Reasoning on the Semantic Web needs to reason both on ontology-based assertions and on ontologies themselves

Frédéric Fürst
Laboratoire de Recherche
en Informatique d'Amiens
33 rue Saint-Leu
80039 Amiens Cedex 01 - France
frederic.furst@u-picardie.fr

Francky Trichet
Laboratoire d'Informatique de Nantes Atlantique
2 rue de la Houssiniere - BP 92208
44322 Nantes - France
francky.trichet@univ-nantes.fr

ABSTRACT

In this article, we present a method which aims at using ontologies to perform different types of reasoning. This method, called *operationalization*, proposes to automatically adapt the representation of the axioms to the type of reasoning the KBS in which the ontology is integrated is dedicated to. This process is based on the description of the operational goal of the KBS through a scenario of use. So, we argue for the distinction between the problems of knowledge representation in ontologies, that are built at the conceptual level, and the problems related to reasoning with ontologies, that occur at the operational level. Moreover, because reasoning on the Web requires to reason both on domain knowledge and on ontologies, for instance for evaluation or alignment purposes, we propose an extension of the operationalization method that permits to reason on ontology by operationalizing a meta-representation of the language used to represent ontologies. The language we use, called OCGL, is based on the Conceptual Graphs model and a tool, called TooCoM, implements this language and the operationalization method.

Categories and Subject Descriptors

I.2.4 [**Artificial Intelligence**]: Knowledge Representation Formalisms and Methods—*representation languages, semantic networks*

Keywords

Knowledge Engineering, Knowledge Representation, Ontologies, Semantic Web, 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. But providing online ontology-based services to the end-users implies that the online services are able to test the available ontologies, to compare, align and merge them, and use them in an operational way in order to provide reasoning capabilities. Thus, to reason on the Web, we need ontologies to represent knowledge included in the online resources, for reasoning on the resources, but we also need applications that must be able to reason on the ontologies themself. 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 and on which can be applied reasoning mechanisms (based on rules and constraints of O) for deducing news assertions.

Moreover, reasoning on the Web requires the representation of axiomatic knowledge, that is not only terminological knowledge, but all the semantics of a knowledge domain: the description of a domain through concepts and relations between these concepts must be enriched by properties, such as subsomption, that express their semantics. The OWL properties correspond to this axiomatic level, but the need of a larger expressivity is now underlined. The wedding between the World Wide Web and Knowledge Engineering can only be celebrated when the « wedding cake » dreamed up by T. Berners-Lee will be completely build [3]. In this context, the question of the representation and the use of axiomatic knowledge is at the heart of most of the recent works on the Semantic Web.

In this article, we focus on the problematics of the operational use of axiomatic knowledge, that is the way properties expressed in ontologies can be used to reason. Because the need of very expressive ontology representation languages, we consider heavyweight ontologies, that include both well-none properties such as OWL properties, and general axioms, expressed as couples (antecedent,consequent). We propose an unified model for operationalizing ontologies that can be applied both to reasoning on domain assertions and on ontologies themself. In our method, the operationalization of an ontology relies on the specification of an operational goal, that describes the way the axiomatic knowledge of the ontology will be used to reason. The operational form of the ontology is then automatically generated, as a set of rules and/or constraints that can be directly used by an inference engine. The same mechanism is used to operationalize a meta-representation of the ontology representation language, to produce an operational form of this meta-representation that allows to reason on every ontology expressed in the language.

WWW2006, May 22–26, 2006, Edinburgh, UK.
.

The language we use in our work, called OCGL (Ontology Conceptual Graph Language), is based on the Conceptual Graph model and permits to represent both terminological and axiomatic knowledge, with classical properties like subsomption or algebraic properties but with also general axioms for representing properties that can not be expressed as classical ones. Ontologies in OCGL are represented in a graphical way, including axioms, and the Conceptual Graph model provides reasoning mechanisms based on graph homomorphism that are used for reasoning both on domain knowledge and ontologies. OCGL, and the operationalization mechanisms, are implemented in a tool called TooCoM (a Tool to Operationalize an Ontology with the Conceptual Graphs Model) available on the Web at *http://sourceforge.net/projects/toocom/*.

