Formalization of the semantics of iconic languages: An ontology-based method and four semantic-powered applications

Jean-Baptiste Lamy[a,*], Lina F. Soualmia[b,a]

[a]*LIMICS, Université Paris 13, Sorbonne Paris Cité, 93017 Bobigny, France, INSERM UMRS 1142, UPMC Université Paris 6, Sorbonne Universités, Paris, France*
[b]*LITIS EA 4108, NormaSTIC CNRS FRE 3638, Université de Rouen Normandie, France*

Abstract

Iconic languages can represent concepts by the combination of graphical components (such as colors or pictograms). There are numerous examples, from traffic signs to computer user interface icons. However, these languages do not associate formalized semantics to their icons, which raises various problems: inconsistent combinations of graphical components, different interpretations of a given icon by two persons, difficulties to map the icons to the concepts of existing termino-ontological resources, *etc*.

In this article, we describe a method that formalizes the semantics of an iconic language with an ontology. This method was initially developed for the VCM iconic language (Visualization of Concepts in Medicine), which enables to represent the main medical concepts (antecedents, disorders, treatments, *etc*.) by icons. We show that it can be generalized to other iconic languages, including traffic signs. We also describe four practical applications made possible by the formalization of the language semantics: the verification of icons consistency, the semi-automatic alignment with terminologies, the automatic generation of a pictogram lexicon and the automatic generation of icon labels. The article also presents the VCM ontology, the implementation details of a semantic iconic server with fast response times, and the evaluation results obtained when evaluating the four applications.

Keywords: Icons, Iconic language, Semantics, Ontologies, Alignment, Medicine

1. Introduction

It is well known that "a picture is worth a thousand words". This is why many icons, symbols and pictograms are used [1] in various domains such as Human-computer interfaces on computers, tablets and mobile phones, signage in public places or chemical product labeling. However, one cannot remember an infinite number of signs and, therefore, when it is necessary to represent a high number of concepts, it is not possible to learn a specific icon for each concept. In this case, the solution consists in the design of an *iconic language* with a proper syntax and semantics. It allows the generation of numerous icons by the combination of a limited number of components, such as colors or pictograms, and it can also improve the interpretation of icons [2].

Several iconic languages have been designed for specific domains. Examples include Stabilis [3], a database on the stability and compatibility of injectable drugs, which uses icons for allowing a multilingual access, and OMICtools [4], an informative directory of software and resources for bioinformatics, which uses icons for categorizing the various types of tools. Traffic signs are well-known everyday life examples: these signs are used to give orders (for example, a speed limitation) to drivers or to inform them about dangers (such as dangerous curves), directions or commodities. Traffic signs are composed using a simple iconic language, by assembling several elements such as a red circle, a left-turn pictogram, *etc*. The iconic aspect of this language serves two objectives: the signs must be read quickly and they should be independent from natural languages (for foreigners). Another more complex example is the VCM (Visualization of Concept in Medicine) medical iconic language [5]. It can represent the main medical concepts, such as disorders, risks, treatments, by icons. This iconic language was designed in order to help health professionals to access medical documents. In fact, the volume of medical data and knowledge has grown considerably over the last decades and the extensive reading of patient records, drugs summary of products characteristics (SPCs) or clinical practice guidelines is a very long and difficult task [6].

We designed VCM ten years ago, and various VCM-powered medical applications have been developed, some of them being routinely used. VCM was initially invented for presenting drug knowledge, such as the contraindications and adverse effects of a given drug in a drug database [5, 7]. It was then extended for representing patient disorders and physiological states, with applications to medical search engines [8], clinical guidelines [9] and the visualization of patient health records [10]. During this decade of evolution, the need for semantics has been increasingly pressing as the language grew in size and complexity, and as the applications became more numerous. Users created nonsense icons. Experts started to use the same pic-

*Corresponding author
This is an author file of the article published in Knowledge Based System, DOI: https://doi.org/10.1016/j.knosys.2017.08.011 ; it is available under Creative Commons Attribution Non-Commercial No Derivatives License.
Email addresses: jean-baptiste.lamy@univ-paris13.fr (Jean-Baptiste Lamy), lina.soualmia@litislab.fr (Lina F. Soualmia)

togram with a slightly different meaning. Medical application developers asked for mappings between VCM and the standard terminological resources they used.

The grammar and the semantics of iconic languages are often informal, or at best described textually (as it was originally the case for VCM). This leads to vagueness in the semantics. Additionally, all tasks relative to the semantics of the language must be performed manually, for example for the production of a pictogram lexicon or the writing of icon labels. Moreover, a grammatically correct icon can still be semantically wrong: for example, a traffic sign with the red triangle meaning "attention" and the "snow tire" pictogram is a nonsense. The absence of a logically validated semantics also makes it more difficult to align the icons with other resources such as the existing terminologies or ontologies of the domain. Alignments with terms or concepts must be performed manually by an expert who interprets the icons, but, as any human interpretation, it is potentially subjective. According to Erwig [11], the formalization of graphical languages has four interests: (a) improving the comprehension of the language, (b) facilitating its implementation, (c) performing automatic reasoning on the language, and (d) helping with the integration of the language in other environments.

In this article, we describe a method that formalizes the semantics of an iconic language using a formal ontology, divided in three modules: the *icon ontology*, the *domain ontology* and the *mapping ontology*. This method was originally designed specifically for checking the consistency of VCM [12] and then reused for mapping VCM to a subset of a medical terminology [13]. Here, we present our method in a more detailed fashion by putting the emphasis on the developing steps, including implementation and technological aspects. Additionally, we provide an up-to-date state of the art related to the formalization of graphical languages. We also generalize the method to other iconic languages, and we apply it to traffic signs as a proof of generalizability. Four semantic-based applications on VCM are presented, illustrating the four objectives proposed by Erwig. Based on our experience with VCM, we can hypothesize that formalizing the semantics of iconic languages could improve their coherence, and facilitate their use and their dissemination. More generally, many icon sets in daily-used graphical user interfaces follow some grammatical or semantic rules, and the formalization of their semantics could lead to similar benefits.

The paper is organized as follows: section 2 presents the previous works related to the formalization of graphical languages. Section 3 gives a reminder on Description Logics (DLs) notations. Section 4 presents the formalization method. Section 5 describes the use of this method in the context of traffic signs. Section 6 does the same for VCM, and describes four practical applications made possible by the formal expression of the language semantics: (a) the verification of icon consistency, (b) the semi-automatic alignment of the icons with a domain terminology, (c) the automatic generation of a pictogram lexicon to help the comprehension of the language, and (d) the automatic generation of multilingual icon labels. Finally, section 7 discusses the advantages and the limits of our approach.

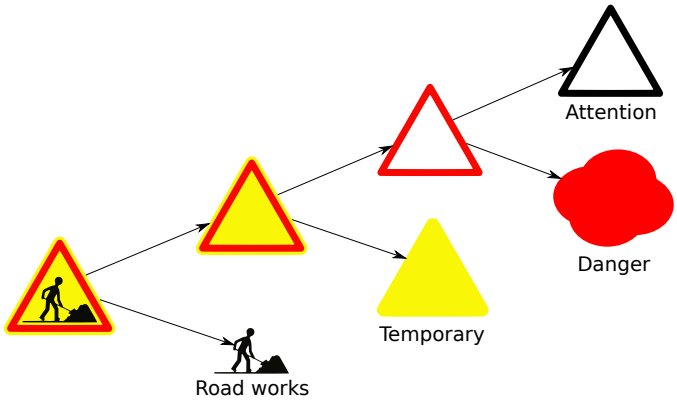

Figure 1: Semiotic structure of the traffic sign meaning "attention road works", according to Meunier [14, 15].

2. Related works on the formalization of graphical languages

Most of the literature on the formalization of graphical languages is about 20-year old, which corresponds to the period of the emergence of graphical user interfaces and visual modeling languages. Today, icons are everywhere on computers, mobile phones and tablets, but paradoxically, less attention is paid to the formalization of their semantics. Recent works focus on graphical design [16] and usability evaluation [17], rather than on the semantics itself. In the medical field, two examples of such works involve the icon design for a user interface of remote patient monitoring mobile devices [18], and for creating icons for an emergency medical information system, using participatory design [19]. Several icon taxonomies were proposed, for instance according to the iconic representation strategy [20] or the objective/context in which the icons are used [21]. Other recent works try to understand how icons are perceived by the brain, for example a neuro-imaging study showed that icons stimulate the semantic system in the brain, but they are cognitively processed as pictures rather than words [22]. In the rest of this section, we focus on studies related to the semantics of icons.

Historically, graphical languages were first studied from a semiologic point of view [14, 15]. These studies typically consisted in breaking down graphical signs such as traffic signs (see Figure 1), in order to associate with each component (color, pictogram, *etc.*) one aspect of the meaning of the global sign. While easy to understand, this approach did not allow a true formalization of iconic languages.

Then, another kind of works emerged with the design of grammar for visual languages. Several approaches were proposed, including *constraint multiset grammars* [23] which were inspired by Chomsky's grammars, *positional grammars* [24] which were based on a classical textual grammar complemented by a position evaluator in charge of the translation of the textual sentences into graphics. These grammars were used for writing parsers and compilers. These approaches were in particular applied to diagrammatic visual languages such as visual programming or querying languages, where a computer program or a database query is expressed by a diagram. Zolotas

		Syntax	Description	Semantics
Constants		$\top$	Thing / Top	$\Delta^{\mathcal{I}}$
		$\bot$	Nothing / Bottom	$\emptyset$ (empty set)
Axioms	TBox	$A \sqsubseteq B$	A is subsumed by B	$A^{\mathcal{I}} \subseteq B^{\mathcal{I}}$
		$A \equiv B$	A is equivalent to B	$A^{\mathcal{I}} = B^{\mathcal{I}}$
	ABox	$A(i)$	i is an instance of A	$i^{\mathcal{I}} \in A^{\mathcal{I}}$
		$R(i, j)$	i and j are related by role R	$(i^{\mathcal{I}}, j^{\mathcal{I}}) \in R^{\mathcal{I}}$
Constructors		$\neg A$	Negation of concept A	$\Delta^{\mathcal{I}} \backslash A^{\mathcal{I}}$
		$A \sqcap B$	Intersection of A and B	$A^{\mathcal{I}} \cap B^{\mathcal{I}}$
		$A \sqcup B$	Union of A and B	$A^{\mathcal{I}} \cup B^{\mathcal{I}}$
		$\exists R.B$	Exists restriction	$\{a \in \Delta^{\mathcal{I}} \mid \exists b, (a, b) \in R^{\mathcal{I}} \text{ and } b \in B^{\mathcal{I}}\}$
		$\forall R.B$	Universal restriction	$\{a \in \Delta^{\mathcal{I}} \mid \forall b, (a, b) \in R^{\mathcal{I}} \rightarrow b \in B^{\mathcal{I}}\}$
		R^{-}	Role inverse	$\{(a, b) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid (b, a) \in R^{\mathcal{I}}\}$
Disjoint axiom		$A \sqcap B \sqsubseteq \bot$	A and B are disjoint	$A^{\mathcal{I}} \cap B^{\mathcal{I}} = \emptyset$

Table 1: The syntax and the semantics of a DLs knowledge base $\mathcal{O}$. A and B are concepts, R is a role, i and j are individuals. Disjoint axiom has been added for convenience, it is inferred by combining intersection and subsumption.

et al. [25] proposed to extend modeling diagrams with iconic attributes such as shape, color or position, and to associate a domain-specific semantic to these attributes. The authors proposed several examples, such as modeling football players and using colors to identify teams. While this "mix" of diagram and icons is interesting, the associated semantics is not formalized.

A US patent targeted pluggable notations and semantics for modeling diagrams [26]. The system associates semantic objects with notation objects. The proposed application was the interoperability between several visual or non-visual languages, *e.g.* designing a modeling diagram in UML (Unified Modeling Language) and then switching to another notation such as C++ class definitions, while keeping the same semantics.

Another approach consisted of transforming the expressions of a graphical language into oriented labeled graphs, without the definition of the grammar, thanks to the use of an *abstract visual syntax* [11, 27]. The resulting graph could then be used for expressing the semantics and for computing proofs. In particular, abstract visual syntaxes were defined for graphical languages deriving from lambda calculus.

Finally, several approaches tried to describe graphical languages from a lexical and syntactical point of view. The first one used Description Logics (DLs) to formalize logical definitions such as "an inheritance relation is a line that connects the boxes of two class and that has a triangle at one of its extremities" [28]. Another approach used UML schema for defining the concrete syntax of a visual language [29]. The correctness of the syntax definition could be proved using first-order logic, ensuring that a given diagram corresponds to only one model. More recently, an approach based on the local context was proposed [30], targeting schematic graphical languages. It allowed a lexical and syntactic definition of the language, by defining the graphical elements of the language (boxes, arrows, *etc*.) and the possible connections between them (including minimal and maximal cardinalities, *etc*.). These two approaches were interesting for describing graphical languages but they were limited for the expression of semantics.

Kuicheu *et al.* proposed an ontology of icons named IcOnto [31], aimed at formalizing icons using DLs. The authors defined an icon as the intersection of a physical part, the image itself described by graphical attributes such as color and shape, and a logical part, the set of attributes or characteristics commonly associated with the icon (*e.g.* an emotion, an activity, a number, a sound). The ontology allowed the inference of attributes according to subsumption relations, but the reasoning capabilities seems limited beyond this simple example.

To conclude this section, several approaches were proposed in the literature about the formalization of graphical languages. Many of them relied on grammars, event for the expression of the semantics. However, most of the proposed approaches targeted schematic or diagrammatic languages such as UML rather than icons or iconic languages.

3. Description Logics (DLs)

In this section, we provide a reminder of DLs; for a more detailed presentation the reader should refer to the literature [32].

Description Logics are a well-known family of knowledge representation formalisms. They are fragments of first-order predicate logic and are equipped with a formal, logic-based semantics. An ontology $\mathcal{O}$ can be defined as a DLs knowledge base $\mathcal{O} =< \mathcal{T}\ \mathcal{A} >$ composed by a set of logic axioms Φ. The axioms are built using a set of individuals $\mathbb{I} = \{i, j, ...\}$, a set of concepts $\mathbb{C} = \{C, D, ...\}$, a set of roles $\mathbb{R} = \{R, S, ...\}$, and a set of constructors $\mathbb{S}$. Four types of axioms are considered: $C \sqsubseteq D$ (subsumption), $C \equiv D$ (equivalence), $C(i)$ (instantiation) and $R(i, j)$ (relation). Constructors are used for combining concepts and/or roles (depending on the constructor) and defining a new concept or role. The set of constructors depends of the DLs family considered, in this paper we will use the DL $\mathcal{ALCI}$ in which $\mathbb{S} = \{\neg, \sqcap, \sqcup, \forall, \exists, R^{-}\}$. Table 1 lists the 4 types of axioms and the 6 constructors. The set of axioms of an ontology $\mathcal{O}$ is

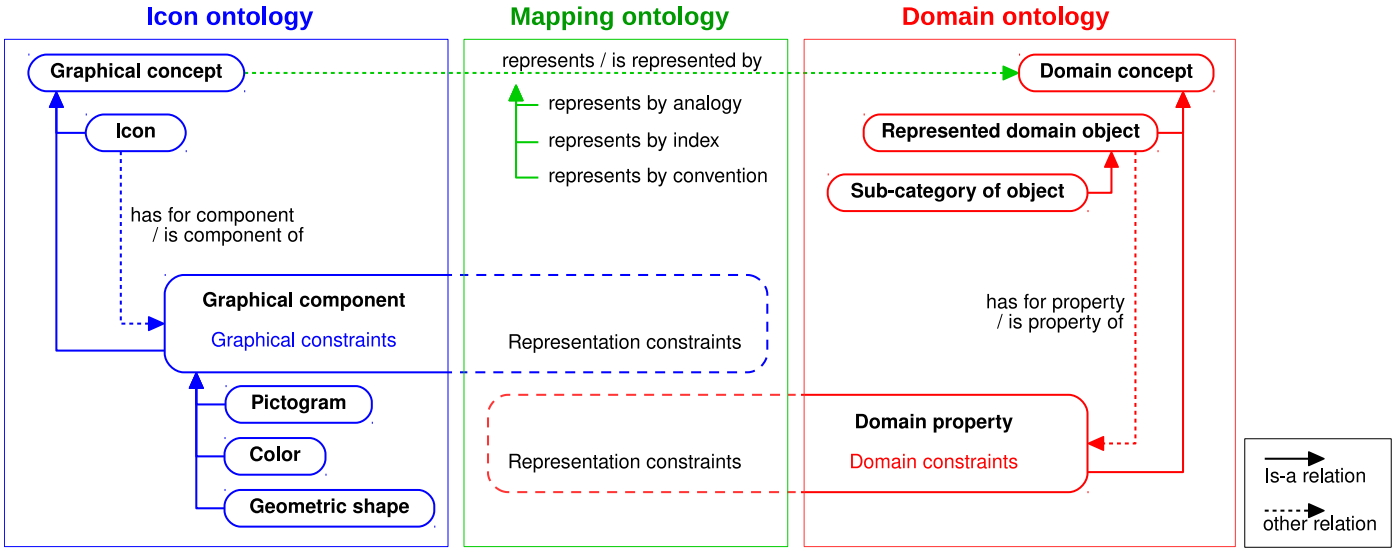

Figure 2: General structure of the icon ontology (in blue on the left), the mapping ontology (in green in the middle) and the domain ontology (in red on the right).

divided into a TBox $\mathcal{T}$ (Terminological Box, the set of concepts and roles axioms) and an ABox $\mathcal{A}$ (Assertional Box, the set of individuals axioms). In this paper, we will only use the TBox part.

