Towards a principled approach to semantic interoperability

Jérôme Euzenat
INRIA Rhône-Alpes
655 avenue de l'Europe, 38330 Montbonnot Saint-Martin (France)
Jerome.Euzenat@inrialpes.fr

Abstract

Semantic interoperability is the faculty of interpreting knowledge imported from other languages at the semantic level, i.e. to ascribe to each imported piece of knowledge the correct interpretation or set of models. It is a very important requirement for delivering a worldwide semantic web. This paper presents preliminary investigations towards developing a unified view of the problem. It proposes a definition of semantic interoperability based on model theory and shows how it applies to already existing works in the domain. Then, new applications of this definition to family of languages, ontology patterns and explicit description of semantics are presented.
Keywords: Semantic interoperability, ontology sharing, knowledge transformation, ontology patterns.

1 Introduction

The vision of a "semantic web" [Berners-Lee, 1998; Berners-Lee *et al.*, 2001] is realized by the annotation of web pages, containing informal knowledge as we know it now, with formal knowledge. These annotations can reference each other and depend on ontologies and background knowledge. Taking advantage of the semantic web requires to be able to gather, compare, transform and compose the annotations. For several reasons (legacy knowledge, ease of use, heterogeneity of devices and adaptability, timelessness), it is not likely that this formal knowledge will be encoded in the very same language. The interoperability of formal knowledge languages must then be studied in order to interpret the knowledge acquired through the semantic web.

The problem of comparing theory is well known but it takes a fantastic importance in the context of the semantic web.

Semantic interoperability is the faculty of interpreting the annotations at the semantic level, i.e. to ascribe to each imported piece of knowledge the correct interpretation or set of models. It will be further characterized below by considering that the final transformed knowledge must have the transformation of the consequences of the initial knowledge as consequences.

There are several approaches to semantic interoperability [Masolo, 2000; Ciocoiu and Nau, 2000; Stuckenschmidt and Visser, 2000]. Although, they are not stated in the same terms, we believe that there can be a unified view of comparison and transformation at a semantic level that can be applied to these approaches.

We first provide some definitions of the concepts at work here (language, representation, semantics and transformation) and a classification of possible interoperability requirements. Then the already available approaches to semantic interoperability are considered and rephrased in the context of model-theory. Afterwards, we turn to consider three possible approaches for the semantic web (and especially when the languages are different): family of languages, ontology patterns and explicit semantics representation. We show how the contribution of these techniques to semantic interoperability can be expressed in comparable terms.

2 Principles

2.1 Language, semantics, transformation

For the simple purpose of the present paper, a language L will be a set of expressions. A representation (r) is a set of expressions in L. No distinction will be made between ontologies, background knowledge and formal annotations: they will all be representations.

There have been many studies of knowledge representation language semantics [Nebel, 1990]. The semantics is generally defined in model theory by using simple set theory. Usually, an interpretation function I, to a domain of interpretation D, is defined iteratively over the structure of the language L. The interpretation function is compositional, i.e. it builds the meaning of an expression from that of its sub-expressions (or components). The expressions δ in a language L are said to be satisfied by interpretation I if they meet a certain condition (usually that $I(\delta)$ belongs to a distinguished subset of the domain). In this framework, a model of a set of assertions $r \subseteq L$, is an interpretation I satisfying all the assertions in r. An expression δ is said to be a consequence of a set of expression r if it is satisfied by all models of r (this is noted $r \models_L \delta$).

A computer has to find if a particular expression (e.g. a query) is the consequence of a set of expression (e.g. a knowledge base). To that extent, executable systems (called provers) are developed which can be grounded on inference rules or more classical programs. From a set of axioms $r \subseteq L$, they establish if the expression $\delta \in L$ is a theorem (noted $r \vdash_L \delta$). These provers are said correct if any theorem is a consequence of the axioms and complete if any consequence of the axioms is a theorem. However, depending on the language and its semantics, the decidability (i.e. the existence of such provers) is not ensured and, even in this event, the algorithmic complexity of such provers can be prohibitive.