The rest of this paper is structured as follows. Section 2 presents the OCGL language and the way the axiomatic knowledge is represented in this model. Section 3 first introduces the process we advocate to operationalize an ontology and then shows the application of this process in the context of the Conceptual Graphs model. Finally, section 4 presents the application of the operationalization to a meta-representation of OCGL in order to reason on the ontologies expressed in OCGL.

2. ONTOLOGY CONCEPTUAL GRAPH LANGUAGE

The OCGL modeling language (*Ontology Conceptual Graphs Language* [6]) we advocate for specifying an ontology (at the conceptual level) is based on the Conceptual Graph model and its extensions [1]. 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 [12].

The conceptual vocabulary consists of a set of **Concepts** (*cf.* figure 1), a set of **Relations** (*cf.* figure 2) and a set of ontological instances of concepts. An ontological instance of a concept is an instance required to express the semantics of the domain. For example, in the domain of mathematics, π is an ontological instance of the concept *Number*, because the expression of many axioms of this domain requires this instance. However, 3.54 is not an ontological instance. The sets of concepts and relations can be structured by using both well-known conceptual properties, called **Schemata Axioms**, and **Domain Axioms**. Schemata axioms are related to one or two conceptual primitives (concepts or relations), so they are represented in the primitive hierarchies, by symbols that complete the trees, as presented in figures 1 and 2.

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

1. the *ISA* link (subsomption property) between two concepts or two relations used to construct concept/relation taxonomies (tree or lattice);
2. the *Abstraction* of a concept, which corresponds to an *Exhaustive-Decomposition* in some works [9];
3. the *Disjointness* of two concepts. Note that it is possible to define a *Partition* [9] by using the abstraction and the disjointness. 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;

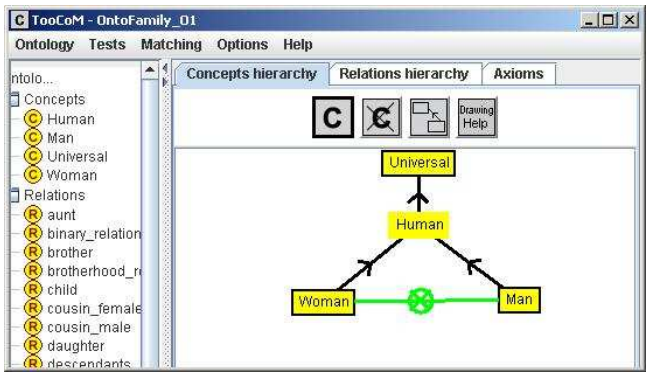

Figure 1: Representation of a concept hierarchy in OCGL. An arrow represents a subsomption link between a concept and one of its parents, a concept without surround is abstract, the crossed circles represent disjointness of concepts.

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. The incompatibility between two relations R_1 and R_2 is formalized by $\neg(R_1 \wedge R_2)$, the exclusivity is formalized by $\neg R_1 \Rightarrow R_2$. Incompatibility between relations is similar to disjointness of concepts;
7. the *Cardinalities* (Minimale and Maximale) of a relation[1].

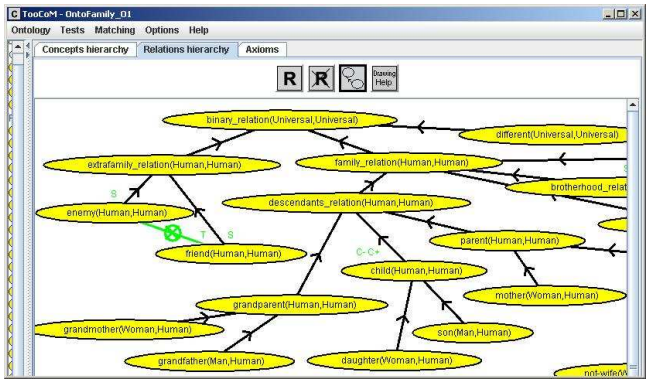

Figure 2: Representation of a relation hierarchy in OCGL. A crossed circle represents an incompatibility (or exclusivity) between two relations, algebraic properties and cardinalities of a relation are indicated by symbols above the name of the relation (S for symmetry, T for transitivity, C+ and C- for the cardinalities, etc.).