DLs have a model-theoretic semantics, which is defined in terms of interpretations. For a given ontology $\mathcal{O}$, an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, .^{\mathcal{I}})$ consists of the domain of the interpretation $\Delta^{\mathcal{I}}$ (a non-empty set) and the interpretation function $.^{\mathcal{I}}$, which maps each concept $C \in \mathbb{C}$ to a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, each role $R \in \mathbb{R}$ to a binary relation $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ and each individual $i \in \mathbb{I}$ to an object in the domain $i^{\mathcal{I}} \in \Delta^{\mathcal{I}}$.

The last column of Table 1 shows the interpretation associated with each axiom and constructor. Using the interpretation function, the logical axioms of the ontology can be transformed into set formula, which express the semantics of the ontology. For instance, the axiom $A \sqsubseteq B \sqcap C$ is translated into $f(A \sqsubseteq B \sqcap C) = A^{\mathcal{I}} \subseteq B^{\mathcal{I}} \cap C^{\mathcal{I}}$.

An interpretation $\mathcal{I}$ satisfies an ontology $\mathcal{O}$ if it satisfies all axioms in $\mathcal{O}$ (*i.e.* $\forall \Phi \in \mathcal{O}, \Phi^{\mathcal{I}}$ is true). An ontology $\mathcal{O}$ is *consistent* if there exists at least one interpretation $\mathcal{I}$ that satisfies $\mathcal{O}$ ($\mathcal{O}$ is *inconsistent* otherwise). A concept C is *satisfiable* in $\mathcal{O}$ if (and only if) there exists an interpretation $\mathcal{I}$ that satisfies $\mathcal{O}$ such that $C^{\mathcal{I}} \neq \emptyset$ (*i.e.* there may exists an individual i that belongs to C).

For a given ontology $\mathcal{O}$, $\Phi \in \mathcal{O}$ means that the axiom Φ belongs to the set $\mathcal{O}$ (*i.e.* the axiom has been *asserted*), and $\mathcal{O} \models \Phi$ means that the axiom Φ can be *inferred* from the axioms in $\mathcal{O}$. The simple transitivity between subsumption relations is not considered as an inference (*e.g.* if $\mathcal{O} = \{A \sqsubseteq B, B \sqsubseteq C\}$ we will also consider that $A \sqsubseteq C \in \mathcal{O}$). In fact, indirect is-a relations can be easily computed and a DLs reasoner is not mandatory for this task.

4. Method for the formalization of the semantics of iconic languages

4.1. General principles

The method we present here aims at formalizing the semantics while keeping a clear distinction between the syntactic and the semantic aspects, in order to prevent confusion between graphical objects (the icons and their components: pictograms, colors, *etc.*, for example the ♡ pictogram) and the objects of the application domain that they represent (for example, the heart organ). Therefore, we chose to represent in two distinct ontologies the syntax of the iconic language (*icon ontology*) and the associated semantics (*domain ontology*), and then to link them together in a third ontology (*mapping ontology*) that imports the two previous ones.

Our method for the formalization of the semantics of iconic languages follows three steps:

1. Designing the *icon ontology* that describes the syntax of the icons of the language and their components.
2. Designing the *domain ontology* that describes the objects represented by the icons; this ontology should be built on existing termino-ontological resources.
3. Designing the *mapping ontology* that integrates the two other ontologies by establishing links and constraints between the concepts of the icon and the domain ontologies, using the two following relations: "<Graphical concept> represents <Domain concept>" and "<Domain concept> is-represented-by <Graphical concept>".

Figure 2 shows the general structure of the three ontologies. Description Logics is a well-known family of knowledge representation formalisms for ontologies and we chose to formalize the three ontologies using this language family. The icon ontology defines Graphical concepts, including the icons and the Graphical components that are combined to create the

icons. Several subcategories of Graphical component are typically considered, such as colors, pictograms and geographic shapes. For example, the "attention road works" road sign (Figure 1) can be decomposed in four Graphical components: the road work pictogram, the triangle shape, the yellow background color and the red border color. In addition, *graphical constraints* can be stated in the icon ontology, for example: "a road sign includes at most one pictogram".

The domain ontology describes the Domain concepts. Typically, various domain objects are represented by icons (Represented domain object on Figure 2); these objects can be organized in subcategories using subclasses and *is-a* relations in the ontology. Represented domain objects are commonly described by Domain properties. For example, "attention road works" can be described as an information regarding danger, having for topic road works, and whose duration is temporary. *Domain constraints* can be stated in the domain ontology, for example: "road works are something drivers could encounter while driving, but not something they can do themselves" (and thus consequently it does not make sense to prohibit a driver from doing road works, or to order him to do so).

The mapping ontology defines the represent/is-represented-by relations between Graphical components and Domain properties. These relations can be used to define *representation constraints*. In the previous example, the "man at work with a shovel" pictogram is only present on traffic signs that represent instructions about road works, and road works are only associated to instructions that are represented by traffic signs including the "man at work with a shovel" pictogram.

Ontologies allow to define in ontology B a constraint on an object defined in another ontology A. Consequently, despite the representation constraints affect Graphical components and Domain properties, they can be asserted in a separate ontology, the mapping ontology. This modeling choice has two advantages: (1) it allows a clear distinction between Graphical concepts and Domain concepts in the first two ontologies (while representation constraints involve both concepts), and (2) it makes it easier to generate the representation constraints from another non-ontological source, such as a text file (see section 4.2 below).

In the mapping ontology, more specific relations inheriting from "represent / is-represented-by" (*i.e.* sub-properties) can be used to distinguish the different strategies for visual representation. For instance, most authors consider three categories of representation: (1) "true icon" that looks like the represented object and relying on analogy (*e.g.* the "♡" sign, which has the same shape as the heart organ), (b) indexes that are associated with the object (*e.g.* a picture of a floppy disk for the "save" action) and (c) symbols that are arbitrary or conventional associations (*e.g.* "H" for "hospital"). The last category can be subdivided further in already well-known symbols and new symbols.

Consequently, constraints are formulated at three levels: *graphical constraints* in the icon ontology, *domain constraints* in the domain ontology, and *representation constraints* in the mapping ontology.

Representation constraints allow the propagation of graphical constraints towards the domain ontology, and of domain constraints towards the icon ontology. In fact, graphical constraints can have an impact on the domain represented by the language and in particular there may be some domain concepts that cannot be represented by icons. Similarly, domain constraints can have graphical consequences and make some icons inconsistent. Our formalization method allows not only to describe the syntax and the semantics of iconic languages, but it also makes possible several practical applications using a reasoner engine for propagating the constraints and deducing the semantics of a given icon.

4.2. Managing the mapping in the mapping ontology

It is natural to express the mapping as a set of triples, *e.g.* (Graphical component A, represents, Domain concept A). However, the logical constraints in the mapping ontology are more complex than simple triples, because one constraint is required for each Graphical component and Domain concept. Thus, the previous example of triple actually leads to two constraints. Moreover, if a Graphical component or a Domain concept is involved in more than one triple, they should be combined in a single constraint. Consequently, authoring the constraints manually would be a tedious task. We propose to design and manage the mapping using a combination of the three following file formats: (1) a text file containing triples, allowing the authoring of the mapping easily, (2) an OWL file containing the mapping ontology, *i.e.* the representation constraints, allowing automatic reasoning, (3) an SQL database, allowing fast querying on the triples.

The last two formats can be produced automatically from the text file. Triples can be stored in an SQL database[1], and we used Python scripts for generating constraints from triples. As the representation constraints are located in a separate ontology (the mapping ontology), it is easy to delete the corresponding OWL file and to generate a new one whenever the text file is modified.

Table 2 shows examples of triples and the produced constraints, for each of the three possible typical cases. Example #1 is a simple one-to-one mapping between a Graphical component and a Domain concept. Example #2 is a case of polysemy, *i.e.* a Graphical component that can represent several Domain properties (although, when included in a given icon, depending on the other components of the icon, the polysemy may remain or not). Polysemy is frequent in iconic languages. Example #3 is a case of a Graphical component that represents the intersection of several Domain properties (noted with "+" in the text file). This occurs when a specific pictogram is designed for a combination of Domain properties. In VCM, obstruction disorders are some examples. They are represented graphically by adding an obstruction in the organ, but the position of the obstruction differs for each organ, thus it is not possible to follow a combinatory approach (*i.e.* obstruction pictogram + organ pictogram) and a specific pictogram must be designed for

[1] From a technical point of view, an RDF triplestore can be used for storing these triples. However, they are not proper RDF triples since they involve classes and not individuals. Therefore, these triples should not be inserted in the ontology or mixed with it – they need to be transformed into constraints before being inserted in the ontology.

#	Triples	Constraints
1	(g_A, represents, d_A)	$g_A \sqsubseteq (\forall is\_component\_of.(\forall represents.(\exists has\_for\_property.d_A)))$ $d_A \sqsubseteq (\forall is\_property\_of.(\forall is\_represented\_by.(\exists has\_for\_component.g_A)))$
2	(g_A, represents, d_A) (g_A, represents, d_B)	$g_A \sqsubseteq (\forall is\_component\_of.(\forall represents.(\exists has\_for\_property.(d_A \sqcup d_B))))$ $d_A \sqsubseteq (\forall is\_property\_of.(\forall is\_represented\_by.(\exists has\_for\_component.g_A)))$ $d_B \sqsubseteq (\forall is\_property\_of.(\forall is\_represented\_by.(\exists has\_for\_component.g_A)))$
3	(g_A, represents, d_A+d_B)	$g_A \sqsubseteq (\forall is\_component\_of.(\forall represents.(\exists has\_for\_property.(d_A \sqcap d_B))))$ $d_A \sqsubseteq (\forall is\_property\_of.(\forall is\_represented\_by.(\exists has\_for\_component.g_A)))$ $d_B \sqsubseteq (\forall is\_property\_of.(\forall is\_represented\_by.(\exists has\_for\_component.g_A)))$

Table 2: Examples of mapping triples and the constraints they generate, for each of the three possible typical cases. "g" stands for "Graphical component" and "d" for "Domain concept".

each organ. Each of these pictograms, *e.g.* "obstructed blood vessel", represents the intersection of the obstruction alteration and the organ.

Notice that all constraints focus on the represent/is-represented-by relation between the icon and the corresponding Represented domain object, and not directly between the Graphical component and the corresponding Domain property (*e.g.* for the first constraint in Table 2, $g_A \sqsubseteq (\forall is\_component\_of.(\forall represents.(\exists has\_for\_property.d_A)))$ and not $g_A \sqsubseteq (\forall represents.d_A)$). While the latter would be simpler, it would not permit automatic reasoning with OWL reasoners. In fact, it would require several free variables (for asserting the following: I represents $J \Rightarrow \forall g \in I, \exists d \in J \mid g$ represents d), while DLs are variable-free and actually correspond to formula with a single free variable [32].

Subsumption is also taken into account : if g_A and $g_{A'}$ are two Graphical components with $g_{A'} \sqsubseteq g_A$, we assume that the triple ($g_{A'}$, represents, d_A) implies (g_A, represents, d_A).

4.3. Formal definition of the icon semantics

In this paper we consider that an icon I is represented by a set of Graphical components, such as pictograms, geometric shapes and colors, and the meaning M (or the semantics) of an icon is represented by a set of Domain concepts (Algorithm 1). On the contrary, any set of Graphical components or Domain concepts does not correspond to a valid icon or meaning, respectively, until the previously defined constraints are satisfied. If some pictograms can be present at several different positions on the icon, each valid (pictogram, position) pair should be considered as a distinct Graphical component.

We also define the get_meaning(I) function, which returns the meaning of a given icon, *i.e.* the set M of the Domain concepts associated with the icon I. The function first uses the mapping triples to compute, for each Graphical component, the set of Domain concepts it can represent. Each Graphical component represents one Domain concept, or several in case of polysemy. D is the tuple of all these sets. Then we compute P, the unordered Cartesian product of all sets in D. Elements in P are the candidate interpretation for I (*i.e.* a choice between the various possible polysemes). Next, P is filtered for removing inconsistent sets of Domain concepts (*i.e.* sets whose conjunction is inconsistent when taking into account graphical, domain and representation constraints), yielding P'. Finally, the resulting meaning M is the union of all sets in P'. Examples will be provided later.

Figure 3: Examples of signs that can be created using French traffic signs as an iconic language. From left to right: attention road works, prohibition to turn left, mandatory turn left, prohibition of road works. This last sign is an example of an inconsistent icon (and thus no such traffic sign exists in real life), because drivers are not supposed to make road works.

In the next section, we apply the presented method to a simple example: traffic signs. In the section that follows, we apply it to the VCM medical iconic language and we describe four possible applications.

5. Application to traffic signs

5.1. Traffic signs

Traffic signs correspond to a simple and well-known iconic language; they allow visually giving instructions to drivers. In this iconic language, an icon is a traffic sign. We describe here the French signs; small variations exist between countries but the general principles remain the same. Four main categories of sign can be distinguished: prohibition signs (with a red circle, sometimes with a cross), mandatory signs (with a blue circle), signs informing about an upcoming danger (red triangle) and signs conveying other information (square). Temporary signs have a yellow background while permanent signs have a white background. Finally, a central pictogram gives the topic of the sign.

A grammar allows combining the external shape, the background and the pictogram together (see Figure 3). However, some combinations are senseless and semantic rules are required, in addition to grammatical rules, to avoid the creation of inconsistent signs. For example, the "left turn" pictogram can be present on both mandatory signs ("mandatory turn left"), prohibition signs ("prohibition to turn left") or danger signs

Algorithm 1 Icon and meaning formal definitions and interpretation.