Hence, system developers must establish a trade-off between expressivity and complexity of representation languages or completeness of the prover. This choice has led to the definition of languages with a low expressivity (like simple conceptual graphs or object-based representations) or modular family of representation languages (like description logics). As a consequence, there are many different representations languages that can be used in the context of the semantic web. Therefore, if some annotation, ontology or background knowledge is found on the semantic web, it might have to be translated from a language to another.

Transformations are applied to representations in order to import them from one language to another. The transformations are functions $\tau : L \longrightarrow L'$ (L' can be L). These transformations must be computable (and so they are syntactic mechanisms) and in the context of the semantic web, they can be implemented as XSLT (or a similar language) stylesheets [James Clark (ed.), 1999]. These transformations can be composed into more complex transformations.

We will consider here the problem of ensuring the interoperability of representations through transformations. There are several levels at which interoperability can be accounted for.

2.2 Levels of interoperability

When trying to assess the understanding of an expression coming from a system by another one, there are several possible levels of interoperability:

- encoding: being able to segment the representation in characters;
- lexical: being able to segment the representation in words (or symbols);
- syntactic: being able to structure the representation in structured sentences (or formulas or assertions);
- semantic: being able to construct the propositional meaning of the representation;
- semiotic: being able to construct the pragmatic meaning of the representation (or its meaning in context).

This layered presentation is arguable in general; it is not as strict as it seems. It makes sense because each level cannot be achieved if the previous ones have not been completed. In the context of the semantic web, it can be assumed that the three first levels can be easily achieved by the use of XML or RDF.

The properties of transformations can be set at these various levels. There are many kinds of properties (e.g. at the syntactic level we could care of just preserving the elements, or their ordering or both). Here are some of their expressions:

- lexical: given a mapping σ between terms of a particular language (that can be the terms in a structured formal language or the lexical units of a natural language). For instance, one can ask that this mapping preserves *synsets* (connected components of the synonymy graph, S), i.e. $\forall t, t' \in L, S(t) = S(t') \Rightarrow S(\sigma(t)) = S(\sigma(t'))$.
- syntactic: Order-preservation, for instance, will require that, given two order relations $\leq_L$ and $\leq_{L'}$, if $r \leq_L s$, then $\tau(r) \leq_{L'} \tau(s)$.
- semantic: Consequence preservation requires that

  (1) $$\forall \delta, r \models_L \delta \Rightarrow \tau(r) \models_{L'} \tau(\delta);$$

  It is noteworthy that consequence preservation is not trivially granted by syntactic preservation, e.g. addition in the structure. As a matter of fact, adding a class or an attribute in a frame-based knowledge base is structure preserving (it preserve both elements and their order) though the second addition is not consequence preserving.
- semiotic: interpretation preservation (let Σ be the interpretation rules and $\models^i$ the interpretation relation for person i, $\forall \delta, \forall i, j, r, \Sigma \models^i \delta \Rightarrow \tau(r), \tau(\Sigma) \models^j \tau(\delta))$). This consists in mapping signs to equivalent signs with respect to the expected interpretation of a reader. These aspects can be related to rhetoric [Rutledge *et al.*, 2000] or pragmatics (i.e. properties not directly relevant to a compositional view of semantics but which interfere with sheer semantic interpretation).

We do not pretend that these properties must be satisfied in the semantic web. There can be situations where only some moderate preservation of meaning or content is sufficient. However, characterizing the exact properties of the available transformations will be very useful.

The present paper will only consider semantic interoperability and especially how the formula 1 can be satisfied in various contexts. This scheme, in which a first statement in a system constrains another one in another system, is necessary for relating statements across languages while pure logical statements can be used (e.g. in modal logics of knowledge [Fagin *et al.*, 1995]) when the language is shared across agents.

3 Related work

In the context of first-order logic ($\mathcal{FOL}$), Claudio Masolo [Masolo, 2000] has investigated the relations between logical theories. He first considers the deductive closure of sets of axioms ($Cn(r) = \{\delta | r \vdash_{\mathcal{FOL}} \delta\}$) and the relationships between these representations derived from the five possible containment relations on their deductive closure. Since this can only be applied to theories with coinciding vocabularies, he goes on by considering the definitional extension (noted $r'|^r$) of a representation r', by the definition of the terms (predicate and function symbols) of a representation r, in functions of the terms in r'. In such a case, importing a representation from an ontology into another is simply $r'|^r \cup r$. And this warrants that:

$$\forall\delta \in \mathcal{FOL}, r \models_{\mathcal{FOL}} \delta \Rightarrow r'|^r \cup r \models_{\mathcal{FOL}} \delta$$

So, consequence is trivially preserved (the importance of this notion in [Masolo, 2000] is related to inter-expressibility of theories). Of course, there can be no definitional extension of r' by r.