Domain Axioms differ from Schemata Axioms in the sense that they are totally specific to the domain whereas

[1]Note that we consider cardinalities of n-ary relations, that can be specified for any of the concepts of the relation signature, as OWL only allows binary relations.

Schemata Axioms represent classical properties of concepts or relations. The OCGL graphical syntax used to express such a Domain Axiom is based on the Conceptual Graphs model. Thus, a Domain 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 3 shows the OCGL graph representing the axiom *"The enemy of my friend is my enemy"* related to OntoFamily, a simple ontology dedicated to family relationships. Note that this axiom is a real Domain Axiom because it cannot be represented by using classical properties, in comparison with the axiom "*The friend of my friend is my friend*" which is represented by the transitivity of the relation called `Friend(Human,Human)`, that is a Schemata Axiom of OCGL.

OCGL has been implemented in a tool called TooCoM (*a Tool to Operationalize an Ontology with the Conceptual Graph Model*)[2] Thanks to this tool, it is possible to define the conceptual primitives (concepts and relations) and to specify the Schemata Axioms and the Domain Axioms in a graphical way.

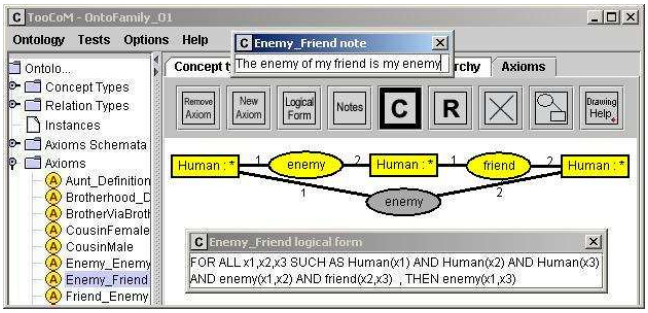

Figure 3: Representation in OCGL of the axiom *"The enemy of my friend is my enemy"*. **The bright nodes represent the antecedent part, the dark ones the consequent part. The logical expression of the graph is automatically generated.**

Note that TooCoM also allows the user to import ontologies expressed in OWL, by using the OWL API[2]. Classes in OWL correspond to Concepts in OCGL, and binary relations of OWL correspond to binary relations in OCGL. Properties of classes and relations in OWL are translated into axiom schemata in OCGL, but some properties are not now translated, because of the difference of expressivity of the two languages. For instance, the *allValuesFrom*, *someValuesFrom* and *hasValue* properties in OWL are not tranlated in OCGL. Conversely, the axioms of OCGL can not be translated in OWL as long as OWL does not offer capability of rule-like axioms representation. But the extension of OCGL is planned, in order to get closer the expressivities of the two languages.

3. REASONING ON DOMAIN WITH ONTOLOGIES

[2]TooCom is dedicated to the edition and the operationalization (*cf.* section 3) of domain ontologies [5]. It is available under GNU GPL license at `http://sourceforge.net/projects/toocom/`.

Most of the current ontologies are *lightweight ontologies*, that only integrate terminological knowledge and a few properties used to structure it, in particular subsomption properties. To reason on the Web, ontologies have to capture the whole knowledge of a domain, and to evolve from lightweight ontologies to *heavyweight* ontologies, that include all axioms that are needed to represent the semantics of the domain. The RuleML initiative [4] and the SWRL initiative [11] are based on the statement that the representation of heavyweight ontologies requires the use of rule-like expressions, that we call axioms.

But, for keeping the independence of an ontology from the applications where it is used, in order to ensure its portability and reutilisability, 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 way the axioms are used in an application to reason [6]. For instance, the symmetry property of a binary relation has a well-known semantics. But this property can be used in different ways to reason, according to the operational goal of the system in which the property is used: for instance to produce new knowledge by deducing, from the existence of such a relation, the existence of a new one; or to check a fact, to verify that if such a relation exist, the symmetric property also exists.