Let us denote:

- O_{icon} the icon ontology
- O_{domain} the domain ontology
- $T_{mapping}$ the set of mapping triples, $T_{mapping} = \{(g, \text{represents}, d)\}$ where $g \sqsubseteq Graphical\_component \in O_{icon}$ and $d \sqsubseteq Domain\_concept \in O_{domain}$
- $O_{mapping}$ the mapping ontology, with constraints between $Domain\_concept$ in O_{domain} and $Graphical\_component$ in O_{icon}
- I an icon, $I \subseteq \{g \sqsubseteq Graphical\_component \in O_{icon}\}$
- M the meaning of an icon, $M \subseteq \{d \sqsubseteq Domain\_concept \in O_{domain}\}$

function get_meaning(I):

$D = \{\{d \mid d \sqsubseteq Domain\_concept \in O_{domain} \text{ and } (g, \text{represents}, d) \in T_{mapping}\} \mid g \in I\}$

$P = \{\{d_1, ..., d_{|I|}\}, \forall (d_1, ..., d_{|I|}) \in \prod D\}$ (*i.e.* P is the unordered Cartesian product of the sets in D)

$P' = \{\{d_1, ..., d_{|I|}\} \in P \mid d_1 \sqcap ... \sqcap d_{|I|} \text{ is satisfiable with regard to } O_{domain} \cup O_{mapping} \cup O_{icon}\}$

$M = \bigcup P'$

return M

("pay attention left turn"). On the contrary, the "man at work with a shovel" pictogram can only be present on danger signs ("pay attention road works"), but not on mandatory or prohibition signs (see the last signs on Figure 3). Moreover, road works are temporary by nature, and thus they cannot be associated with the white background (corresponding to permanent signs).

5.2. Formalization of the semantics of traffic signs

We described traffic signs with an ontology (*icon ontology*), as well as the associated driving instructions (*domain ontology*), and we related them (*mapping ontology*). This leads to the ontology shown on Figure 4. In the domain ontology, we distinguished two main categories of driving instructions: orders and information. Orders include obligations and prohibitions. Information includes information about danger and other information. Each instruction is associated to a temporality (permanent or temporary) and a topic, which can be: (a) an *action of the driver* (for example turn left, drive at 50 km/h, pass another vehicle, drive a tractor, *etc.*), or (b) a *possible encounter* involving an element exterior to the driver and his car (for example a left turn, a hospital, wild animals, *etc.*). Obviously, orders can only be associated with actions of the driver (it is a nonsense to order the driver to encounter something), and information with encounters (traffic signs are not there for informing the driver about his own actions). We translated that using two domain constraints.

5.3. Examples and practical applications

The ontology we presented allows the formalization of the semantics of traffic signs. By associating the various elements composing road signs, it is possible to build the existing signs but also to generate new ones. A first application consists in determining the consistency and the semantics of these signs.

Figure 5 shows the description of the "prohibition to turn left" sign. We can note that a given Graphical component (here, the "left arrow" pictogram) can represent several concepts in the domain ontology (here the "left turn" encounter and the "turn left" action). This type of ambiguity is frequent in iconic languages, because icons are often less precise than a text (or an ontology).

The previously defined get_meaning(I) function (Algorithm 1) can be used to obtain the set of Domain concepts M, for a traffic sign I. For example, the "prohibition to turn left" sign can be formalized as:

I = { Circle, Red border, White background,
Left arrow pictogram }

Then, the function creates D, the tuple of the sets of Domain concepts represented by each Graphical component in I. *Is-a* relations are taken into account, *e.g.* the Circle Graphical component can represent the Order Domain concept, but also its two descendants, Prohibition and Obligation.

D = ({ Order, Prohibition, Obligation },
{ Danger, Prohibition, Information of danger },
{ Permanent },
{ Left turn encounter, Turn left action })

P is the set of candidate interpretations for I, and corresponds to the Cartesian product of the sets in D.

P = { { Prohibition, Turn left action, Permanent },
{ Order, Danger, Turn left action, Permanent },
{ Order, Danger, Left turn encounter, Permanent },
{ Obligation, Danger, Turn left action, Permanent },
... (18 elements in P) }

P' is the subset of consistent interpretations in P. In the above example, the third set is inconsistent because an Encounter cannot be associated with an Order in the domain ontology, and the last set is inconsistent because an Obligation cannot be associated with a Danger.

P' = { { Prohibition, Turn left action, Permanent },
{ Order, Danger, Turn left action, Permanent },
... }

Finally, the meaning M is the union of all sets in P'.

M = { Prohibition, Order, Danger, Turn left action,
Permanent }

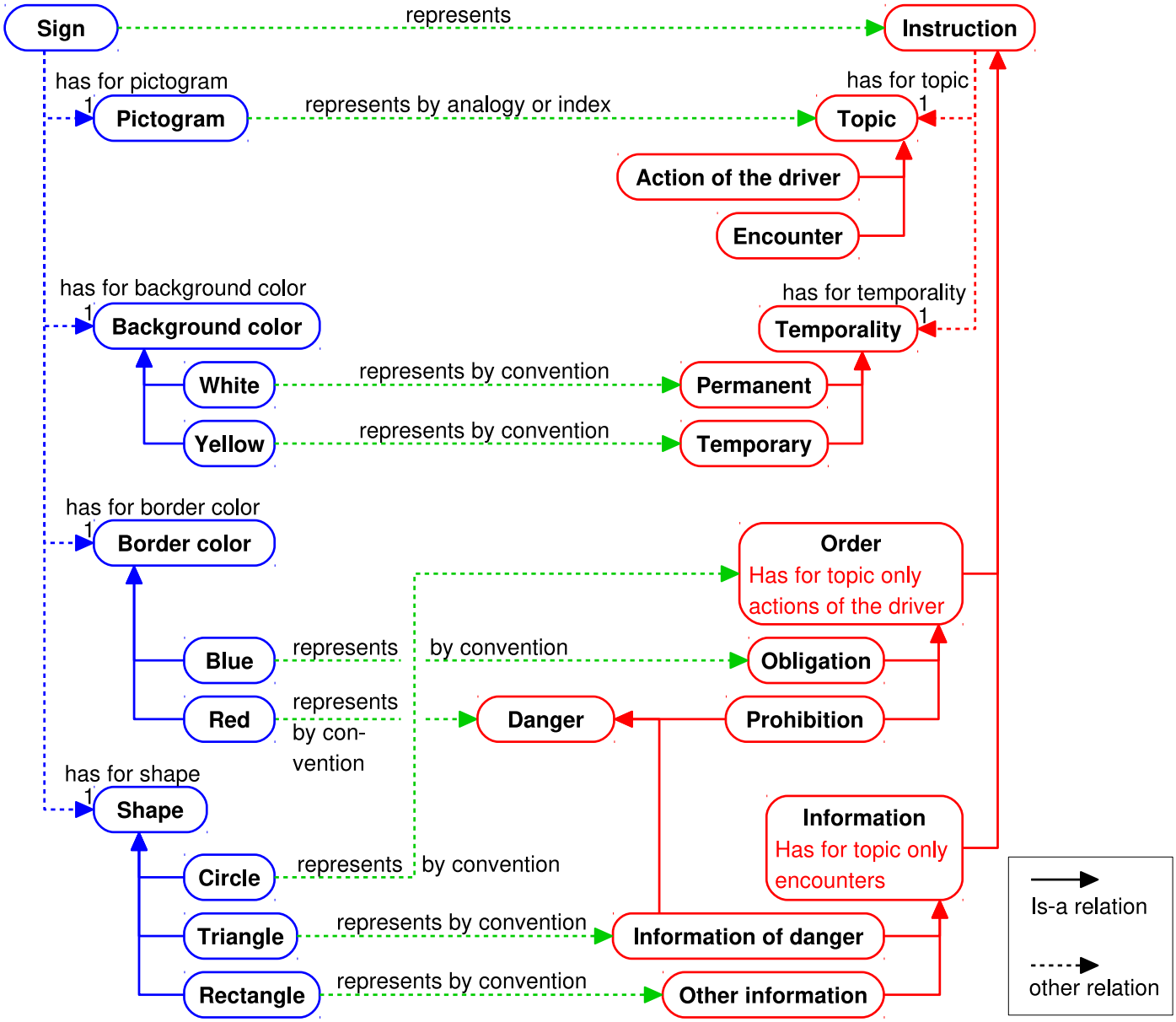

Figure 4: Icon ontology for traffic signs (in blue on the left) and the associated domain ontology (in red on the right).

When an inconsistent sign is described, such as the "prohibition of road works" sign (see Figures 6 and 7), the constraints that are present in the ontology allow inferring the inconsistency. The inference can be automatized using a reasoner, as we will show with VCM (section 6).

Traffic signs have often been used as a source of inspiration for designing other icons or iconic languages, outside the traffic domain. An example is the "pay attention danger: do not drive" icon that figures on some French drug boxes (Figure 8). A second application of the ontology is to verify that these derivative icons are complying with the original semantics of road signs.

The semantics obtained with the ontology for the "pay attention danger: do not drive" icon is not the expected one. In fact, in traffic signs, the triangle shape represents information about an upcoming dangerous encounter. Therefore, the meaning of the icon becomes: "pay attention danger: risk of encountering cars"!

A third application consists of improving the consistence of traffic signs. For example, in France, some prohibition signs have a cross whereas others do not have. Some signs share a common semantic elements, represented visually in a similar way but not identically: for instance, the left arrow differs on the "prohibition to turn left" and the "mandatory turn left" signs (see the second and the third signs on Figure 3).

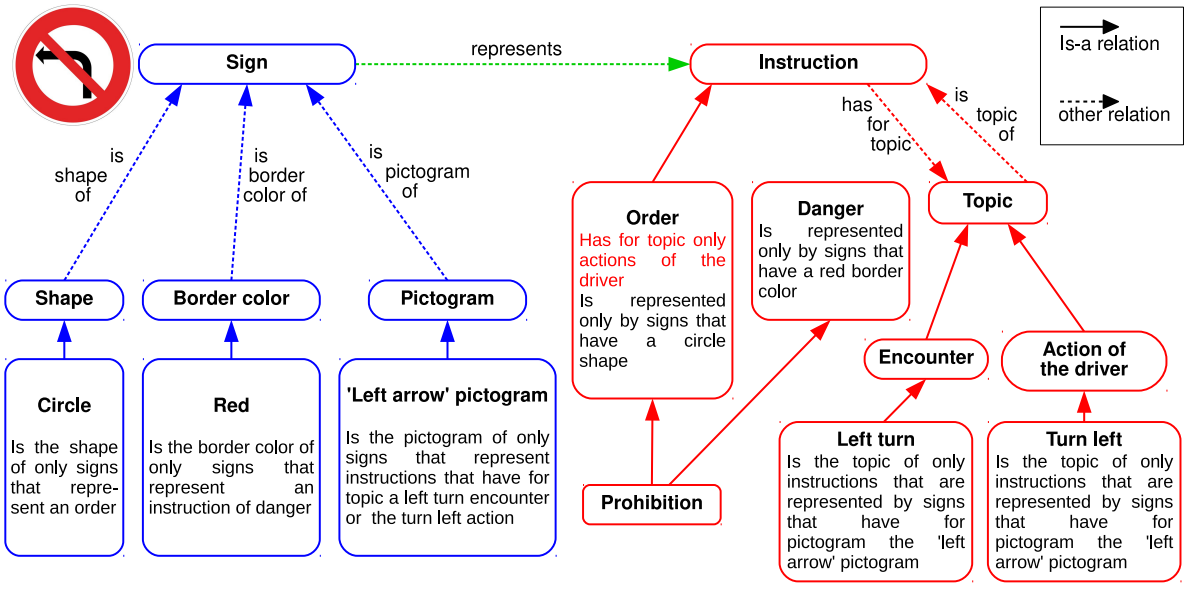

Figure 5: Representation of "prohibition to turn left" in the traffic sign ontology and the associated semantics.

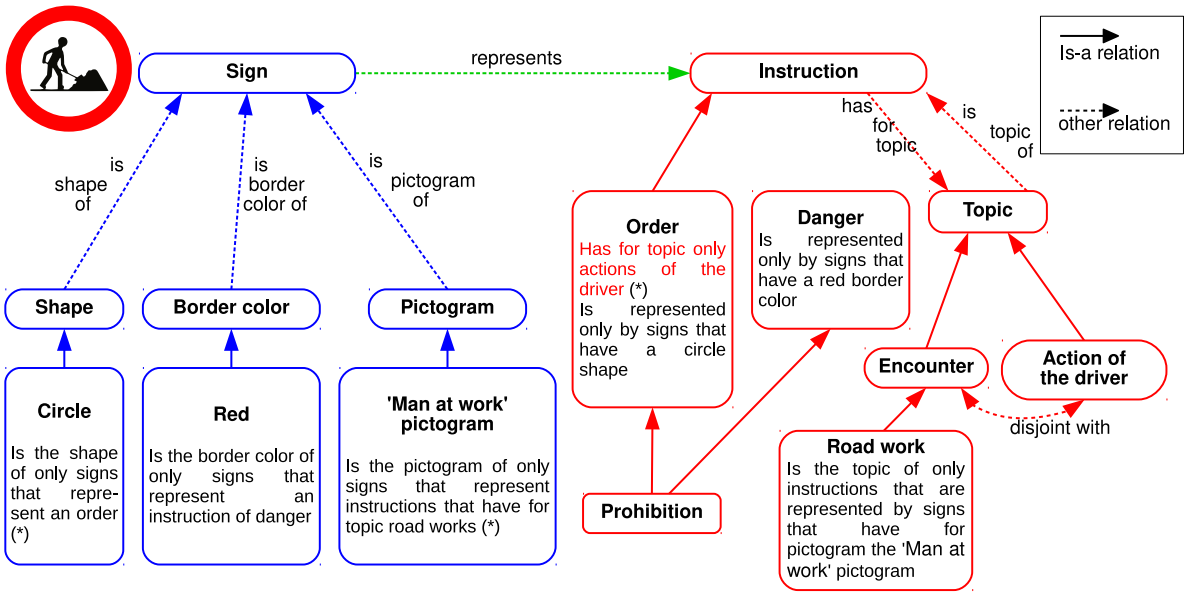

Figure 6: Representation of the inconsistent sign "prohibition of road works" in the traffic sign ontology and the associated semantics. The three constraints marked by a star (*) are the ones that allow inferring the inconsistency of the sign. Two representation constraints allow deducing that this sign represents an order dealing with road works. However, a domain constraint states that orders can be applied only to actions of the driver, and not to encounters such as road work.

$$
\begin{aligned}
\text{Prohibition\_of\_road\_works} \equiv\ & \text{Icon} \\
\sqcap\ & \forall\, \mathit{has\_shape}.\text{Circle} \\
\sqcap\ & \forall\, \mathit{has\_border\_color}.\text{Red} \\
\sqcap\ & \forall\, \mathit{has\_pictogram}.\text{Man\_at\_work\_pictogram} \\
\text{Circle} \equiv\ & \text{Shape} \\
\sqcap\ & \forall\, \mathit{is\_shape\_of}.(\text{Icon} \sqcap \forall\, \mathit{represents}.\text{Order}) \\
\text{Red} \equiv\ & \text{Color} \\
\sqcap\ & \forall\, \mathit{is\_border\_color\_of}.(\text{Icon} \sqcap \forall\, \mathit{represents}.\text{Danger}) \\
\text{Man\_at\_work\_pictogram} \equiv\ & \text{Pictogram} \\
\sqcap\ & \forall\, \mathit{is\_pictogram\_of}.(\text{Icon} \sqcap \forall\, \mathit{represents}.(\text{Instruction} \sqcap \exists\, \mathit{has\_topic}.\text{Road\_works})) \\
& \\
\text{Order} \equiv\ & \text{Instruction} \\
\sqcap\ & \forall\, \mathit{has\_topic}.\text{Action\_of\_the\_driver} \\
\sqcap\ & \forall\, \mathit{is\_represented\_by}.(\text{Icon} \sqcap \exists\, \mathit{has\_shape}.\text{Circle}) \\
\text{Danger} \equiv\ & \text{Instruction} \\
\sqcap\ & \forall\, \mathit{is\_represented\_by}.(\text{Icon} \sqcap \exists\, \mathit{has\_border\_color}.\text{Red})) \\
\text{Prohibition} \equiv\ & \text{Order} \\
\sqcap\ & \text{Danger} \\
\text{Road\_works} \equiv\ & \text{Encounter} \\
\sqcap\ & \forall\, \mathit{is\_topic\_of}.(\text{Instruction} \sqcap \forall\, \mathit{is\_represented\_by}. \\
& \qquad (\text{Icon} \sqcap \exists\, \mathit{has\_pictogram}.\text{Man\_at\_work\_pictogram})) \\
\text{Encounter} \sqcap \text{Action\_of\_the\_driver} \equiv\ & \bot
\end{aligned}
$$

Figure 7: A sub-part of the TBox related to the "Prohibition of road works" traffic sign, corresponding to the example given in Figure 6. This icon is inconsistent according to the concepts definitions in the TBox.