Claudio Masolo also considers translations (ϕ) which are transformations preserving the structure of formulas (i.e. atomic formulas are replaced by arbitrary formulas and the rest of the transformation is defined by induction on the structure of formulas) and renaming (σ) which are translations only affecting the name of predicates and functions. These translations can be though of as our transformation τ. The theories can be compared based on mutual translations:

$$\begin{aligned}
r \approx_{\tau,\tau'} r' &\text{ iff } r' \vdash_{\mathcal{FOL}} \tau(r) \text{ and } r \vdash_{\mathcal{FOL}} \tau'(r')\\
r \prec_{\tau,\tau'} r' &\text{ iff } r' \vdash_{\mathcal{FOL}} \tau(r) \text{ and } r \not\vdash_{\mathcal{FOL}} \tau'(r')\\
r \succ_{\tau,\tau'} r' &\text{ iff } r' \not\vdash_{\mathcal{FOL}} \tau(r) \text{ and } r \vdash_{\mathcal{FOL}} \tau'(r')\\
r \times_{\tau,\tau'} r' &\text{ iff } r' \not\vdash_{\mathcal{FOL}} \tau(r) \text{ and } r \not\vdash_{\mathcal{FOL}} \tau'(r')
\end{aligned}$$

For our purposes, this is equivalent to define:

$$r \preceq_{\phi} r' \text{ iff } r' \vdash_{\mathcal{FOL}} \phi(r)$$

and thus (by induction on the formula structures):

$$\forall\delta, r \models_{\mathcal{FOL}} \delta \Rightarrow \phi(r) \models_{\mathcal{FOL}} \phi(\delta)$$

Claudio Masolo shows that the equivalence relations through translations and the equivalence through definitional extensions are indeed equivalent (in terms of deductive closures). These relations have their equivalent characterization in model theory:

$$\begin{aligned}
&r \approx_{\tau,\tau'} r' \text{ iff } \forall\delta, r \models_{\mathcal{FOL}} \delta \Leftrightarrow r' \models_{\mathcal{FOL}} \delta\\
&r \prec_{\tau,\tau'} r' \text{ iff } \forall\delta, r \models_{\mathcal{FOL}} \delta \Rightarrow r' \models_{\mathcal{FOL}} \delta\\
&r \succ_{\tau,\tau'} r' \text{ iff } \forall\delta, r' \models_{\mathcal{FOL}} \delta \Rightarrow r \models_{\mathcal{FOL}} \delta\\
&r \times_{\tau,\tau'} r' \text{ iff } \exists\delta \in L, \delta' \in L', r \models_{\mathcal{FOL}} \delta, r' \models_{\mathcal{FOL}} \delta',\\
&\qquad\qquad\qquad r \not\models_{\mathcal{FOL}} \delta' \text{ and } r' \not\models_{\mathcal{FOL}} \delta
\end{aligned}$$

He demonstrates that, provided with completeness of first order logic, the straightforward semantics characterization for the theory with coinciding vocabularies (and theories equivalent through renaming alone) is equivalent to the syntactic one. The formulation of the equivalent of definitional extensions is related to the notion of "coalescent models" which is not detailed here.

A last contribution of [Masolo, 2000] is the comparison of logics whose set of models coincide while they do not use translatable primitives (e.g. the geometry based on points and those based on spheres). The notion of model-structure transformations (i.e. transformation applying at a semantic level) are introduced.

Ciocoiu [Ciocoiu and Nau, 2000], takes into account the implicit knowledge (K) that is not formally expressed in the ontologies but should be taken into account by building the models. Their framework uses a first-order logical language as a pivot languages in which both languages can be translated (and from which models can be extracted) and an ontology of explicit assumptions expressed in the formal language. The goals of this work is the translation-checking (i.e. knowing if a representation is the translation of another, i.e. if it has the same set of models). This is theoretically achieved by comparing the corresponding the sets of extracted models.