So, to be used for reasoning on the Web, an (heavyweight) ontology has to be thrown into the operational level, that is the formal semantics specified by the axioms has to be completed by the specification of an operational semantics which describes the way the axioms are implemented to reason. First, the formalism of such an operational ontology must be operational, that is it must provides operational mechanisms allowing the KBS to manipulate the representations for reasoning purposes. Secondly, the representation of the ontology in this operational formalism must be in accordance with the types of reasonings the system is dedicated to. So, the operationalization of an ontology consists, on the one hand, in choosing the operational representation language which offers manipulation mechanisms compatible with the considered operational goal and, on the other hand, in adapting the representation of the ontology to this goal by specifying the operational semantics of the knowledge expressed at the conceptual level. This operational semantics is determinated by the considered application, whereas the formal semantics depends on the considered domain (and of course on the formal semantics in which the ontology is written). Thus, as shown in figure 4, an ontology can be used to produce several operational ontologies for different kind of reasoning.

3.1 Operationalization: basic fundations

To adapt an ontology to an operational goal, we propose to specify this goal through a **scenario of use**, that describes the operational semantics of the ontology. A scenario of use specifies the way the knowledge specified in the ontology will be used. It essentially describes what the domain axioms (and schemata axioms) will be used for. Because the representation of terminological knowledge of the domain does not depend on the many possible application contexts, the representation of a concept or a relation will be the same for a system dedicated to knowledge validation or a system dedicated to knowledge production. Then, only operational

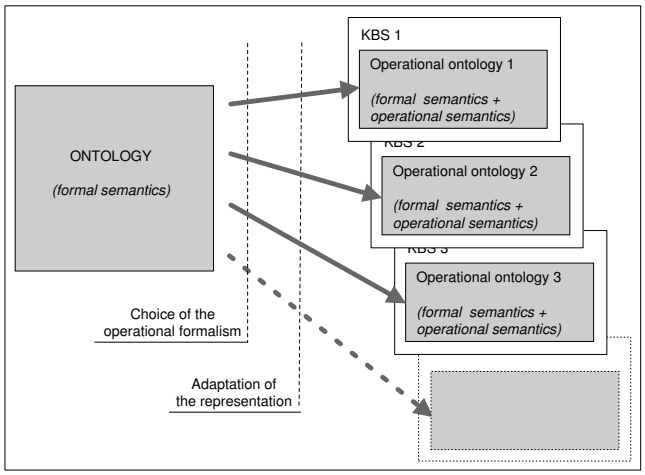

Figure 4: Operationalization process: an overview. The same ontology can lead to several operational ontologies appropriated to different KBS.

representations of axioms have to be adapted to the goal of the considered application[3].

Generally speaking, knowledge is used to produce new knowledge or to validate existing knowledge. The knowledge manipulation can also be done automatically by the system or it can be driven by the user. So, to specify a scenario of use, we propose to consider two criteria:

- the **inferential** or **validation** use of the knowledge expressed in the ontology. 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).

- the **implicit** (*i.e.* automatic) or **explicit** (*i.e.* driven by the user) use of the knowledge expressed in the ontology. The previous axiom can be used to automatically complete or check a knowledge base, without the user asking for this. But the user can also be forced to apply the axiom, for instance for educational purpose, what implies that the axiom is only managed by the user.

The combination of this two criteria produces four possible scenarios of use: the inferential and implicit one, the inferential and explicit one, the validation and implicit one and the validation and explicit one. Two particular cases of scenario of use can be distinguished: the pure validation scenario, where all the axioms are only used to validate a knowledge base according to the semantics of the domain, and the inferential and implicit scenario where the axioms are automatically used to produce new facts, without user intervention. In the last case, which is those of expert systems, the automatic inferences are supposed to produce knowledge in accordance with the semantics of the domain and no validation is required. The most common scenarii combine inferential and validation uses of axioms. For instance, a scenario dedicated to a computer-aided teaching application allows the user to apply knowledge to deduce new facts and also to check his work. Such a scenario comprises automatic inferences and validation processes, in accordance with the level of the user. So, in these mixed scenarii, the knowledge engineer must specify for each axiom the way it will be use in the KBS. We call this specification the **context of use** of an axiom[4].

[3]Of course, using different conceptual paradigms to represent the ontology and the operational ontologies (for instance the Frame paradigm and the Entity/Relation paradigm) requires in addition a modification of the representation of the terminological knowledge. In TooCoM, only the Entity/Relation paradigm is used.