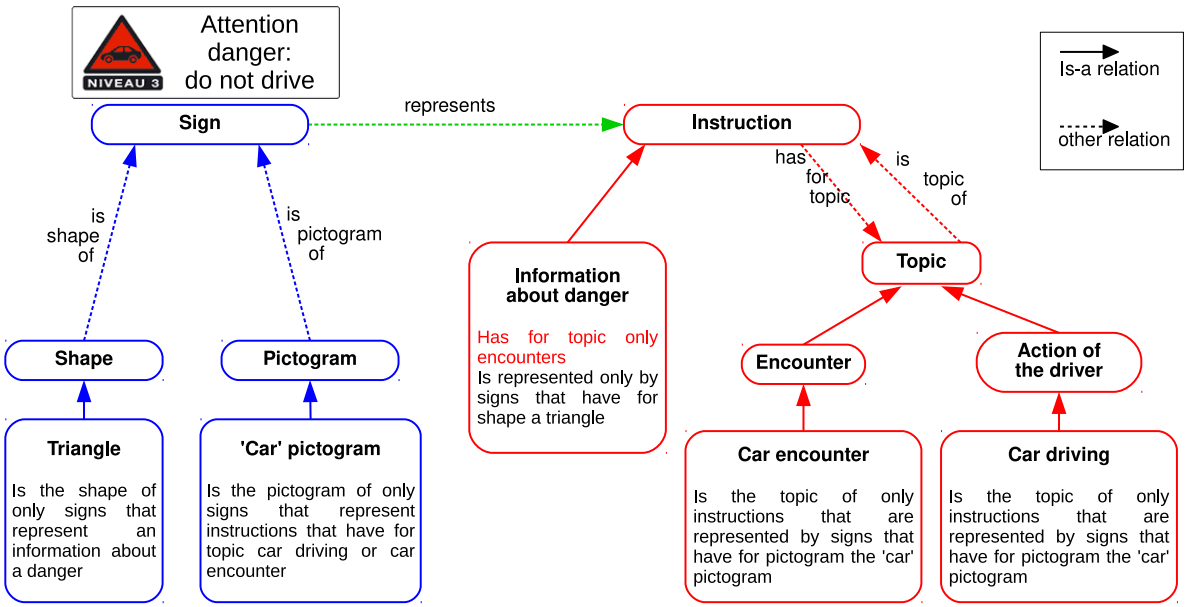

Figure 8: The "pay attention danger: do not drive" icon that figures on some French drug boxes and its representation in the traffic sign ontology. This representation leads to a different meaning for the icon: "pay attention: risk of encountering cars", because the triangle shape is normally associated with information of an upcoming dangerous encounter, but not with a prohibition.

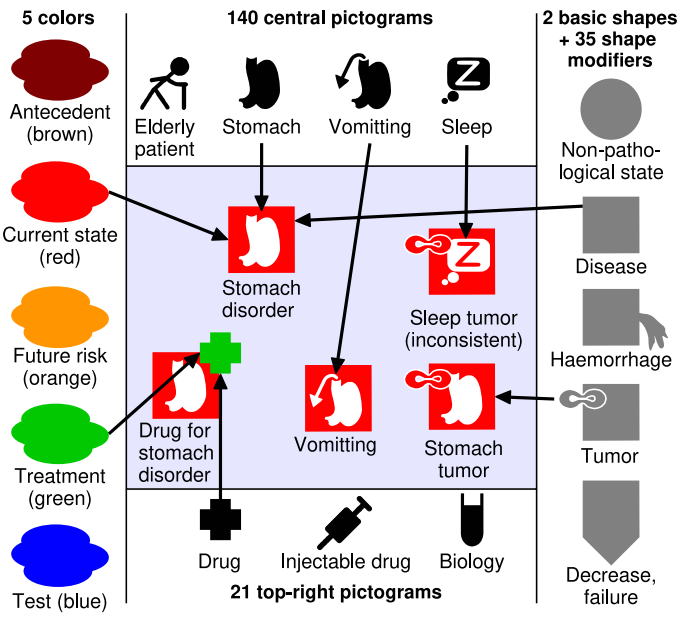

Figure 9: Creation of VCM icons by the combination of various graphical elements.

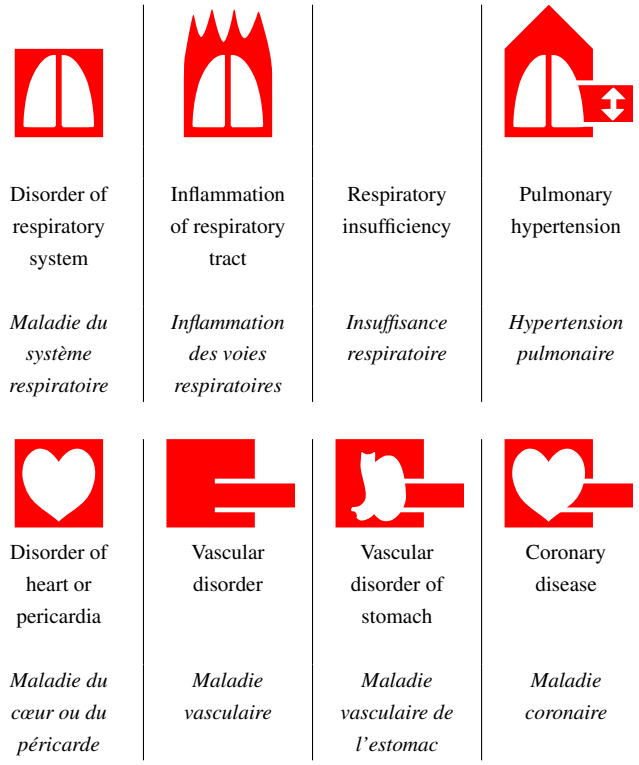

Figure 10: Examples of VCM icons with their corresponding English and French labels.

6. Application to the VCM medical iconic language

6.1. The VCM language

The VCM language [5] proposes icons to represent a patient's main clinical conditions (including symptoms, disorders and physiological states, such as age classes or pregnancy), risk and history of disorders, use of drug and non-drug treatments, laboratory tests and follow-up procedures. It aims to complement medical texts (and not replace them) by highlighting pieces of text or helping physicians to find the desired part of the text. VCM includes a set of graphical components (shapes, pictograms and colors), and uses graphical language to combine these elements and create icons. A training software is available online at the following address: http://vcm.univ-paris13.fr/content/vcm-tutorial.

A VCM icon can be described by a color, a basic shape and a set of shape modifiers, a central pictogram, a top-right color and one or two top-right pictograms; Figure 9 illustrates the graphical combinations of these elements and 10 shows examples of icons. A simple icon can be created by combining:

1. A color indicating the temporal aspect of the icon: red for current states of the patient, orange for risks of future states and brown for past states (such as antecedents or history);
2. A basic shape: a circle for physiological states (*i.e.* normal states) or a square for pathological states (disorders or symptoms);
3. A central white pictogram indicating the anatomico-functional location (for example, a heart pictogram meaning both heart and cardiac function) or the patient's characteristic involved (such as pregnancy);
4. Zero, one or several shape modifiers indicating general types of disorders and morphologies (for example, a small bacterium for bacterial infection or a downward arrow for deficiency) or "transversal" anatomical structures that are present in most organs (such as blood vessels).

Two approaches are employed for representing the morphology of symptoms and disorders: if they are specific to a given anatomo-functional system (such as vomiting, which is specific to the stomach), a modified central pictogram is used; if the morphology is general and could apply to several anatomo-functional systems (such as infections, tumors or functional deficiencies), a shape modifier is used. These two approaches can be combined, and several shape modifiers can be used together as long as they do not overlap spatially on the icon.

Icons representing treatments or follow-up procedures are generated by taking the corresponding icon for the disorder treated or the risk of disorder followed-up, and by adding a top-right pictogram in green (treatment) or blue (follow-up procedure). The shape of the top-right pictogram indicates the type of treatment (for example, drug treatment, oral drug or surgery) or follow-up procedure (including laboratory tests and medical imaging). A second top-right pictogram can be added to represent health professionals or medical documents; for example, the cardiologist icon is created by adding the health professional top-right pictogram to the cardiac disorder icon.

VCM icons can either be used directly, for enriching medical texts, or a set of icons can be collected and displayed on a specific graphical user interface called "Mister VCM" [7]. This interface presents a schematized drawing of a Human body, with the various organ pictograms approximately placed at their real position (*e.g.* eye, ear, mouth and brain in the head). These

pictograms are grayed by default, and they are replaced by the VCM icon that includes this pictogram, if any. If several icons are located at the same pictogram, they are merged into a single, more general, icon. Examples of "Mister VCM" usage are synthesizing the disorders of a given patient, or the adverse effect of a given drug.

Due to the highly combinatory design of VCM, the total number of possible icons is very high; we evaluated it to about 200 million. However, most of these possible icons are inconsistent, or consistent but not useful (*e.g.* they do not correspond to existing disorders).

6.2. The VCM ontology

The VCM ontology aims at helping with the validation of the semantics of the VCM language. It was initially developed for verifying the consistency of icons [12], and then reused for establishing a preliminary mapping between VCM and a subset of SNOMED CT (Systematized Nomenclature of Medicine - Clinical Terms) [13]. This section describes the general principles followed during the design of the ontology in a more detailed fashion by putting the emphasis on the developing steps, including implementation details and technological aspects, and then presents the developed ontology. We also describe four applications:

1. The verification of icon consistency,
2. The semi-automatic generation of a mapping between VCM and SNOMED CT,
3. The automatic generation of a pictogram lexicon,
4. The automatic generation of multilingual labels for icons.

The ontology, as well as the alignment and the programs presented in this article, are available as Free Software (GNU LGPL license) and integrated in PyMedTermino[2] [33], a module for the manipulation of medical terminologies (including VCM) in the Python programming language.

6.2.1. General principles for the design of the ontology

The first principle that we applied for the design of the ontology was to distinguish the *icon ontology*, with the VCM icons and Graphical components, from the *domain ontology*, with the associated Domain concepts, as described in the general method in Section 4. To sum up, the "lung" pictogram is distinct from the "lung" organ.

The second principle was to employ whenever possible the *is-a* subsumption relations instead of other types of relations such as *part-of* meronymic relations, because most of existing tools, including ontology editors and reasoners founded on Description Logics, support subsumption but not meronymy. Indeed in such reasoners, all the reasoning services are reduced to inconsistency verification which is founded on subsumption calculation. More particularly, we chose in our ontology to model anatomical structures as "adjective + structure" using *is-a* relationship rather than only their organ name which could imply a *part-of* relationship; for instance we say that "a *stomachal structure* is a *digestive structure*" rather than "the *stomach* is a part of the *digestive tract*". Organs can then be added as sub-concepts of the anatomical structures: "the *stomach* is a *stomachal structure*". This transforms meronymic relations into subsumption relations. It contradicts the "minimal encoding bias" principle, however a similar approach is found in most medical terminologies, including SNOMED CT [34], and we adopted it.

The third principle is that all icons describe patient conditions, including the icons for treatments or follow-up procedures. For example the "anti-asthmatic drug treatment" icon is described as "patient treated by an anti-asthmatic drug". This corresponds to the way VCM represents treatments and procedures, by reusing the icon for the disorder being treated or the risk being monitored.

The three categories of constraints are present:

1. *Graphical constraints* in the icon ontology, involving VCM Graphical components. For example, the "tumor" and "virus" shape modifiers occupy the same place at the left of the icons, and thus they cannot be used together.
2. *Medical domain constraints* in the domain ontology, involving anatomical structures, biological functions, morphologies, *etc*. For example, a tumor is a morphology that can be applied to anatomical structures, but not to biological functions.
3. *Representation constraints* that link together the Graphical concepts and the Domain concepts.

During the design of the ontology, a set of about a hundred test icons was used iteratively, and the HermiT reasoner [35] was used to test the consistency of the ontology and to verify the results obtained on the test icon set.

6.2.2. Structure of the VCM ontology

The VCM ontology (Figure 11) has been divided in three modules. The icon ontology (240 concepts, 21 relations and 2597 axioms) describes VCM Graphical components and icons. It was automatically generated with Python scripts, from a text file listing the VCM components. The generated OWL file was then imported and manually edited into Protégé for adding the graphical constraints. These constraints restrict the components of an icon (for example, at most one central pictogram) and prevent overlapping components (for example, the "virus" and "tumor" shape modifiers).

The second module, the domain ontology (369 concepts, 18 relations and 828 axioms), describes the Domain concepts represented by the VCM graphical components: anatomical structures, biological functions, morphologies, pathological processes, patient characteristics (such as age classes) and types of treatments and follow-up procedures. The ontology includes basic Domain concepts and combination rules, but it does not include the entire list of disorders, treatments or procedures that can be generated by combination (*i.e.* it relies on *post-coordination* rather than on *pre-coordination*). The next subsection will provide more details on the design of the second module.

[2] `https://pypi.python.org/pypi/PyMedTermino`

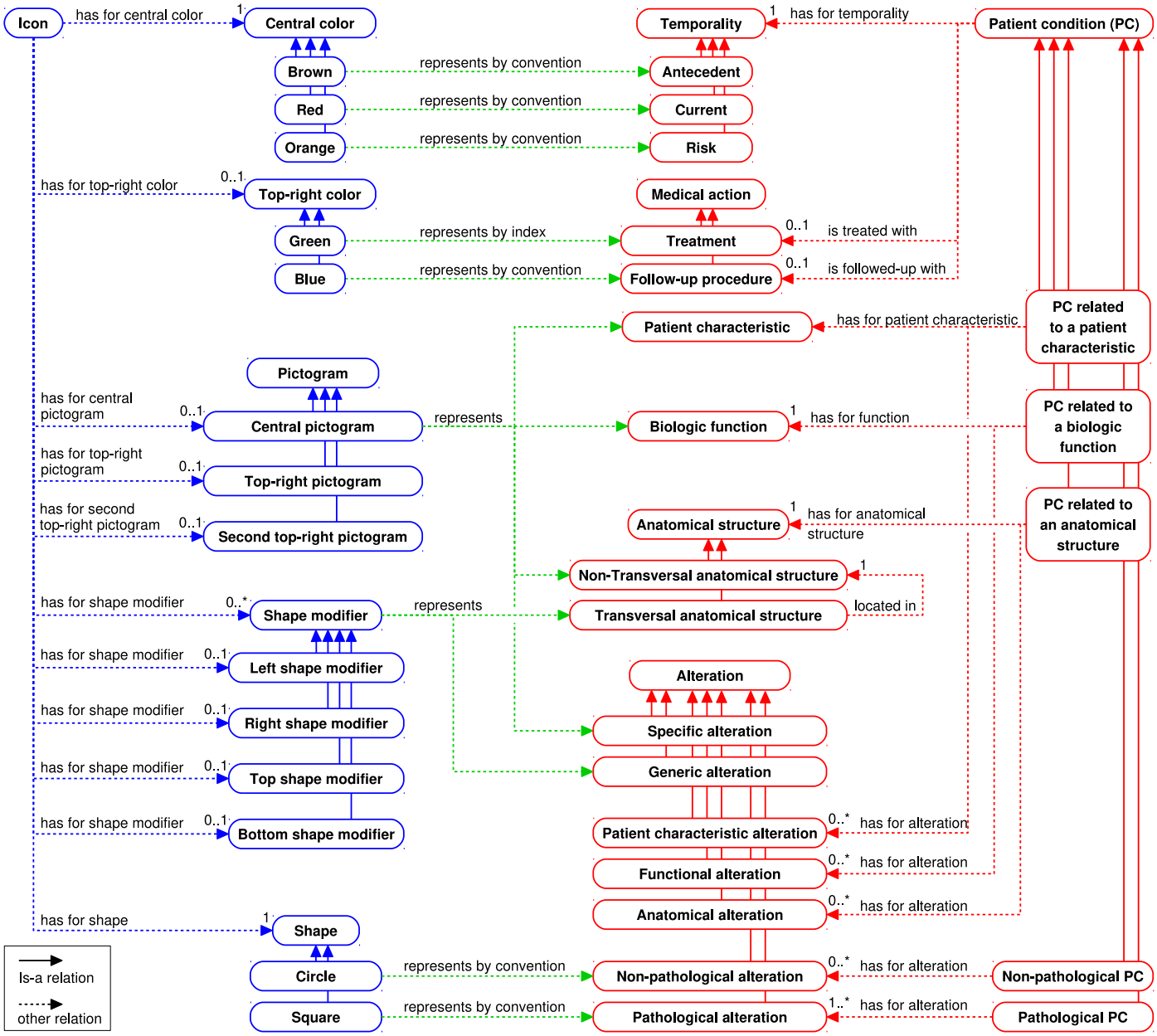

Figure 11: The main concepts and relations in the VCM ontology. The icon ontology is in blue, the domain ontology in red and the "represents" relations in green.