This can be further refined by considering two background knowledge sets (K and K'), the knowledge transformations can be justified in our framework without the common ontology and logical translation:

$$\forall\delta, K, r \models_{\mathcal{FOL}} \delta \Rightarrow K', \tau(r) \models_{\mathcal{FOL}} \tau(\delta)$$

In a similar vein, [Stuckenschmidt and Visser, 2000] introduced the idea that, beside the correct syntactic transformation required for semantic interoperability, there is room for several completeness levels that must be taken into account. The most basic level is sheer translation of what is (syntactically) transcriptible from the source representation. A second level consists of ensuring that whatever is a consequence of the source representation that can be expressed in the target language is indeed translated. In case of a more expressive source language this might require the use of a prover in order to deduce these formulas that can be represented by the target (but more generally, a prover might be required whatever the expressivity of either languages). This means that, given a sheer syntactic transformation τ, one must build a semantic transformation $\overline{\overline{\tau}}$ such that:

$$\forall\delta \in L', \tau(Cn(r)) \models_{L'} \delta \Rightarrow \overline{\overline{\tau}}(r) \models_{L'} \delta$$

or

$$\forall\delta \in L, r \models_{L} \delta \Rightarrow \overline{\overline{\tau}}(r) \models_{L'} \tau(\delta)$$

The translations of [Masolo, 2000] are such semantic transformations.

A further refinement, well represented in [Ciocoiu and Nau, 2000], is the explicitation of implicit knowledge, that can be added as background for the translated theory. In the context of geographical information integration [Visser *et al.*, 2000], the authors have integrated the domain ontologies by providing translations from the source ontologies in a target language and by reclassifying the corresponding concepts (grounded on their descriptions) with regard to each other.

OntoMorph [Chalupsky, 2000] is a system of syntactic transformation of ontologies with a syntactic transformation language not very different from XSLT. It however is integrated with a knowledge representation system (PowerLoom) which provides the opportunity to have semantically-grounded rules in the transformations. The system can query assertions for not only being syntactically in the source representation, but also for being a consequence of this initial representation (as soon as PowerLoom is semantically complete). This is a generic implementation of what is proposed in [Stuckenschmidt and Visser, 2000]. Of course, this option requires to use PowerLoom as an initial pivot language and the problem of translation arises when transforming from the source representation to the PowerLoom representation.

Semantic, knowledge or ontology patterns [Staab *et al.*, 2000; Clark *et al.*, 2000; Stuckenschmidt, 2000] have recently been introduced as the equivalent, in the ontology engineering field, of the design patterns (or rather frameworks) in software engineering. They are used for factoring out notions that are common across, and despite, languages. Instead of considering a knowledge representation construct in isolation, it considers a set of constructs and their interrelations satisfying a particular function (e.g. how to deal with part-whole relations, how to deal with class specialization). To that extent, ontology patterns offer a language $\mathcal{P}$ for expressing the required constructs and the constraints that hold between them. A pattern p is usually made of a set of terms T, a set of grammar rules for articulating them G and a set of constraints C. Implementations consist in instantiating the patterns, i.e. mapping the constructions and constraints to the concrete language. The mapping μ is specified in terms of signature morphism between the pattern p and an actual language L such that:

$$\forall \delta, p \models_{\mathcal{P}} \delta \Rightarrow \mu(p) \models_L \mu(\delta)$$

These contributions have especially considered semantic interoperability within the same language (using different sets of axioms or ontologies) or generic to specific languages (through pattern mapping). We will now take a look at several proposals for expressing semantic interoperability across different languages as it shall happen on the semantic web.

4 Language family approach

In the words of Tim Berners-Lee, the semantic web requires a set of languages of increasing expressiveness and anyone can pick up the right language for each particular semantic web application.

A modular family of languages is a set $\mathcal{L}$ of languages that have a similar kind of formulas (e.g. build from a subset of the same set of formula constructors) and the same kind of semantic characterization (i.e. if a formula belongs to two languages, it is interpreted in the same way in both). It is then easy to transform a representation from one language to another and one can take advantage of more efficient provers or more expressive languages.