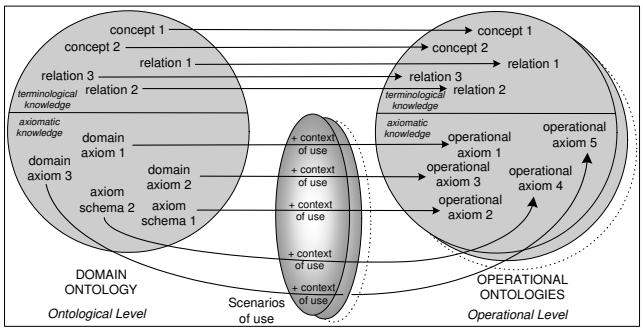

Figure 5: Operationalization process: in details. Each combination of contexts of use produces an operational ontology different from the others.

The scenario of use of an ontology is then composed of all the contexts of use of the domain axioms (and schemata axioms) of the ontology (*cf.* figure 5). The contexts of use we propose are those which correspond to the combination of the criteria previously given[5]:

- The **inferential and explicit** context of use: the user applies the axiom by himself on a fact base to produce new facts;

- The **inferential and implicit** context of use: the axiom is automatically applied by the system on a fact base to produce new facts;

- The **validation and implicit** context of use: the axiom is applied by the system to verify that a fact base is in accordance with the semantics of a domain.

For the schemata axioms, the same context of use can be specified for all the axioms that correspond to a given schema. For example, the user can choose an inferential and implicit context of use for all the axioms that express a symmetry relationship, in order to automatically produce symmetric relations.

[4]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 (deduction, abduction, induction), or the general goal of the application (*e.g.* teaching system or corporate memory management).

[5]We do not consider the **validation and explicit** context of use, because allowing this context for an axiom does not ensure that the knowledge base is in conformity with to the semantics expressed by this axiom since the user can choose to not check the base with this axiom.

	Inferential and Implicit Context of use	Inferential and Explicit Context of use	Validation and Implicit Context of use
axiom schema that concerns concept	$\emptyset$	$\emptyset$	$\emptyset$
incompatibility and exclusivity	1 implicit negative constraint + n implicit rules (for a n-ary relation)		
axiom schema (that concerns relation) and axiom with only relations in the consequent	1 implicit rule	1 explicit rule + a set of negative and implicit constraints	a set of negative and implicit constraints
axiom with concepts in the consequent	1 implicit rule	1 explicit rule	$\emptyset$

Figure 6: Automatic operationalization rules for the CGs model.

When the scenario of use is specified for an ontology and an application that exploits this ontology, the corresponding operational ontology must be produced in the choosen operational language. For instance, we have define rules for operationalizing ontologies expressed in OCGL into the Conceptual Graphs model. This model offers rules and constraints that enable the representation of axiomatic knowledge at the operational level. Both positive and negative constraints are available: the semantics of a positive constraint is *if the hypothesis part of the constraint is present, then the conclusion part must be present*; the semantics of a negative constraint is *if the hypothesis part of the constraint is present, then the conclusion part must be absent.* So, given an domain axiom or a schema axiom of an ontology expressed in OCGL and the context of use of the axiom, the operationalization of the axiom produces a set of rules and/or constraint, explicit or implicit, according to operationalization rules that are summarized in figure 6.

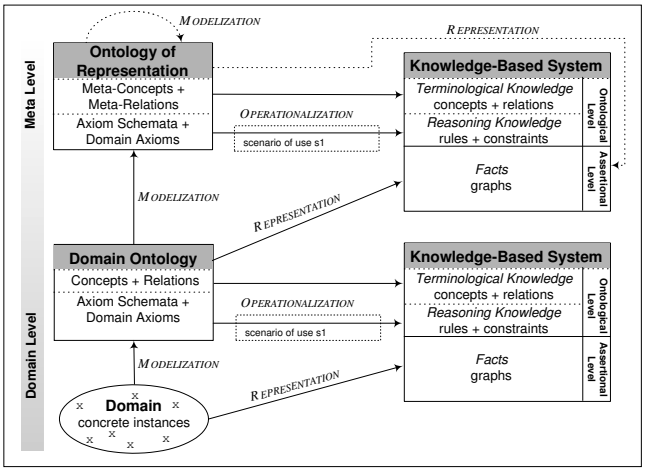

Figure 7: Overview of the interactions between Domain Ontology, Ontology of Representation and KBS.