The concepts of these two ontologies, icon and domain, were linked by "represents" and "is-represented-by" relations (509 axioms), which constitute the third module, the mapping ontology. For example, the "lung" central pictogram is only present on icons that represents patient conditions involving a pulmonary structure or the respiratory function. And the pulmonary structure medical concept is only associated with patient conditions that are represented by icons with the "lung" central pictogram. The third module was generated automatically from a text file mapping each Graphical component to the medical concept(s) it represents, as described in section 4.2.

Each module was stored in a separate OWL/XML file, the third one importing the two others. The whole VCM ontology (*i.e.* the three modules) includes 609 concepts, 41 relations and 3934 axioms, defined using the OWL-DL language. It belongs to the $\mathcal{ALCIQ}$ family of Description Logics (Attribute Language, Complex concept negation, Role, Inverse property, Qualified cardinality restriction), for which it has been proved that the reasoning is decidable [36].

6.2.3. *Design of the domain ontology*

The domain ontology was modeled manually with the Protégé editor. It was inspired by the structure of medical terminologies such as ICD10 (International Classification of Diseases, release 10), SNOMED CT and the Semantic Network of UMLS (Unified Medical Language System) [37]. It was deliberately limited to a high level of granularity, corresponding to the one used by VCM. The original version of the domain on-

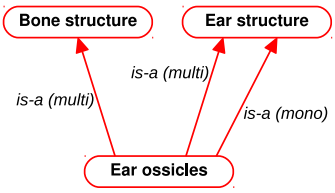

Figure 12: Representation of ear ossicles in the domain ontology.

tology was further refined during the design of a mapping with ICD10[3] and then with SNOMED CT (see section 6.4).

In ontologies and in medical terminologies, the use of multiple inheritance is a long debate. For instance, should ear ossicles be considered as both a bone structure and an ear structure, or should it be attached to a single parent structure? Although Guarino [38] argue for limiting *is-a overloading*, multiple inheritance is commonly used for automatic reasoning, *e.g.* for finding all bone-related disorders in a patient profile, and it is used in many multiaxial terminologies such as SNOMED CT.

However, while this representation of ear ossicles is true from an ontological point of view, it does not correspond to what clinicians would expect: in fact, disorders are usually classified by medical specialties, each organ and its disorders being associated to a single specialty. In our example, ear ossicle disorders fall in the ENT (Ear-Nose-Throat) specialty and not in rheumatology. Thus a physician seeing the VCM icon with the bone central pictogram would not think about an ear ossicle disorder. Single inheritance is often preferable when classifying disorders for presenting them to clinicians.

This is the reason why we chose to represent a given disorder by a single VCM icon, unless the disorder involves several, distinct, organs. We defined *two* is-a hierarchies in the domain ontology (Figure 12): one with multiple inheritance, *is-a (multi)*, and one with single inheritance, *is-a (mono)*. In the rest of the paper, is-a (multi) will be used for reasoning on semantics, unless explicitly stated otherwise. Is-a (mono) is only used when generating icons from Domain concepts. For example when obtaining the central pictogram associated with ear ossicles, we use is-a (mono) and thus we consider only the central pictogram inherited from ear structure, but not the one inherited from bone structure.

6.3. Verification of the icon consistency

6.3.1. Principles

The first application of the VCM ontology was the verification of the icons consistency [12]. In fact, some combinations of VCM components lead to inconsistent icons: for example an icon associating the "tumor" shape modifier and the "cardiac rhythm" central pictogram means "tumor of cardiac rhythm", which is semantically absurd. These inconsistent icons are problematic, especially when users have to create icons by themselves, by selecting and combining several components. Similarly to what we presented on traffic signs in section 5.3, the VCM ontology allows the verification of the consistency of icons.

Figure 13 shows the representation of an inconsistent icon in the ontology. The inconsistency can be inferred from the constraints modeled in the ontology, as follows:

1. The icon has the "cardiac rhythm" pictogram, and thus it represents a patient state related to the cardiac rhythm biological function.
2. The icon has the "tumor" shape modifier, and thus it represents a patient state involving the tumor pathological alteration.
3. Tumor is an anatomical alteration, and therefore it can only be applied to a patient condition related to an anatomical structure.
4. Anatomical structure and biological function are disjoint. Consequently, the icon cannot represent a patient state related to an anatomical structure.

This reasoning has been reproduced automatically with the Hermit reasoner [35], for a single icon (*i.e.* an individual in the ontology) but also for classes of icons that share a common subset of components (*e.g.* all icons with a given central pictogram and a given shape modifier). The inconsistencies inferred from the reasoner were then evaluated by experts [12]. It has also been used to detect errors in the VCM training software. The software contains 521 icons, 25 of them were classified as not consistent by the ontology. A manual check showed that 14 (out of 25) were errors in the training software (which have been fixed later), 5 were incomplete icons used as intermediary results for explaining how to combine the various components for creating icons, and 6 were actually consistent.

For consistent icons, the get_meaning(I) function can be used to compute its meaning. Below is an example of the computation of the meaning of the "disorder of respiratory system" icon (the first in Figure 10).

I = { Red central color, Square, Lung pictogram }

$D = \Big($ { Current },
{ Pathological alteration },
{ Respiratory structure, Respiration } $\Big)$

$P = \Big\{$ { Current, Pathological alteration, Respiratory structure },
{ Current, Pathological alteration, Respiration } $\Big\}$

$P' = P$

M = { Current, Pathological alteration, Respiratory structure, Respiration }

Note that *two* interpretations remain in P', which means that the icon itself is polysemic (it actually means either disorder of a respiratory structure or disorder of respiration). Indeed, VCM has polysemic pictograms but also polysemic icons.

6.3.2. Implementation

The semantic service operations performed by reasoning engines, such as consistency checking, have a huge performance cost since they require to run the ontology reasoner. In order to optimize the performances, the results are cached by storing them in a relational database (using SQLite3).

[3] Not described here because the mapping was purely manual and thus of low interest from a computer science point of view.

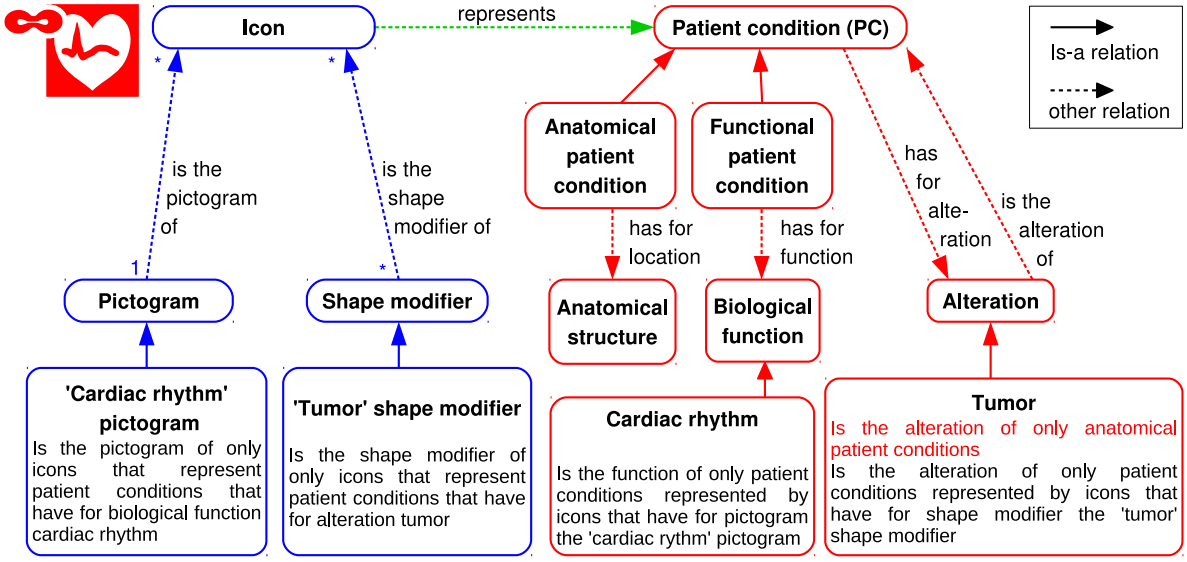

Figure 13: Representation of the "tumor of cardiac rhythm" inconsistent icon in the ontology.

For consistency checking, the inferred results obtained from the reasoner are cached in the database. For example, for the "disorder of respiratory system" icon, the following class I can be created :

$$
\begin{aligned}
I \sqsubseteq Icon \quad & \sqcap \quad \exists has\_for\_shape.Square \\
& \sqcap \quad \forall has\_for\_shape.Square \\
& \sqcap \quad \exists has\_for\_central\_color.Red \\
& \sqcap \quad \forall has\_for\_central\_color.Red \\
& \sqcap \quad \exists has\_for\_central\_pictogram.Lung\_pictogram \\
& \sqcap \quad \forall has\_for\_central\_pictogram.Lung\_pictogram \\
& \sqcap \quad \neg\exists has\_for\_shape\_modifier.Thing \\
& \sqcap \quad \neg\exists has\_for\_top\_right\_color.Thing \\
& \sqcap \quad \neg\exists has\_for\_top\_right\_pictogram.Thing \\
& \sqcap \quad \neg\exists has\_for\_second\_top\_right\_pictogram.Thing
\end{aligned}
$$

Then, a reasoner can be used to test whether I is consistent. Due to the very high number of possible icons (about 200 million), the cache does not have one entry per icon. The cache includes three relational tables: a table listing inconsistent pairs of Graphical components (*i.e.* any icon including one of these pairs is inconsistent), a table listing inconsistent $\big($ shape, { shape modifiers,... }, central pictogram $\big)$ tuples (excluding those having an inconsistent pair), and a table listing inconsistent icons (excluding those whose inconsistency can be computed using the two previous tables). The first table contains 3,292 rows, the second 36,550, and the third 1,999,911.

Similarly, for determining the meaning of an icon (the get_meaning() function), results were cached in the database. For example, for testing whether the previously defined icon I means the Lung organ, we can create I' as follows:

$$I' \sqsubseteq I \sqcap (\exists represent.(\exists has\_for\_anatomical\_structure.Lung))$$

Then, we use the reasoner to test whether I' is consistent. If (and only if) it is, then the lung organ is part of the meaning of the icon I (*i.e.* $Lung\_pictogram \in M$). For each icon, we tested all (icon, domain concept) pairs for each possible Domain concept. Pairs were grouped by batches of 10,000 and tested with the HermiT reasoner.

The entire generation of the cache takes about 3 hours on a recent computer, and the whole database size is about 120 Mb.

6.4. *Alignment of VCM icons with medical terminologies: the example of SNOMED CT*

The use of VCM in medical software requires the alignment of VCM with the existing medical terminologies, in order to associate automatically icons with the resources indexed by these terminologies, such as the disorders coded in patient records. In this section, we will describe the semi-automatic design of a mapping between VCM and SNOMED CT. SNOMED CT is a terminology that covers the various medical concepts, including anatomy, clinical conditions and disorders, procedures, *etc*. It also defines many relations between these concepts: *is-a* subsumption relations, *part-of* mereologic relations but also relations between clinical conditions and the anatomical structures and the morphologies they involve (for example hepatitis *is-located-in* liver and *has-for-morphology* inflammation).

The method for designing the semi-automatic alignment relies on the compositional nature of both SNOMED CT and the VCM ontology. To perform it, two steps are necessary:

1. The manual alignment of the Domain concepts in the VCM domain ontology (n=369) with the SNOMED CT corresponding concepts (mainly anatomical structures and morphologies),
2. The automatic alignment of SNOMED CT concepts for clinical conditions to VCM icons, by decomposing SNOMED CT concepts in anatomical structures and morphologies, then translating these anatomical structures and morphologies in VCM using the manual alignment produced at the previous step.

Algorithm 2 Algorithm for mapping the SNOMED CT clinical condition s to one (or several) VCM icon(s). Since SNOMED CT has some ontological feature without being a proper ontology, we formalized it in two part: an ontology ($\mathcal{O}_{SNOMEDCT}$) including the concepts and the is-a relations between them and a set of triples ($T_{SNOMEDCT}$) including the other (less formal) relations.

Let us denote :

- $O_{SNOMEDCT}$ the ontological part of SNOMED CT ($O_{SNOMEDCT}$ is limited to concepts related by is-a relations)
- $T_{SNOMEDCT}$ the other relations in SNOMED CT

  $T_{SNOMEDCT} = \{(s_1, r, s_2)\}$ where $s_1 \sqsubseteq SNOMEDCT\_concept \in O_{SNOMEDCT}$
  and $s_2 \sqsubseteq SNOMEDCT\_concept \in O_{SNOMEDCT}$
  and $r \sqsubseteq SNOMEDCT\_role \in O_{SNOMEDCT}$
- $T_{VCM \rightarrow SNOMEDCT}$ the manual mapping between Domain concepts of the VCM ontology and SNOMED CT

  $T_{VCM \rightarrow SNOMEDCT} = \{(d, \text{manually\_mapped\_to}, s)\}$ with $d \sqsubseteq Domain\_concept \in O_{domain}$
  and $s \sqsubseteq SNOMEDCT\_concept \in O_{SNOMEDCT}$

function map_snomedct_to_vcm(s):

$C_s = \{c \sqsubseteq SNOMEDCT\_concept \in O_{SNOMEDCT} \mid (s, \text{has-for-finding-site}, c) \in T_{SNOMEDCT}$
or $(s, \text{has-for-associated-morphology}\ c) \in T_{SNOMEDCT}$
or $(s, \text{has-for-pathological-process}, c) \in T_{SNOMEDCT}$
or $(s, \text{has-definitional-manifestation}, c) \in T_{SNOMEDCT}$
or $(s, \text{associated-with}, c) \in T_{SNOMEDCT}$
or $(s, \text{due-to}, c) \in T_{SNOMEDCT}\}$

$C'_s = \{c' \sqsubseteq SNOMEDCT\_concept \in O_{SNOMEDCT} \mid \text{there exists } c \in C_s \text{ such that } c \sqsubseteq c' \in O_{SNOMEDCT}$
or $(c, \text{part-of}, c') \in T_{SNOMEDCT}\}$

$M_s = \{d \sqsubseteq Domain\_Concept \in O_{domain} \mid \text{there exists } c' \in C'_s \text{ such that } (d, \text{manually-mapped-to}, c') \in T_{VCM \rightarrow SNOMEDCT}\}$

$I_s = \text{create\_icons}(M_s)$

return I_s

function create_icons(M):

$G_M = \{g \sqsubseteq Graphical\_component \in O_{icon} \mid \text{there exists } d \in M \text{ such that } O_{mapping} \models g \sqsubseteq \exists represents.d\}$

$I_M = \text{smallest set } \{ I \subseteq G_M \mid I \text{ is satisfiable with regard to } O_{domain} \cup O_{mapping} \cup O_{icon} \} \text{ such that } \bigcup I_M = G_M$

return I_M

This method was initially applied to a subset of SNOMED CT, the CORE problem list (6,173 terms) [13]. We present here the results for all clinical findings in SNOMED CT (99,626 terms).