This is what have been developed by the description logic community over the years: a family of representation languages with known decidability, complexity and equivalence results [Donini *et al.*, 1994]. It has been experimented for the web with the "Description Logic Mark-up Language"[1] (DLML) that we have developed.

DLML is not a language but rather a modular system of document type descriptions (DTD) encoding the syntax of many description logics. It takes advantage of the modular design of description logics by describing individual constructors separately. The specification of a particular logic is achieved by declaring the set of possible constructors and the logic's DTD is automatically build up by just assembling those of elementary constructors. The actual system contains the description of more than 40 constructors and 25 logics.

[1]http://co4.inrialpes.fr/xml/dlml/

To the DLML language is a associated a set of transformations (written in XSLT) allowing to convert a representation from a logic to another. The simplest transformation is the transformation from a logic to another syntactically more expressive one (i.e. which adds new formulas). The transformation is then trivial, but yet useful, because the initial representation is valid in the new language, it is thus identity:

$$\forall \delta \in \mathcal{AL}, r \models_{\mathcal{AL}} \delta \Rightarrow r \models_{\mathcal{ALC}} \delta$$

This trivial interpretation of semantic interoperability is one strength of the "family of languages" approach because, in the present situation, nothing has to be done for gathering knowledge. For this case, one can define the relation between two languages L and L' as $L \ll L'$ which has to comply with $L \subseteq L'$. We can then define $L \equiv L'$ as equivalent to $L \ll L'$ and $L' \ll L$.

We can further define $L \overline{\vee} L'$ by $L \ll L \overline{\vee} L'$ and $L' \ll L \overline{\vee} L'$ and there exists no other language L'' such that $L \ll L'' \ll L \overline{\vee} L'$ and $L' \ll L'' \ll L \overline{\vee} L'$. For all L and L', $L \overline{\vee} L'$ and $L \overline{\wedge} L'$ have to satisfy $L \cup L' \subseteq L \overline{\vee} L'$ and $\overline{\wedge} L' \subseteq L \cap L'$ (in the case of term-based languages such as description logics we have $L \overline{\wedge} L' = L \cap L'$, but not necessarily $L \overline{\vee} L' = L \cup L'$, see figure 1). This defines the syntactic structure of $\mathcal{L}$.

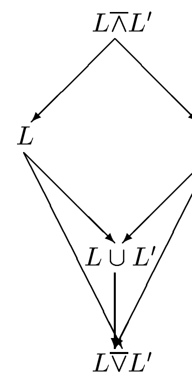

Figure 1: The relations between syntactic languages

If $L \not\ll L'$, the transformation is more difficult. The initial representation r can be restricted to what is (syntactically) expressible in L': $\overline{r}$. However, this operation (which is correct) is incomplete because it can happen that a consequence of a representation expressible in L' is not a consequence of the expression of that representation in L':

$$\exists \delta \in L'; \overline{r} \not\models_{L'} \delta \text{ and } r \models_L \delta$$

To solve this problem, as stated in [Visser *et al.*, 2000], it is necessary to deduce from r in L whatever is expressible in L'. Let $\overline{\overline{r}} = \overline{Cn(r)}$ be this expression. It is such that

$$\forall r \subseteq L, \forall \delta \in L \overline{\wedge} L', r \models_L \delta \Rightarrow \overline{\overline{r}} \models_{L'} \delta$$

The preceding proposal is restricted in the sense that it only allows, in the target language, expressions expressible in the source language, while there are equivalent non-syntactically comparable languages. This is the case of the description logic languages $\mathcal{ALC}$ and $\mathcal{ALUE}$ which are known to be equivalent while none has all the constructors of the other. For that purpose, one can define $L\overline{\overline{\lll}}L'$ if and only if the models are preserved, i.e. $\exists\overline{\overline{\tau}}$;

$$\forall r \subseteq L, \forall\langle I, D\rangle; \langle I, D\rangle \models_{L'} \tau(r), \Rightarrow \langle I, D\rangle \models_L r$$