4. REASONING DOMAIN ONTOLOGIES AT THE META-LEVEL

4.1 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 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.

To test this idea, we have built the ontology of OCGL, called MetaOCGL, which represents knowledge about the OCGL language, the primitives of the language, and their semantics expressed through axioms. MetaOCGL is an ontology of the OCGL language, expressed in OCGL and can then be considered as an ontology at the meta-level [9]. As shown in figure 8, MetaOCGL includes all the concepts of OCGL and their relations (*isa* relation, exclusivity/incompatibility between relations, disjointness of concepts, links between relations and concepts in a graph that expresses an axiom). MetaOCGL also includes schemata axioms and domain axioms which express the formal semantics of OCGL.

A domain ontology can then 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 (including schemata axioms), a part which is dedicated to the representation of the relation hierarchy (including schemata axioms), and as many part as axioms in the ontology. For instance, figure 9 shows the MetaOCGL graphs dedicated to the representation of the two axioms *"the enemy of my enemy is my friend"* and *"the enemy of my friend is my friend"*, and their corresponding meta-graphs in MetaOCGL.

Figure 7 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 7) 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 7). 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

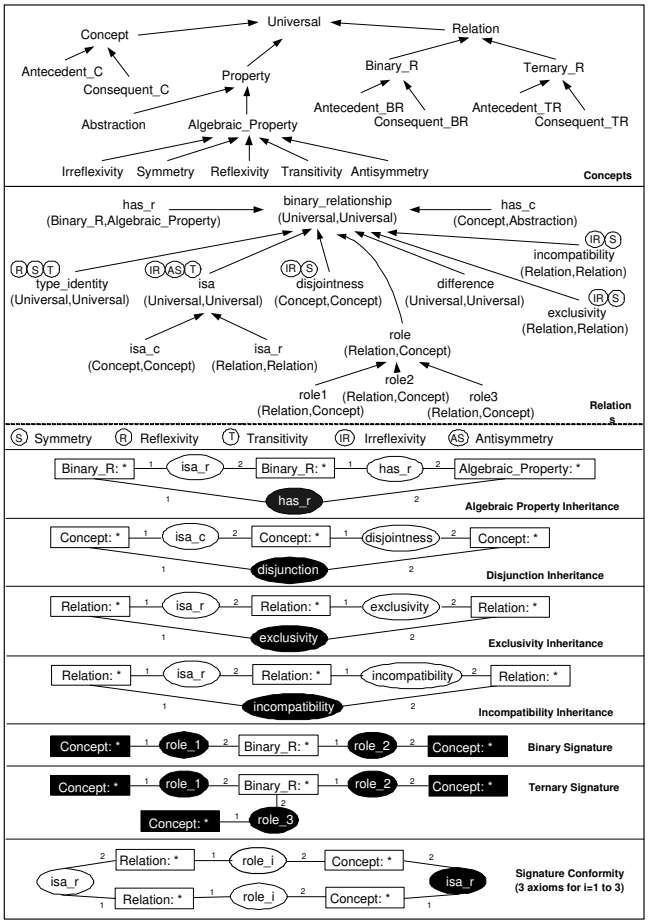

Figure 8: Concepts, relations and some of the axioms of MetaOCGL.

