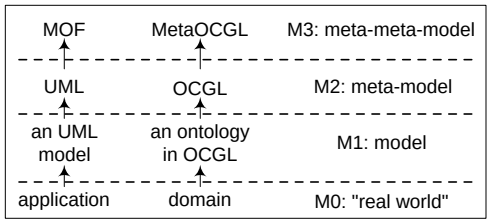

Fig. 7. Standard OMG layered organization illustrated with UML and comparison with ontology engineering and OCGL.

level and thus totally specific to a predefined knowledge representation language. This approach can also increase the efficiency of ontology mapping, by enabling axiom comparison and primitive matching at the meta-level, independently from syntactical considerations.

Moreover, a link can be etablished between our approach in ontology engineering and model engineering techniques. Ontologies currently evolve from lightweight ontologies, used as simple descriptions of domains, to heavyweight ontologies, used to reason on domains. In a similar way, models evolve from descriptions of systems to sources for automatic application building [4]. In this context, ontology engineering and model engineering both focus on using more and more abstract representations, in order, on the one hand, to improve model or ontology building and (re)using and, on the other hand, to increase their independence from platforms.

References

1. F. Fürst, M. Leclère, and F. Trichet. Operationalizing domain ontologies: a method and a tool. In R. Lopez de Mantaras and L. Saitta, editors, *European Conference on Artificial Intelligence (ECAI'2004)*, pages 318–322. IOS Press, 2004.
2. F. Fürst and F. Trichet. Axiom-based ontology matching: a method and an experiment. Research report 05-02, LINA, Nantes, `http://www.sciences.univ-nantes.fr/lina/fr/research/reports/`, 2005.
3. A. Gomez-Perez, M. Fernandez-Lopez, and O. Corcho. *Ontological Engineering.* Springer, Advanced Information and Knowledge Processing, 2003.
4. J. Greenfield and K. Short. Software Factories: Assembling Applications with Patterns, Models, Frameworks and Tools. In *Proceedings of the 18th Conference on Object-Oriented Programming, Systems, Languages and Applications (OOPSLA'2003)*, pages 16–27, 2003.
5. Y. Kalfoglou and M. Schorlemmer. Ontology mapping: the state of the art. *The Knowledge Engineering Review*, 18(1):1–31, 2003.
6. R. Soley and the OMG staff. Model-Driven Architecture. 2000.
7. J. Sowa. *Conceptual Structures : information processing in mind and machine.* Addison-Wesley, 1984.
8. S. Staab and A. Maedche. Axioms are objects too: Ontology engineering beyong the modeling of concepts and relations. Research report 399, Institute AIFB, Karlsruhe, 2000.