Similarly, $L\overline{\overline{\equiv}}L'$ if and only if $L\overline{\overline{\lll}}L'$ and $L'\overline{\overline{\lll}}L$. Moreover, $L\overline{\overline{\nabla}}L'$ is defined by a language such that $L\overline{\overline{\lll}}L\overline{\overline{\nabla}}L'$ and $L'\overline{\overline{\lll}}L\overline{\overline{\nabla}}L'$ and there exists no other language L'' such that $L\overline{\overline{\lll}}L''\overline{\overline{\lll}}L\overline{\overline{\nabla}}L'$ and $L'\overline{\overline{\lll}}L\overline{\overline{\nabla}}L'$. [2]

The $\overline{\overline{\tau}}$ transformation is not easy to produce (and it can generally be computationally expensive) but we show, in §6 how this could be practically achieved.

Another possibility is to define $\widetilde{\lll}$ as the existence of an isomorphism between the models of r and those of $\tau(r)$

$$\exists\tau; \forall\langle I, D\rangle, \exists\langle I', D'\rangle; I, r \models_L \delta \Rightarrow I', \tau(r) \models_{L'} \tau(\delta)$$

This also ensures that $r \models_L \delta \Rightarrow \tau(r) \models_{L'} \tau(\delta)$.

This provides to the family of languages a structure based on semantics.

5 Pattern-based approach

The generic pattern based approach provides patterns of constructs involved in a language. In the present article, a pattern $p \in \mathcal{P}$ is characterized by a set of constructions that can be mapped to that of a language and an interpretation of these constructions that must be preserved by the mapping to a concrete language (this is also seen as a constraint). For instance, a CONJ(.) pattern is interpreted over sets as the intersection of conjunct constructions. It is mapped to the AND and ANDROLE operators in description logics or the class constructor in frame-based languages. The patterns do not provide a direct way to ensure the interoperability between two languages.

However, translating between two languages which share some patterns should be easier than the general case. As a matter of fact, if, as is assumed, the mappings preserve meaning (i.e. $p \models_{\mathcal{P}} \delta \Rightarrow \mu(p) \models_L \mu(\delta)$). Then, in the case of reversible or bijective mappings (i.e. such that

$$\exists\mu^{-1}; r \models_L \delta \Leftrightarrow p = \mu^{-1}(r) \models_{\mathcal{P}} \mu^{-1}(\delta))$$

it is possible to ensure that the transformation from L to L' made by $\mu' o \mu^{-1}$ indeed preserves meaning. More precisely, since the constraints are implemented and satisfied by both systems, only the mapping of constructors and grammar are required to be bijective.

Of course, not all such mappings are bijective, though there should be numerous cases in which the mapping is indeed bijective: it should be possible to establish when μ is bijective and to take advantage of this. Moreover, even if just a part of the constructors used in the pattern are in a bijective relationship with the target language – or if only a few patterns, among those instantiated by the language, are bijective – it is possible to take advantage of the affected knowledge in the target language (the transformation is not anymore complete, but at least it is correct).

With pattern languages, it seems desirable to decompose the language L in two parts: $\hat{L}$ and $\check{L}$ such that $\mu(p) = \hat{L}$ and $\check{L} = L - \hat{L}$. We then have $L = \hat{L} \vee \check{L} \lll \hat{L} \lll p$. But no non-trivial results are currently available about such a decomposition.

Like in the family of language approach, the mappings could be used for refining patterns themselves (i.e. $\mu : \mathcal{P} \longrightarrow \mathcal{L} \cup \mathcal{P}$). Then, as before, meet and join among patterns can be defined and a lattices of patterns can be extracted from which the frontier between bijective and non-bijective mappings for a particular language L can be extracted and systematically exploited.

6 Semantic description and transformations

As seen above, the expression of semantic interoperability relies on two ingredients: τ and $\models$. Its expression in machine-readable form can be achieved in various ways. τ can be expressed in XSLT or some similar language, but nothing really practical has been set up for $\models$.

We have defined the notion of Document Semantic Description (DSD) which enables to describe the formal semantics of an XML language (just like the DTD or schemas express the syntax). The DSD language, defined in XML takes advantage of Xpath for expressing references to sub-expressions and MathML for expressing the mathematical gear. The DLML family of languages contains the DSD of all the covered operators and is able to build automatically from the description of a logic the DSD of that logic.