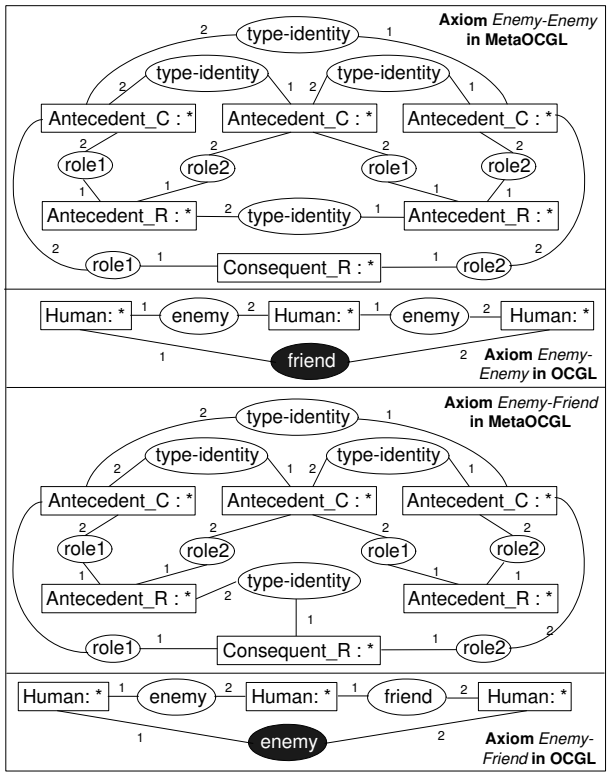

Figure 9: Two axioms represented in MetaOCGL *"The enemy of my friend is my enemy"* **and** *"The enemy of my enemy is my friend"* **. The** *type_identity* **links denote that the nodes are of the same type in the axiom. The two graphs are the same without considering type_identity links, but they differ when considering these links, because relations in the hypothesis part of the axiom at the top have the same type (***enemy***), but not those of the axiom at the bottom (***enemy* **and** *friend***).**

representation of a particular domain ontology, for example a graph in MetaOCGL which represents an ontology expressed in OCGL. 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.

Operationalizing MetaOCGL consists in choosing the way the axioms will be used to reason about an ontology expressed in OCGL. To complete an ontology expressed in OCGL, 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 an ontology according to the OCGL formal semantics, MetaOCGL have to be operationalized in a validation scenario of use.

4.2 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 [9]), it is necessary to operationalize it in a validation and implicit scenario of use, *i.e.* all the axioms of MetaOCGL are used to validate a fact base which corresponds to the meta-representation of a domain ontology. In the example of the figure 10, the 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 8), 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.

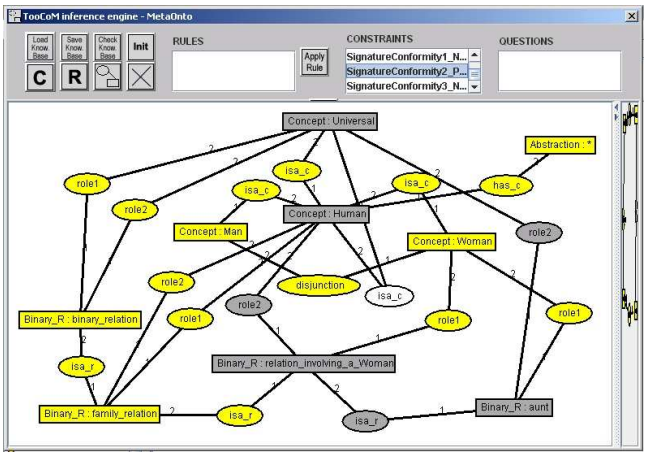

Figure 10: Operationalization of MetaOCGL for ontology verification.

4.3 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 [10].

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 9, domain axioms are represented in MetaOCGL, in order to compare their structures independently of the labels that appear in their nodes. 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 9, 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 9 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 [8, 7]. 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 subsomption links into account.

5. CONCLUSION

In this article, we have presented a method which aims at using ontologies to perform different types of reasoning. This method proposes to automatically adapt the representation of the axioms to the type of reasoning the KBS in which the ontology is integrated is dedicated to. This *operationalization* of ontology, based on the description of the operational goal of the KBS through a scenario of use, can also be applied for reasoning on the ontologies themself, by operationalizing an ontology of the language used to represent the ontologies.

What we claim here is that the representation of axioms in the ontologies of the Web have to be neutral towards the different types of reasoning, to ensure at most as possible the portability and the reutilisability of ontologies. But this neutrality supposes, for reasoning on the Web, that ontologies can be operationalized according to the different operational goals of the Web applications.

We only consider in this article a very narrow set of scenario of use, but the operationalization can be extended, for instance to adapt an ontology to an application according to a particular decidability or complexity class. In our mind, the problem of decidability or complexity have not to be considered at the ontological level, but only at the operational one. An ontology has to capture the whole knowledge of a domain, independently from reasoning problem. The construction of ontologies for the Semantic Web has to ignore the different class of OWL languages. At this step, only the expressivity of the language is considered. The restriction of such an ontology to a class of OWL or OWL+SWRL occurs when the ontology is used in an operational system. At this step, the operationalization will then consists both in changing some representation of axioms and removing some of them.