6.4.1. Manual alignment between the VCM domain ontology and SNOMED CT

The Domain concepts of the VCM ontology were manually aligned with SNOMED CT. SNOMED CT uses multiple inheritance. However, we have seen in section 6.2.3 that multiple inheritance is not desirable when presenting disorders to clinicians, because they usually classify disorders by medical specialty, each disorder belonging to a single specialty. For instance, ear ossicle disorders should be represented by a single icon, with the "ear" central pictogram, and not by two icons, one with the "bone" pictogram and the other with the "ear". Consequently, we used is-a (mono) relationship when generating icons for SNOMED CT.

We designed a first manual alignment between the domain ontology and SNOMED CT. But the use of multiple inheritance in SNOMED CT resulted in several anatomical concepts that were mapped to several VCM Domain concepts (*i.e.* SNOMED CT concepts S_1 and S_2 were respectively manually mapped to Domain concepts D_1 and D_2, but then SNOMED CT concepts S_3, inheriting from both S_1 and S_2, was associated to both D_1 and D_2 through inheritance).

Therefore, we enriched the Domain ontology with "ambiguous" anatomical structures, *i.e.* the ones that belong to several hierarchies in SNOMED CT, such as S_3 above, or ear ossicles in the previous example. Thanks to Python scripts we developed, we searched for all SNOMED CT anatomical structures that would lead to several VCM central pictograms through multiple inheritance (n=181). Each of these anatomical structures was added to the domain ontology, with is-a (multi) relations similar to those in SNOMED CT and is-a (mono) relation manually determined, according to the medical specialty associated with the anatomical structure and the corresponding disorders, and their position in mono-axial terminologies such as ICD10.

This has led to the creation of new concepts (n=97, fewer than 181 because some close concepts were grouped together). The resulting manual alignment involved 1,753 SNOMED CT concepts and 369 concepts of the VCM domain ontology.

6.4.2. Automatic alignment between SNOMED CT and VCM

Then, clinical condition concepts of SNOMED CT were automatically aligned with VCM icons, by decomposing them, following Algorithm 2. Each clinical condition s was decom-

posed using the relations in SNOMED CT, yielding the set C_S of the associated anatomical structures, morphologies, *etc*. C_s is then enriched using the *is-a* and *part-of* relations in SNOMED CT, to produce a larger set C'_s. Next, using the manual mapping, SNOMED CT concepts in C'_s are translated into Domain concepts of the VCM ontology, producing the set M_s. Finally, the Domain concepts in M_s are translated to Graphical components which are assembled together, generating one or more VCM icons. This final step (performed by the create_icons() function in Algorithm 2) can produce several icons when it is not possible to represent all the Domain concepts on a single one (*e.g.* if two distinct organs are present in M_s, each of them requires a different central pictogram, and thus two icons are required).

For example, the "Uveitis" SNOMED CT clinical condition was decomposed into C_s = { Uveal tract anatomical structure, Inflammation morphology }. "Uveal tract" is a part of "Entire eye" which is a "Structure of visual system". Thus, $C'_s = C_s \cup$ { Entire eye anatomical structure, Structure of visual system }. In the manual mapping, "Structure of visual system" was mapped to the "Visual structure" Domain concept, and "Inflammation morphology" to "Inflammation". Therefore, M_s = { Visual structure, Inflammation }, leading to the following set of Graphical components: G_M = { Eye pictogram, Flaming square shape modifier }. The two Graphical components are then associated to generate the appropriate VCM icon.

The resulting alignment involved the 99,626 clinical findings in SNOMED CT and 1,957 VCM icons. 77,754 (78.0%) of the clinical findings were mapped to a single VCM icons, 7,573 (7.6%) were mapped to 2 icons and 517 (0.5%) were mapped to 3 or more icons. 13,782 (13.8%) of the clinical findings were mapped to an "empty" VCM icon (*i.e.* an icon without central pictogram and shape modifier); the manual analysis of these icons showed that they mostly included: concepts that were not considered as clinical conditions in VCM (*e.g.* "drug therapy finding"), concepts that qualify clinical findings (*e.g.* "clinical stage finding"), very general concepts (*e.g.* "alive") or symptoms without a proper location (*e.g.* "erythema", non localized, contrary to "erythema of skin").

6.5. Automatic generation of a pictogram lexicon for VCM

The VCM language documentation includes a lexicon of the various pictograms. This lexicon is used for learning VCM, but also serves as a reference for experts. The lexicon is a hierarchical list, each line associating a pictogram with its label(s) and its identifier.

The original version of the lexicon (Figure 14) has been written manually, in both French and English. This raises several problems:

1. The lexicon must be manually updated each time the VCM language is modified, for example with the addition of new pictograms.
2. The two versions of the lexicon, French and English, must be kept consistent one with each other.
3. The lexicon still contains ambiguities, for instance some labels are too generic, and thus for a given organ an expert may hesitate between two pictograms.

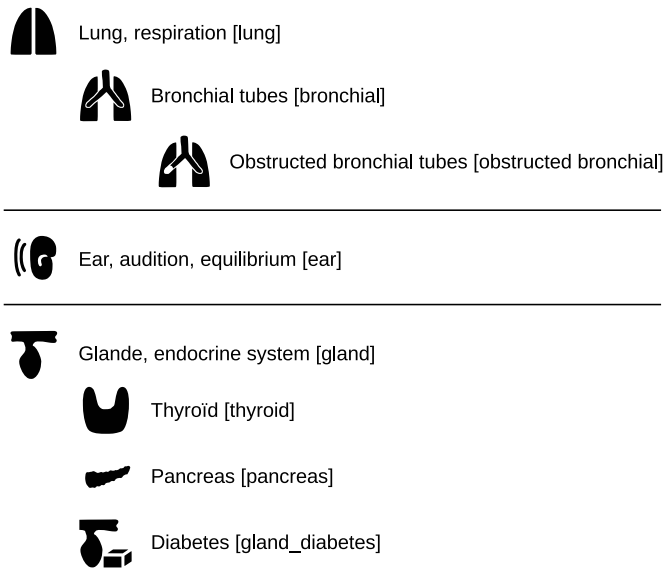

Figure 14: Three excerpts of the manually written VCM pictogram lexicon. The pictogram identifiers are shown in brackets.

4. The lexicon still lacks some information. Some organs have no corresponding label in the lexicon; in this case a more general pictogram must be used. For instance, there is no "pituitary" pictogram, and the user has to guess that the "gland" pictogram must be used instead. However, adding more labels to the lexicon would improve it.

In this section, we propose a method for the automatic generation of a pictogram lexicon for an iconic language whose semantics is described by an ontology.

6.5.1. Method for generating the lexicon

The generation of the lexicon was performed in four steps (Algorithm 3):

1. Extracting P, the whole set of pictograms from the VCM icon ontology.
2. For each pictogram p, obtaining M_p, the set of the Domain concepts represented by p, using the relations in the mapping ontology.
   A pictogram is often associated with several Domain concepts. Indeed, VCM frequently uses the same pictogram for representing both an organ and its biological function. The list of Domain concept includes the descendants concepts (children, grandchildren, *etc*., for example for the "lung" pictogram, "Pleural structure" is a descendant of "Pulmonary structure"), excepted those that are related to another, more specific, pictogram (for example, "Bronchial structure" which is related to the "bronchial" pictogram, is more specific than the "lung" one).
3. Sorting the concepts in M_p.
   When several concepts are associated with a given pictogram, the following question arises: in which order the concepts should be displayed in the lexicon ? We set up rules to solve this problem, taking into account the nature

Algorithm 3 Algorithm for generating the pictogram lexicon. The function has_priority() compares two domain concepts d and d', and returns true if d should be placed before d', and false otherwise. In the function has_priority(), the letters at the end of the lines refer to the sorting rules described in the text.

$P = \{p \mid p \sqsubseteq Pictogram \in O_{domain}\}$
for each p in P :
  $M_p = \Big\{d \mid d \sqsubseteq Domain\_concept \in O_{domain}$ and $(p, \text{represents}, d) \in T_{mapping}$
    and $\big($there does not exist p', such that $p' \sqsubseteq p \in O_{domain}$ and $(p', \text{represents}, d) \in T_{mapping}\ \big)\Big\}$
  D_p = sort the Domain concepts in M_p, using the comparison function has_priority(d, d')
  Generate the lexicon entry for Pictogram p, with the labels associated with the Domain concepts D_p

function has_priority(d, d') :
  if $(d \sqsubseteq Organ \in O_{domain})$ and $(O_{domain} \models d' \sqcap Organ \sqsubseteq \bot)$ then return true *(a)*
  if belongs_to_two_hierarchies(d) and not (belongs_to_two_hierarchies(d')) then return true *(b)*
  if $(d \sqsubseteq AnatomicalStructure \in O_{domain})$ and $(d' \sqsubseteq BiologicalFunction \in O_{domain})$ then return true *(c)*
  if $(d \sqsubseteq AnatomicalRegion \in O_{domain})$ and $(O_{domain} \models d' \sqsubseteq Tissue \sqcup Cell \sqcup Liquid)$ then return true *(d)*
  if $(d \sqsubseteq Tissue \in O_{domain})$ and $(O_{domain} \models d' \sqsubseteq Cell \sqcup Liquid)$ then return true
  if $(d \sqsubseteq Cell \in O_{domain})$ and $(d' \sqsubseteq Liquid \in O_{domain})$ then return true
  if $(d' \sqsubseteq d \in O_{domain})$ then return true *(e)*
  return false

function belongs_to_two_hierarchies(d):
  $A = \{a \mid a \sqsubseteq AnatomicalStructure \in O_{domain}$ and $d \sqsubseteq a\}$
  if there exists (a, a'), $a \in A$, $a' \in A$ such that $O_{domain} \models a \sqcap a' \sqsubseteq \bot$ then return true
  return false

of the concepts (anatomical structures or biological functions), the scale level (macroscopic or microscopic), the specificity (general or specific) and the readability of the corresponding labels (organ names are usually shorter and more readable than the structure name, for example "lung" is more readable than "pulmonary structure"). The rules are the following (in decreasing order of priority):

(a) When an organ is present in the concepts, it comes first in the lexicon.
(b) Concepts belonging to several anatomical hierarchies (such as ear ossicles, which are both in the "auditive structure" and the "bone structure" hierarchies) are typically more specific than those found in a single hierarchy. Consequently, they are put at the end, and written in gray.
(c) Anatomical structure concepts are placed before biological function concepts.
(d) Anatomical structure concepts are ordered from macroscopic to microscopic. A four-level scale was considered: anatomical region, tissue, cells, liquid, as defined in the VCM ontology.
(e) More general concepts were placed before the more specific ones (according to the *is-a* relations in the ontology, for example "diabetes" is placed above "diabetes type 2").

4. Gathering the labels associated with each Domain concept, and create the lexicon entry. For each concept, the ontology included a preferred term and eventually one or more synonyms or hyponyms. The preferred term is put first, followed by the "including" mention and the other terms.

The order in which the pictograms appear in the lexicon was defined manually, following the same anatomical order as in the original manual lexicon.

6.5.2. Results : the produced lexicon

Figure 15 shows excerpts of the lexicon produced from the ontology. Compared to the original lexicon (Figure 14), the new one is incontestably richer.

The alignment of Domain concepts with SNOMED CT (section 6.4.1) guarantees the covering of the whole domain, especially for anatomy. Thus, the generated lexicon has almost no lacking organs or concepts : all anatomical structures are either present in the lexicon, or a parent structure is present (for example, "pleural structure" was missing in the original lexicon and is present in the generated one). Moreover, when designing the mapping with SNOMED CT, we detected the 181 ambiguous medical concepts that were associated with more than one pictograms. All these ambiguous concepts were automatically detected, then manually disambiguated and clearly associated with a single pictogram. As a result, all 181 ambiguous concepts are now present in the lexicon and associated with the right pictogram.

6.5.3. Evaluation

The new lexicon was evaluated by checking that the entries from the old and manually written, lexicon were still present in it. The old lexicon included 499 entries, 389 of them have been found in the new lexicon. The 110 missing entries were analyzed manually. We classified them in 7 categories: linguistic variants (45), *e.g.* "behavior" *vs* "behaviour"; synonyms (17)

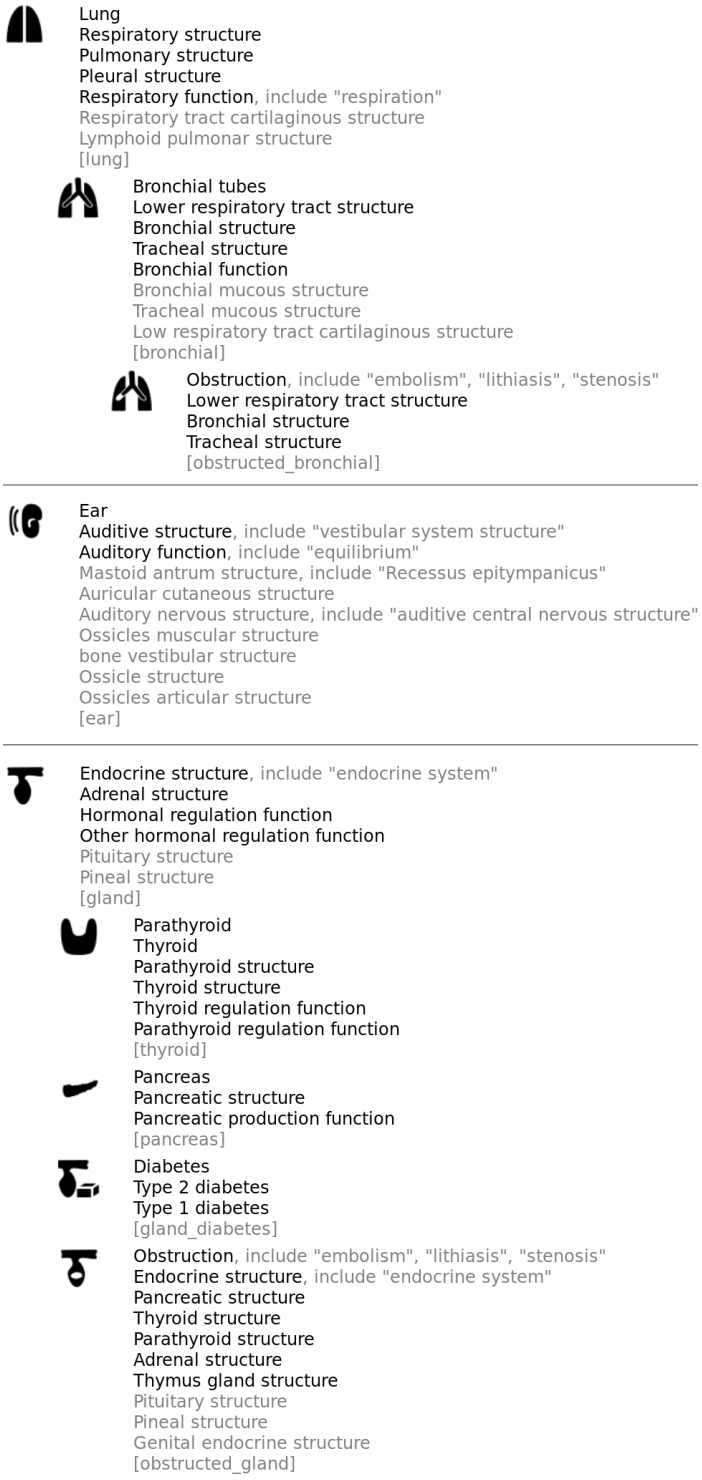

Figure 15: Three excerpts of the VCM pictogram lexicon generated from the ontology.

often with a better term in the new lexicon, *e.g.* "extrapyramidal troubles" (old) *vs* "extrapyramidal syndrome" (new); erroneous terms in the old lexicon (3), *e.g.* adnexa (of skin) instead of hairs, orthographic errors in the old lexicon (14), *e.g.* "bone narrow" instead of "bone marrow", term corresponding to pictograms that have been removed from VCM later (10), and terms related to obstructions (21), which are represented as two separate semantic entities in the new lexicon (one for the obstruction and one for the obstructed organ).