DSD can be used for many purposes:

documenting language semantics for the user or the application developer who will require a precise knowledge of the semantics of constructs. This is eased by a transformation from DSD to LaTeX.

computing interpretations from the input of the base assignment of the variables.

checking proof of transformations is a very promising application in the line of the "web of trust" idea [Berners-Lee, 1998].

proving transformations in an assisted or automatic way;

inferring transformations from the semantics description is a very hard problem. However, from a given proof, it can be a straightforward task.

This program is rather ambitious. However, in some very restricted setting, this can be quite easy to set up. As an example, one can take the DLML context. Here, the languages have the same syntactic structure and the semantics of the operators remains the same across languages. Consider the $\mathcal{ALC}$ and $\mathcal{ALUE}$ languages which are known to be equivalent. The proof of equivalence is a demonstration that any operator missing in one language can be expressed in the other language (preserving interpretations). This iterative proof can

[2] $L\overline{\overline{\nabla}}L'$ and $L\overline{\overline{\wedge}}L'$ are defined as sets of languages and not as a particular language. If we want to define a lattice of language from these operators, they must be grouped in congruence classes (modulo equiexpressivity, e.g. $\mathcal{ALC}$ and $\mathcal{ALUE}$ are in the same class). But we cannot always guarantee that the result is a lattice.

be expressed, by a human being, that way:

$$\begin{aligned} &\forall\langle D,I\rangle, \ldots \\ I((\texttt{not Nothing}))) &\Leftrightarrow I(\texttt{Anything}) \\ I((\texttt{not}\ c)) &\Leftrightarrow I((\texttt{anot}\ c)) \qquad \text{for } c \in \mathcal{N}_C \\ I((\texttt{not}\ (\texttt{anot}\ c))) &\Leftrightarrow I(c) \\ I((\texttt{not}\ (\texttt{all}\ r\ c))) &\Leftrightarrow I((\texttt{csome}\ r\ (\texttt{not}\ c))) \\ &\ldots \end{aligned}$$

It is straightforward to transform that proof into the following XSLT templates:

```
...
<xsl:template mode="process-not" match="dl:NOTHING">
  <dl:ANYTHING/>
</xsl:template>

<xsl:template mode="process-not" match="dl:CATOM">
  <dl:ANOT>
    <xsl:apply-templates select="."/>
  </dl:ANOT>
</xsl:template>

<xsl:template mode="process-not" match="dl:ANOT">
  <xsl:apply-templates select="*"/>
</xsl:template>

<xsl:template mode="process-not" match="dl:ALL">
  <dl:CSOME>
    <xsl:apply-templates select="*[1]"/>
    <xsl:apply-templates mode="process-not"
                         select="*[2]"/>
  </dl:CSOME>
</xsl:template>
...
```

The last rule tells that when encountering a `ALL` in the scope of a negation (`mode="process-not"`), it must be transformed in a `CSOME` with the non-negated transformation of the first argument and the negated transformation of the second one.

Moreover, if we use a language for describing proofs in conjunction with DSD, then it is possible to document the languages with DSD, the transformation with the proof and a client application will have everything that is required for proof-checking the transformation before using it.

This shows that this approach can be useful in the context of family of languages. It can be useful in the context of the ontology pattern too. Again, having the proof of the semantic preservation of μ^{-1} is the key to having a correct transformation from L to L'.

7 Conclusion and discussion

The semantic web could be a distributed web of knowledge structure and interoperability can be problematic when knowledge is expressed in different languages or in function of different ontologies. This will be an obstacle to taking advantage of imported knowledge. Semantic interoperability attempts to ensure that the interpretation of imported and transformed knowledge remains the same across languages.

We have presented a framework for expressing semantic interoperability based on the notion of transformations and semantic consequences. It has been used to analyze the various techniques employed in order to enforce interoperability. Because it is a common framework, it can be used in order to articulate the various proposals and compose all the solutions into a global one. We showed that, applied to restricted settings, it helps a lot.

This work is only a preliminary study of the relation between our expression of semantic interoperability and the implemented tools for that purpose. It seems clear, however, that many refinements are possible in the context of particular families of languages or restrictions of knowledge patterns for formally ensuring interoperability.