So, we argue that reasoning on the Web supposes that the ontological level and the operational level are separated, and that problems relative to reasoning are only considered at the operational level. Thus, the problem of ontology representation, and specially of axioms representation only depends of the expressivity the ontology representation languages are supposed to offer. The reasoning problems leads to adapt each ontology, according to the kind of reasoning for which we want to use it. This approach allows to ensure the reutilisability of ontologies, and to capture the whole semantics of a domain independently of any reasoning problem. Moreover, ontologies can be more easily used

to reason, because the operationalization process produces an operational ontology adapted to the kind of reasoning the KBS is supposed to performed.

In the case of two agents that communicate on the Web, only the ontology on which the dialog is based has to be shared. The operational scenario used by each agent depends on the reasoning the agent performs, and this operational scenario has to be shared only if the communication between the agents is about this scenario. Moreover, a scenario of use can be seen has an instance of a particular kind of PSM ontology, and can then be shared as ontology by Web agents.

The second point we want to underline is that reasoning on the Web both requires to reason on domain knowledge and on ontologies, because KBS on the Web require ontology evaluation and alignment. The method that we have presented allows to perform these two kinds of reasoning, by using a meta-representation of the used ontology representation language.

This work is also an example of the joint use of domain ontologies and meta-ontologies. The next step is to integrate in the same KBS domain ontologies, meta-ontologies, PSM ontologies and/or high-level ontologies, to perform more complex reasonings on the Web. In particular, PSM ontologies can be used to specify the scenario of use of the operationalization of a domain ontology.

6. REFERENCES

[1] J.F. Baget and M.L. Mugnier. Extensions of Simple Conceptual Graphs: the Complexity of Rules and Constraints. *Journal of Artificial Intelligence Research*, 16:425–465, 2002.

[2] Sean. Bechhofer, Raphael Volz, and Phillip Lord. Cooking the Semantic Web with the OWL API. In *Proceedings of the second International Semantic Web Conference (ISWC'2003)*, 2003.

[3] T. Berners-Lee. Conceptual Graphs and the Semantic Web. In *http://www.w3.org/DesignIssues/CG.html*, 2001.

[4] H. Boley, S. Tabet, and G. Wagner. Design rationale of ruleml : a markup language for semantic web rules. In *Proceedings of the Semantic Web Working Symposium (SWWS'2001)*, 2001.

[5] F. Fürst. TooCoM: a Tool to Operationalize an Ontology with the Conceptual Graph Model. In *Proceedings of the second Workshop on Evaluation of Ontology-Based Tools (EON'2003) at the International Semantic Web Conference (ISWC'2003)*, pages 57–70, 2003.

[6] F. Fürst, M. Leclère, and F. Trichet. Operationalizing domain ontologies: a method and a tool. In *Proceedings of the 16th European Conference on Artificial Intelligence (ECAI'2004)*, volume 110, pages 318–322. IOS Press, 2004.

[7] F. Fürst and F. Trichet. Aligner des ontologies lourdes : une méthode basée sur les axiomes. In *Actes des 16e Journées Francophones d'Ingénierie des Connaissances (IC'2005)*. Presses Universitaires de Grenoble, 2005.

[8] F. Fürst and F. Trichet. Axiom-Based Ontology Matching: a method and an experiment. Technical report RR 05-02, LINA, *http://www.sciences.univ-nantes.fr/lina/fr/research/reports/*, 2005.

[9] A. Gomez-Perez, M. Fernandez-Lopez, and O. Corcho. *Ontological Engineering*. Springer-Verlag, Advanced Information and Knowledge Processing, 2003.

[10] Y Kalfoglou and M. Schorlemmer. Ontology mapping: the state of the art. *The Knowledge Engineering Review*, 18(1):1–31, 2003.

[11] M. O'Connor, H. Knublauch, S.W. Tu, B.N. Grosof, M. Dean, W.E. Grosso, and M.A. Musen. Supporting Rule System Interoperability on the Semantic Web with SWRL. In *International Semantic Web Conference*, pages 974–986, 2005.

[12] 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.