6.6. Automatic generation of multilingual labels for VCM icons

In order to facilitate the learning of the VCM language, the association of a textual label to each icon is needed, such as the ones in Figure 10. In this section, we show how to generate such labels automatically, in several languages.

6.6.1. Method for generating the labels

The labels have been divided in six parts:

1. pre-prefix, which indicates the treatment of the follow-up procedure involved (*e.g.* "drug for"), if any,
2. prefix, which indicates the temporality (*e.g.* "antecedents of" or "risk of", or empty string for current state),
3. adjective (adj.), which refers to one or more adjectives qualifying the patient state (*e.g.* "vascular"), if any,
4. base, which refers to the noun of the patient state (*e.g.* disorder or hypertension),
5. complement (comp.), which is a grammatical complement qualifying the patient state (*e.g.* "of respiratory tract"), if any,
6. suffix, which refers to an additional complement (*e.g.* "with obstruction"), if any.

To enable the automatic generation of labels, we manually created a dictionary that maps sets of Domain concepts (from the VCM domain ontology) to multilingual label parts. The dictionary can be formalized as a set of (key, value) pairs, where the key is a set of Domain concepts and the value is a triplet including the English label part, the French one, and the name of the part (see excerpts in Algorithm 4).

The dictionary includes label parts of various granularity, from very general (*e.g.* "disorder") to very specific ("pulmonary hypertension"). Dictionary entries with more specific subsets (*i.e.* including a higher number of concepts and/or more specific ones) are prioritized over more generic entries. For example, the entry for {PathologicalAlteration, Hypofunction} is prioritized over the entry for {PathologicalAlteration}. In addition, priority rules have been manually edited between entry with the same granularity level, when needed.

A label is generated from an icon in four steps (Algorithm 5):

1. Computing M_{I1}, the set of the Domain concepts associated with the icon in the VCM ontology, using the get_meaning() previously described function.
2. Selecting E_I, the set of dictionary entries matching M_I. The previously defined priority order is taken into account and, whenever a match is found, the dictionary entry is added to E_I, and the process continues with the remaining concepts (*i.e.* the matched concepts are consumed and thus no longer available for future matches).
3. Selecting V_I, the label parts of the matched dictionary entries E_I.

Algorithm 4 Definition of the dictionary mapping subsets of Domain concepts to multilingual label parts, and an excerpt of the dictionary content with the label parts required for generating the labels of the icons in Figure 10.

part_names = { pre-prefix, prefix, adj, base, comp, suffix }
dictionary = $\left\{(k, v), k = \{d \mid d \sqsubseteq \mathit{DomainConcept}\}, v = (\text{label\_part}_{\text{En}}, \text{label\_part}_{\text{Fr}}, \text{part\_name} \in \text{part\_names})\right\}$

dictionary = {
({DomainConcepts,...}, ("English label", "*French label*", part_name))
({PathologicalAlteration}, ("disorder", "*maladie*", base)),
({PathologicalAlteration, Hypofunction}, ("insufficiency", "*insuffisance*", base)),
({PathologicalAlteration, Inflammation}, ("inflammation", "*inflammation*", base)),
({RespiratoryStructure}, ("of respiratory tract", "*des voies respiratoires*", comp)),
({RespiratoryFunction}, ("respiratory", "*respiratoire*", adj)),
({RespiratoryStructure, RespiratoryFunction}, ("of respiratory system", "*du système respiratoire*", comp)),
({PathologicalAlteration, RespiratoryStructure, Hyperfunction, BloodPressureRegulation},
("pulmonary hypertension", "*hypertension pulmonaire*", base)),
({CardiacStructure}, ("of heart or pericardia", "*du coeur ou du péricarde*", comp)),
({CardiacStructure, CardiacFunction}, ("of heart or pericardia", "*du coeur ou du péricarde*", comp)),
({GastricStructure}, ("of stomach", "*de l'estomac*", comp)),
({VascularStructure}, ("vascular", "*vasculaire*", adj)),
({VascularStructure, CardiacStructure}, ("coronary", "*coronaire*", adj)),
... }

Algorithm 5 Algorithm for generating the multilingual label ($L_{I,En}$, $L_{I,Fr}$) for a given icon I.

I is an icon
M_I = get_meaning(I)
E_I = {∅}
Do while true:
  candidate_entries = { $(k, v) \in$ dictionary $\mid k \subseteq M_I$ }
  if *candidate_entries* = {∅} : break
  best_entry = $(k, v) \in$ *candidate_entries* such that (k, v) is the entry with the highest priority according to the has_priority() function
  Remove all concepts in k from M_I
  Add *best_entry* to E_I
$V_I = \{v \mid (k, v) \in E_I\}$
$L_{I,En}$ = concatenate(getEn(V_I, pre-prefix), getEn(V_I, prefix), getEn(V_I, adj), getEn(V_I, base), getEn(V_I, comp), getEn(V_I, suffix))
$L_{I,Fr}$ = concatenate(getFr(V_I, pre-prefix), getFr(V_I, prefix), getFr(V_I, base), getFr(V_I, adj), getFr(V_I, comp), getFr(V_I, suffix))

function has_priority($(k, v), (k', v')$):
  Take into account manual rules if some apply
  if $|k| > |k'|$: return true
  if $k' \sqsubseteq k$: return true
  return false

function get_En(V, *part_name*):
  P = { $\text{label\_part}_{\text{En}} \mid (\text{label\_part}_{\text{En}}, \text{label\_part}_{\text{Fr}}, \text{part\_name'}) \in V$ and part_name' = part_name }
  return P

function get_Fr(V, *part_name*):
  P = { $\text{label\_part}_{\text{Fr}} \mid (\text{label\_part}_{\text{En}}, \text{label\_part}_{\text{Fr}}, \text{part\_name'}) \in V$ and part_name' = part_name }
  return P

4. Assembling the label parts V_I to generate the icon label in the desired language. In English, the order of the six parts is: pre-prefix, prefix, adjective, base, complement, suffix. In French, the order is different: pre-prefix, prefix, base, adjective, complement, suffix (adjective and base are reversed). When several parts of the same type are present, *e.g.* two adjectives, they are placed in an arbitrary order.

For example, for the "respiratory insufficiency" icon, the Domain concepts are M_I = { Pathological alteration, Current, Respiratory function, Hypofunction }. It matches the following entries in the dictionary: { Pathological alteration, Hypofunction } and {Respiratory function} (the { Pathological alteration } subset is also present, but { Pathological alteration, Hypofunction } has a higher priority and it consumes the Pathological alteration concept, which is thus no longer available). The assembly of the corresponding label parts leads to the "respiratory insufficiency" English label.

6.6.2. Results : the produced labels

The automatic label generating system allows the generation of bilingual English and French labels, for any consistent VCM icon. For example, labels in Figure 10 have been generated automatically using this method. The entire dictionary includes 460 bilingual label parts, 257 of them have been manually ordered for priority.

6.6.3. Evaluation

In order to evaluate the generated labels, we randomly chose 100 icons from a set of 35,226 icons. This set included the 1,957 icons from the mapping with SNOMED, with various central colors and exponents (35,226 = 1,957 icons × 3 central colors × 6 exponents). The 100 icons and their generated labels in French and English were reviewed by the two authors, independently.

Only one label was considered as really problematic ("risk of arrest related to pregnancy" instead of "risk of miscarriage"). Two spelling mistakes were also identified ("bénine" instead of "bénigne", "goître" instead of "goitre"), as well as two awkward phrasing ("history of..." instead of "past history of...", "drug treatment for" instead of "drug treatment of").

The encountered problems were corrected after the evaluation.

6.7. The VCM iconic server

To facilitate the integration of VCM in medical applications, we implemented an *iconic server* for VCM. The server relies on the HTTP protocol and can be interrogated with either a web browser or ReST XML services (Representational State Transfer). It has been implemented in Python 3, using PyMedTermino [33], which provides access to medical terminologies but also semantically-enabled set operations.

We associated a unique ID with each icon, the ID being the concatenation of the codes corresponding to its Graphical components. Typical queries to the server are:

- Obtain the SVG image file from an icon ID (including consistency, meaning *i.e.* set of associated Domain concepts, and English and French labels, located in metadata of the SVG file).
- Obtain the icon ID(s) for a given SNOMED CT concept.
- Merge several icons into a more generic one.
- Generate VCM-based user interface, such as "Mister VCM", from a set of icons.

7. Discussion

In this paper, we described a method for formalizing the semantics of an iconic language, using a formal ontology that describes the icons (or signs) of the language and their syntax, the object represented by the icons and their properties, and the "represents/is-represented-by" relations that exist between them. We have shown that this method is generic enough to be applied to two different iconic languages, the traffic signs and the VCM medical iconic language. We also showed the interest of this formalization through four practical applications: the verification of the icon consistency, the alignment of the icons with existing terminological resources, the automatic generation of a pictogram lexicon and the automatic generation of multilingual labels for the icons. These applications could have been achieved using knowledge representations specific to each application (for instance, grammatical rules for verifying the icon syntax), however the ontology we designed allowed the implementation of all applications from a single, unique knowledge source, which is easier to maintain.

7.1. Grammar vs. *semantics*

In related works (section 2), we have shown that several approaches proposed for formalizing the semantics of graphical languages were actually based on grammatical formalisms. For example, we could have determined the consistency of traffic signs or VCM icons using a formal grammar, such as the following one:

<sign> ::= <prohibition sign> | <obligation sign>
| <danger information sign>
| <other information sign>
<danger information sign> ::= red triangle with an <encounter> inside
<prohibition sign> ::= red circle with an <action> inside
<encounter> ::= left turn, road works,...
<action> ::= left turn, drive above 50 km/h,...

However, this formalization integrates heterogeneous elements without distinguishing them: graphical elements (red circle) and non-graphical ones (road works) are mixed. Moreover, it does not separate the represented object from its graphical representation, nor the grammar from the semantics. Actually, the "road works are prohibited" traffic sign or the "tumor of cardiac rhythm" VCM icon should be perfectly correct from a

grammatical point of view, despite they are absurd and inconsistent from a semantic point of view. However, when using a formal grammar (such as the one presented above), these icons are detected as grammatically incorrect. Therefore, using a grammar for expressing the semantics seemed inappropriate to us. In addition, the semantics often relies on hierarchical inheritance relations (*is-a*), and thus an ontology seemed more adapted.

Moreover, the four applications we presented could not have been implemented if we did not express the semantics independently from the grammar. For example, it would have been difficult to generate a pictogram lexicon from a grammar in which pictograms and medical concepts would have been mixed.

7.2. Polysemy

Polysemy is one of the major difficulty when dealing with the semantics of iconic languages. Indeed iconic languages are often highly polysemic because icons usually do not aim at being as precise as text. The two iconic languages we worked on in this study have polysemy, but not at the same level. Traffic signs have polysemic pictograms, but not polysemic icons (each traffic sign has a single well-defined meaning). On the contrary, VCM has both polysemic pictograms (*e.g.* most organ pictograms are polysemic and mean both the organ and the associated function) and polysemic icons (*e.g.* icons that can represent several distinct disorders, usually closely related ones). The ontology-based methods (expressed in Description Logics) and algorithms we described were able to cope with both forms of polysemy. In the literature, ontologies have already been proposed for dealing with polysemy in the medical domain [39].

7.3. Comparison with the literature

The method we presented here for formalizing the semantics of an iconic language (section 4) shares some similarities with some of the approaches described in the state of the art (section 2), more precisely (a) semiotic studies [15] for the decomposition of icons in graphical components (pictograms, geometrical shapes, colors), (b) abstract visual syntaxes [11, 27] for the use of a non-graphical syntax, more abstract and formal than graphics, and (c) the works of Haarslev [28] for the use of a formal ontology relying on description logics (DLs). However, contrary to Haarslev, we tried to represent in the ontology the syntax of the language but also its semantics.

7.4. Discussion about the semantic-powered applications

The first application of the ontology was the verification of icon consistency. In the literature, consistency checking has been widely studied for auditing medical terminologies [40], and tracking inconsistent terms. Two categories of methods are distinguished for searching for inconsistent terms [41]: *linguistic-based methods* searching for lexical inconsistency, and *ontological methods* searching for inconsistent classifications. Both methods can rely either on *extrinsic knowledge*, *i.e.* the terminology is compared to another source of knowledge such as another terminology, or on *intrinsic knowledge*, *i.e.* the terminology consistency is checked with regards to knowledge inferred from the terminology itself, either manually or automatically. However, linguistic methods could not be applied to iconic languages. Ontological methods typically consist of defining formal constraints and then searching for terms or concepts violating these restrictions [41]. Such methods have been applied to the GALEN project [42] and to various medical terminologies including the Medical Subject Heading (MeSH) [43], the Standardized Nomenclature of Medicine Clinical terms (SNOMED CT) [44, 45], the International Classification of Diseases 10th release (ICD10) [46], the Foundational Model of Anatomy (FMA) [47, 48], the National Cancer Institute (NCI) thesaurus [49] and the Unified Medical Language System (UMLS) [50, 51].

The second application is the alignment of VCM icons with SNOMED CT, a reference terminological resource in medicine. This alignment has been done semi-automatically thanks to the relations present in both the VCM ontology and SNOMED CT. In the literature, three methods are generally considered for establishing mapping between medical terminologies [52]: (1) chaining several existent mappings, (2) using lexical methods for searching identical or similar terms, (3) designing the mapping manually, and (4) using mapping ontology alignment or matching methods. This fourth approach can only be used when the two mapped terminologies are structured and described using Description Logics (DLs), and thus it has rarely been used on medical terminologies. Our work is one of the rare examples of this semantic-powered approach. This semi-automatic approach is also interesting for updating the mapping when new pictograms are added to VCM or when a new version of SNOMED CT is available (twice a year).

However, we did not manage yet to reproduce a similar approach for less structured terminologies such as ICD10. Poorly structured resources would probably still require a manual alignment.

The third application is the automatic generation of a pictogram lexicon from the iconic language's ontology. This automatically generated lexicon is clearly richer than a manually written lexicon. In addition, the alignment of the concepts of the ontology with external resources such as SNOMED CT ensure that there is no missing information or ambiguity in the lexicon. Moreover, the lexicon is bilingual (English and French), and it can be automatically updated when the ontology is modified.