One of our goal is the construction of transformations satisfying semantic interoperability by composing more elementary transformations satisfying it. This will depend on the kind of property satisfied by the transformations and the kind of composition. If the transformations and their properties are published in the semantic web, then it becomes possible to create such compound transformations more conveniently.

It is worth recalling, that semantic interoperability is not total interoperability and that even with semantic properties there can be other interesting properties than full-fledged correctness and completeness. We hope that future work will enable to characterize precisely the expected properties in semantic terms.

References

[Berners-Lee *et al.*, 2001] Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web. *Scientific american*, (5), 2001. http://www.scientificamerican.com/2001/0501issue/0501berners-lee.html.

[Berners-Lee, 1998] Tim Berners-Lee. Semantic web roadmap, 1998. http://www.w3.org/DesignIssues/Semantic.html.

[Chalupsky, 2000] Hans Chalupsky. OntoMorph: a translation system for symbolic knowledge. In *Proceedings of 7th international conference on knowledge representation and reasoning (KR), Breckenridge, (CO US)*, pages 471–482, 2000.

[Ciocoiu and Nau, 2000] Mihai Ciocoiu and Dana Nau. Ontology-based semantics. In *Proceedings of 7th international conference on knowledge representation and reasoning (KR), Breckenridge, (CO US)*, pages 539–546, 2000. http://www.cs.umd.edu/ nau/papers/KR-2000.pdf.

[Clark *et al.*, 2000] Peter Clark, John Thompson, and Bruce Porter. Knowledge patterns. In *Proceedings of 7th international conference on knowledge representation and reasoning (KR), Breckenridge, (CO US)*, pages 591–600, 2000.

[Donini *et al.*, 1994] Francesco Donini, Maurizio Lenzerini, Daniele Nardi, and Andrea Schaerf. Deduction in concept languages: from subsumption to instance checking. *Journal of logic and computation*, 4(4):423–452, 1994.

[Fagin *et al.*, 1995] Ronald Fagin, Joseph Halpern, Yoram Moses, and Moshe Vardi. *Reasoning about knowledge*. The MIT press, Cambridge (MA US), 1995.

[James Clark (ed.), 1999] James Clark (ed.). XSL transformations (XSLT) version 1.0. Recommendation, W3C, 1999. http://www.w3.org/TR/xslt.

[Masolo, 2000] Claudio Masolo. *Criteri di confronto e costruzione di teorie assiomatiche per la rappresentazione della conoscenza: ontologie dello spazio e del tempo*. Tesi di dottorato, Università di Padova, Padova (IT), 2000.

[Nebel, 1990] Bernhard Nebel. *Reasoning and revision in hybrid representation systems*. Lecture Notes in Artificial Intelligence 422. Springer Verlag, Berlin (DE), 1990.

[Rutledge *et al.*, 2000] Lloyd Rutledge, Jim Davis, Jacco van Ossenbruggen, and Lynda Hardman. Inter-dimensionnal hypermedia communicative devices for rhetorical structure. In *Proceedings of the Multimedia Modeling conference, ?, (?)*, 2000. http://www.cwi.nl/ lynda/publications.html.

[Staab *et al.*, 2000] Steffen Staab, Michael Erdmann, and Alexander Mädche. Semantic patterns, 2000. http://www.aifb.uni-karlsruhe.de/ sst/Research/Publications/semanticpatterns.pdf.

[Stuckenschmidt and Visser, 2000] Heiner Stuckenschmidt and Ubbo Visser. Semantic translation based on approximate re-classification. In *Proceedings of the KR workshop on semantic approximation granularity and vagueness, Breckenridge, (CO US)*, 2000. http://www.tzi.de/ heiner/public/ApproximateReClassification.ps.gz.

[Stuckenschmidt, 2000] Heiner Stuckenschmidt. A pattern-based ontology language, 2000. in preparation.

[Visser *et al.*, 2000] Ubbo Visser, Heiner Stuckenschmidt, G. Schuster, and Thomas Vögele. Ontologies for geographic information processing. 2000. http://www.tzi.de/buster/papers/Ontologies.pdf.