An important difficulty we encountered when working on the generation of the lexicon was relative to the order of the items shown in the lexicon. In fact, the ontology does not define any order between its concepts. For example an ontology can define the *mouth*, the *esophagus* and the *stomach* concepts as being three *digestive structures*. However, the three concepts are not ordered even if there is actually an intuitive order between them. We proposed some rules for ordering the various concepts and labels listed for a given pictogram; these rules considered several criteria: the scale level, the specificity and the nature of the concept, and the readability of the labels. On the contrary, the order of the pictograms in the lexicon was determined manually. This order is important because a typical user expects the pictograms of a given system (*e.g.* digestive

or cardiovascular systems) to be grouped together. Moreover, for some systems like digestive systems, there is an intuitive order for presenting the organ or the function, *e.g.* following the course of the alimentary bolus (mouth, esophagus, stomach, intestine, anus). More detailed ontologies of anatomy, such as FMA (Foundational Model of Anatomy) [53, 54], include connecting relations between organs (*e.g.* the mouth *is-connected-to* the esophagus). However, even with those relations, it is not possible to determine the exact order for presenting the organs (*i.e.* from mouth to anus or from anus to mouth ?).

In the literature, most of the works relative to ontologies and lexicons aimed at creating an ontology from an existent lexicon, which is the opposite of what we described here: Natural Language Generation (NGL) from an ontology [55]. However, the generation of a lexicon from an ontology has already been proposed [56], in particular in well-defined technical domains. The problem we encountered for ordering the items in the lexicon is similar to the one encountered by the tools generating descriptions in natural language for the concepts of an ontology, such as NaturalOWL [57, 58]. These tools produce a textual definition from a concept and its relations. The order in which the relations are considered and appear in the text is typically configured manually by the user, and not inferred from the ontology.

The fourth application is the automatic generation of multilingual labels for each icon. We applied this method to English and French, and our method could probably work for most occidental languages. However, the automatic generation of labels in languages more distant (such as Arabic or Chinese) or with different fundamental structures (for example declensional languages, such as German) could raise new research problems.

7.5. Perspectives

A first perspective is the development of additional semantic-powered applications. An example would be the automatic generation of a diagram explaining the meaning of an icon by decomposing it (in the spirit of Figure 1). Such diagrams could help the learning of iconic languages, such as VCM. Another example would be the automatic generation of icons for a pictorial retrieval system [59], based on semantics.

A second perspective is the application of our formalization method to other iconic languages. VCM is one of the most complex domain-specific iconic languages. However, it is also very structured, and thus less-structured languages may raise new problems.

Another type of iconic languages is "universal" general iconic languages, usually targeting either foreigners and tourists, or persons with impaired language cerebral center. These persons are unable to use natural languages but they manage to use graphical languages. Examples of such graphical languages are VIL [60], Miracle [61], and Blissymbolics. These languages use icons, but also iconic sentences which organize several icons according to a specific grammar. Our method could help to describe their semantics, but it needs to be extended for formalizing iconic sentences.

A third perspective is the use of the method beyond iconic languages. For example, the method we described for building semi-automatic mapping between an iconic language and a terminology could easily be generalized to the mapping from a terminology to another one.

8. Conclusion

In conclusion, we described a generic method for the formalization of the semantics of iconic languages. The method has been applied to traffic signs and to VCM, an iconic language for medical concepts such as disorders or treatments. The formalization clarified the syntax, the grammar and the semantics of the language. It also permitted various semantic-based applications. The formalization of the semantics could facilitate the conception of new iconic languages, improve the consistency and the quality of these languages, ease their learning through the automatic generation of multilingual high-quality lexicons and labels, and finally facilitate their use in practice by helping to establish mapping with existing termino-ontological resources of the application domain.

Acknowledgments

This work was partly supported by the French national research agency (ANR, *Agence Nationale de la Recherche*) during the L3IM [grant number ANR-08-TECS-007] and SiFaDo [grant number ANR-11-TECS-0014] research projects.

References

[1] Dreyfuss H, Symbol sourcebook: An Authoritative Guide to International Graphic Symbols, John Wiley and sons, 1984.

[2] X. Ma, J. P. Cahier, Graphically structured icons for knowledge tagging, Journal of Information Science 40 (6) (2014) 779–795.

[3] J. Vigneron, I. Gindre, M. Daouphars, P. Monfort, S. Georget, B. Demoré, E. Chenot, V. Noirez, N. Commun, F. Laurelli, A. Perrin, M. Lux, M. A. Hoffman, M. Hoffman, Stabilis 3: a European database on the stability and compatibility of injectable drugs, European Journal of Hospital Pharmacy Practice 12 (6) (2006) 77–78.

[4] V. J. Henry, A. E. Bandrowski, A. S. Pepin, B. J. Gonzalez, A. Desfeux, OMICtools: an informative directory for multi-omic data analysis, Database : the journal of biological databases and curation 2014.

[5] J. B. Lamy, C. Duclos, A. Bar-Hen, P. Ouvrard, A. Venot, An iconic language for the graphical representation of medical concepts, BMC Medical Informatics and Decision Making 8 (2008) 16.

[6] J. W. Ely, J. A. Osheroff, M. H. Ebell, M. L. Chambliss, D. C. Vinson, J. J. Stevermer, E. A. Pifer, Obstacles to answering doctors' questions about patient care with evidence: qualitative study, BMJ 324 (7339) (2002) 710.

[7] J. B. Lamy, A. Venot, A. Bar-Hen, P. Ouvrard, C. Duclos, Design of a graphical and interactive interface for facilitating access to drug contraindications, cautions for use, interactions and adverse effects, BMC Medical Informatics and Decision Making 8 (2008) 21.

[8] N. Griffon, G. Kerdelhué, S. Hamek, S. Hassler, C. Boog, J. B. Lamy, C. Duclos, A. Venot, S. J. Darmoni, Design and usability study of an iconic user interface to ease information retrieval of medical guidelines, J Am Med Inform Assoc 21 (e2) (2014) e270–7.

[9] S. Pereira, S. Hassler, S. Hamek, C. Boog, N. Leroy, M. C. Beuscart-Zéphir, M. Favre, A. Venot, C. Duclos, J. B. Lamy, Improving access to clinical practice guidelines with an interactive graphical interface using an iconic language, BMC medical informatics and decision making 14 (1) (2014) 77.

[10] C. Simon, S. Hassler, M. C. Beuscart-Zephir, M. Favre, A. Venot, C. Duclos, J. B. Lamy, Using an iconic language to improve access to electronic medical records in general medicine, Stud Health Technol Inform 205 (2014) 333–7.

[11] Erwig M, Semantics of Visual Languages (1997).

[12] J. B. Lamy, L. F. Soualmia, G. Kerdelhué, A. Venot, C. Duclos, Validating the semantics of a medical iconic language using ontological reasoning, J Biomed Inform 46 (1) (2013) 56–67.

[13] J. B. Lamy, R. Tsopra, A. Venot, C. Duclos, A Semi-automatic Semantic Method for Mapping SNOMED CT Concepts to VCM Icons, Stud Health Technol Inform 192 (2013) 42–6.

[14] Meunier JG, La structure générique des systèmes sémiotiques, Recherche sémiotique / Semiotic inquiries (RSSI) 8 (1988) 75–107.

[15] Meunier JG, The categorial structure of iconic languages, Theory&Psychology 8 (6) (1998) 805–825.

[16] H. Huang, H. H. Lai, Factors influencing the usability of icons in the LCD touchscreen, Displays 29 (4) (2008) 339–344.

[17] S. Ghayas, S. Sulaiman, M. Khan, J. Jaafar, The effects of icon characteristics on users' perception, in: International Visual Informatics Conference (IVIC2013), Lecture Notes in Computer Science, Vol. 8237, 2013, pp. 652–663.

[18] L. Kascak, C. B. Rébola, R. Braunstein, J. A. Sanford, Icon design for user interface of remote patient monitoring mobile devices, in: Proceedings of the 31st ACM international conference on design of communication, Vol. 77-84, 2013.

[19] Y. B. Salman, H. I. Cheng, P. E. Patterson, Icon and user interface design for emergency medical information systems: a case study, Int J Med Inf 81 (1) (2012) 29–35.

[20] C. Nakamura, Q. Zeng-Treitler, A taxonomy of representation strategies in iconic communication, Int. J. Human-Computer Studies 70 (2012) (2012) 535–551.

[21] X. Ma, N. Matta, J. P. Cahier, C. Qin, Y. Cheng, From action icon to knowledge icon: Objective-oriented icon taxonomy in computer science, Displays 39 (2015) 68–79.

[22] S. C. Huang, R. G. Bias, D. Schnyer, How are icons processed by the brain? Neuroimaging measures of four types of visual stimuli used in information systems, Journal of the association for information science and technology 66 (4) (2015) 702–720.

[23] Marriott K, Constraint multiset grammars, in: Proceedings of IEEE symposium on Visual Languages, 1994.

[24] G. Costagliola, A. De Lucia, S. Orefice, G. Tortora, A parsing methodology for the implementation of visual systems, in: IEEE Transactions on Software Engineering, Vol. 23, 1997, pp. 777–799.

[25] A. Zolotas, D. S. Kolovos, N. Matragkas, R. F. Paige, Assigning semantics to graphical concrete syntaxes, in: Extreme modeling workshop (XM2014), 2014.

[26] V. Rajarajan, C. L. Kiernan, S. P. MacLeod, S. E. Oberst, Pluggable notations and semantics for visual modeling elements (US Patent 7,320,120) (2008).

[27] S. Ellner, W. Taha, The Semantics of Graphical Languages, in: Proceedings of the ACM SIGPLAN Symposium on Partial Evaluation and Semantics-Based Program Manipulation, 2007.

[28] Haarslev V, Visual Language Theory, Vol. 261-292, Springer, New York, 1998, Ch. A fully formalized theory for describing visual notations.

[29] Baar T, Correctly defined concrete syntax for visual modeling languages, in: Model driven engineering languages and systems, Lecture Notes in Computer Science, Vol. 4199, Springer, Berlin, Heidelberg, 2006, pp. 111–125.

[30] G. Costagliola, M. De Rosa, V. Fuccella, Extending local context-based specifications of visual languages, Journal of Visual Languages and Computing 31 (2015) 184–195.

[31] N. C. Kuicheu, N. Wang, G. N. Fanzou Tchuissang, F. Siewe, D. Xu, Description logic based icons semantics: An ontology for icons, in: International Conference on Signal Processing (ICSP 2012), Vol. 1260-1263, Beijing, China, 2012.

[32] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, P. L. Patel-Schneider, The description logic handbook: theory, implementation and applications, Cambridge University Press, 2007.

[33] J. B. Lamy, A. Venot, C. Duclos, PyMedTermino: an open-source generic API for advanced terminology services, Stud Health Technol Inform 210 (2015) 924–928.

[34] Cornet R, Definitions and qualifiers in SNOMED CT, Methods Inf Med 48 (2) (2009) 178–183.

[35] B. Motik, R. Shearer, I. Horrocks, Hypertableau reasoning for description logics, Journal of Artificial Intelligence Research 36 (2009) 165–228.

[36] I. Horrocks, U. Sattler, Decidability of SHIQ with complex role inclusion axioms, Artificial Intelligence 160 (2003) 2004.

[37] V. Kashyap, A. Borgida, Representing the UMLS Semantic Network in OWL, in: Proceedings of ISWC 2003 (International Semantic Web Conference), Vol. 1-16, 2003.

[38] Guarino N, Some ontological principles for designing upper level lexical resources, in: Proceedings of the First International Conference on Language Resources and Evaluation (LREC), 1998.

[39] D. M. Pisanelli, A. Gangemi, M. Battaglia, C. Catenacci, Coping with medical polysemy in the semantic web: the role of ontologies, Stud Health Technol Inform 107 (2004) 416–419.

[40] J. Geller, Y. Perl, M. Halper, R. Cornet, Special issue on auditing of terminologies, J Biomed Inform 42 (3) (2009) 407–411.

[41] X. Zhu, J. W. Fan, D. M. Baorto, C. Weng, J. J. Cimino, A review of auditing methods applied to the content of controlled biomedical terminologies, J Biomed Inform 42 (3) (2009) 413–425.

[42] A. L. Rector, S. Bechhofer, C. A. Goble, I. Horrocks, W. A. Nowlan, W. D. Solomon, The Grail concept modelling language for medical terminology, Artif Intell Med 9 (1997) 139–171.

[43] L. F. Soualmia, C. Golbreich, S. J. Darmoni, Representing the MeSH in OWL: towards a semi-automatic migration, in: Proceedings of the International Workshop on Formal Biomedical Knowledge Representation (KRMeD), Whistler, Canada, 2004.

[44] G. Héja, G. Surján, P. Varga, Ontological analysis of SNOMED CT, BMC Medical Informatics and Decision Making 8 (Suppl 1) (2008) –8.

[45] Y. Wang, M. Halper, D. Wei, H. Gu, Y. Perl, J. Xu, G. Elhanan, Y. Chen, K. A. Spackman, J. T. Case, G. Hripcsak, Auditing complex concepts of SNOMED using a refined hierarchical abstraction network, J Biomed Inform 45 (1) (2012) 1–14.

[46] G. Héja, G. Surján, G. Lukácsy, P. Pallinger, M. Gergely, GALEN based formal representation of ICD10, Int J Med Inf 76 (2-3) (2007) 118–123.

[47] R. Cornet, A. Abu-Hanna, Two DL-based methods for auditing medical terminological systems, in: Proc AMIA Symp, Vol. 166-170, 2005.

[48] H. H. Gu, D. Wei, J. L. Mejino, G. Elhanan, Relationship auditing of the FMA ontology, J Biomed Inform 42 (3) (2009) 550–557.

[49] F. Mougin, O. Bodenreider, Auditing the NCI thesaurus with semantic web technologies, in: Proc AMIA Symp, Vol. 500-504, 2008.

[50] H. Erdogan, E. Erdem, O. Bodenreider, Exploiting UMLS semantics for checking semantic consistency among UMLS concepts, Stud Health Technol Inform 160 (Pt 1) (2010) 749–53.

[51] M. Halper, C. P. Morrey, Y. Chen, G. Elhanan, G. Hripcsak, Y. Perl, Auditing hierarchical cycles to locate other inconsistencies in the UMLS, AMIA Annual Symposium proceedings 2011 (2011) 529–36.

[52] H. Saitwal, D. Qing, S. Jones, E. V. Bernstam, C. G. Chute, T. R. Johnson, Cross-terminology mapping challenges: a demonstration using medication terminological systems, J Biomed Inform 45 (4) (2012) 613–25.

[53] C. Rosse, V. Mejino JL, A reference ontology for biomedical informatics: the Foundational Model of Anatomy, J Biomed Inform 36 (2003) 478–500.

[54] L. T. Detwiler, J. L. V. Mejino, J. F. Brinkley, From frames to OWL2: Converting the Foundational Model of Anatomy, Artif Intell Med 69 (2016) 12–21.

[55] N. Bouayad-Agha, G. Casamayor, L. Wanner, Natural language generation in the context of the Semantic Web, Semantic Web 5 (6) (2014) 493–513.

[56] Hirst G, Handbook on ontologies, Springer, 2009, Ch. Ontology and the Lexicon.

[57] S. Konstantopoulos, V. Karkaletsis, D. Vogiatzis, D. Bilidas, Language Technology for Cultural Heritage, Vol. 115-132, Springer Berlin Heidelberg, 2011, Ch. Authoring semantic and linguistic knowledge for the dynamic generation of personalized descriptions.

[58] I. Androutsopoulos, G. Lampouras, D. Galanis, Generating natural language descriptions from OWL ontologies: the NaturalOWL system, Journal of Artificial Intelligence Research 48 (2013) 671–715.

[59] S. Y. Sung, T. Hu, Iconic pictorial retrieval using multiple attributes and spatial relationships, Knowledge-Based Systems 19 (2006) 687–695.

[60] Leemans NE M, VIL: A Visual Inter Lingua, Ph.D. thesis (2001).

[61] H. Maurer, R. Stubenrauch, D. G. Camhy, Foundations of MIRACLE: Multimedia Information Repository, A Computer-supported Language Effort, Journal of Universal Computer Science 9 (4) (2003) 309–348.