LexInfo: A Declarative Model for the Lexicon-Ontology Interface

P. Cimiano [a,*], P. Buitelaar [b], J. McCrae [a], M. Sintek [c]

[a] *Semantic Computing Group, Cognitive Interaction Technology Center of Excellence (CITEC), University of Bielefeld, Germany*

[b] *Unit for Natural Language Processing, Digital Enterprise Research Institute, National University of Ireland, Galway*

[c] *Knowledge Management Dept. & Competence Center Semantic Web, DFKI, Germany*

Abstract

In this paper we motivate why it is crucial to associate linguistic information with ontologies and why more expressive models, beyond the label systems implemented in RDF, OWL and SKOS, are needed to capture the relation between natural language constructs and ontological structures. We argue that in the light of tasks such as ontology-based information extraction (i.e., ontology population) from text, ontology learning from text, knowledge-based question answering and ontology verbalization, currently available models are not sufficient as they only allow us to associate literals as labels to ontology elements. Using literals as labels, however, does not allow us to capture additional linguistic structure or information which is definitely needed as we argue. In this paper we thus present a model for linguistic grounding of ontologies called *LexInfo*. LexInfo allows us to associate linguistic information with respect to any level of linguistic description and expressivity to elements in an ontology. LexInfo has been implemented as an OWL ontology and is freely available together with an API. Our main contribution is the model itself, but even more importantly a clear motivation why more elaborate models for associating linguistic information with ontologies are needed. We also further discuss the implementation of the LexInfo API, different tools that support the creation of LexInfo lexicons as well as some preliminary applications.

Key words: lexicon-ontology interface, lexicon ontologies, ontologies, natural language processing

* Corresponding author. Tel: +49 (0) 521 106 12249
Email addresses: cimiano@cit-ec.uni-bielefeld.de (P. Cimiano), paul.buitelaar@deri.org (P. Buitelaar), jmccrae@cit-ec.uni-bielefeld.de

1 Introduction

Several standards for representing ontologies have been developed in the last decade, in particular RDF Schema ([32,10]) and OWL ([6,36]). While ontologies are logical theories and independent of any natural language [1], a grounding in natural language is nevertheless needed for several reasons:

- When engineering an ontology, human developers will be able to better understand and manipulate ontologies. Associating linguistic information to ontologies (in the simplest form by labels) allows people to ground concepts and relations defined in the ontology with their own linguistic and cognitive system.
- In ontology population, automatic procedures for ontology-based information extraction from text will be able to better link textual data to ontology elements if they are associated with information about the way they are typically realized linguistically.
- When querying an ontology in natural language (see [34] and [7]), we need information about how the words/constructs used in the query map to classes, instances and properties modeled in the ontology.
- When verbalizing an ontology, i.e., generating natural language text for easier human consumption (as in [19], [9]), richer models capturing how concepts and relations can be realized linguistically will be needed.

All the above mentioned scenarios would benefit from a principled approach allowing enrichment of ontology elements with information about how they are realized linguistically. However, the development of models that allow us to associate linguistic information (part-of-speech, inflection, decomposition, subcategorization frames, etc.) to ontology elements (concepts, relations, individuals, etc.) is not as advanced as the corresponding ontology representation languages. While RDF(S) and OWL allow us to associate labels to ontology elements, we argue that this is not enough for actual use of ontologies in connection with human users and textual data as described above. SKOS [39] merely introduces further typology of labels (preferred, alternative, hidden, etc.) and does not go beyond RDF(S) as it only supports the representation of atomic terms without a possibility for representing their linguistic (sub-) structure. However, SKOS was not developed with the aim of associating lexical and linguistic information to arbitrary (domain) ontologies, but with the

(J. McCrae), sintek@dfki.de (M. Sintek).

[1] Some authors have argued that ontologies should be constructed following our understanding of language [52,5]. Irrespective of the way in which ontologies are constructed, they are certainly artifacts engineered for a computer in the first place as outlined also in [41]. Therefore, a grounding in language is not needed by the computer itself—it can not make reasonable use of such a grounding anyway!—but by humans interacting with the ontological structures, the results of a query etc.

goal of producing a data-model (building on RDF(S) and OWL) to represent classification schemas, such as thesauri etc. Thus, by definition SKOS does not fulfill the need for a richer model allowing us to associate linguistic structure to arbitrary domain ontologies (and it was not designed for this purpose).

In this paper we introduce a principled model allowing to associate linguistic information to ontologies with respect to any level of linguistic description and expressivity, i.e., the *LexInfo* model. The main characteristic of LexInfo is that it allows us to represent the connection between an ontology and the way we speak about the different ontology elements in a declarative way [2], such that the information is reusable across systems. The current situation in this respect is that every system mentioned above, be it an ontology population, ontology-based question answering or ontology verbalization system, needs to establish a connection between language and the ontology repeatedly for each ontology the system supports. This situation is clearly undesirable as it does not allow a distribution of effort. A clear modularization of tasks (separating the creation of lexicons from their usage in a particular system) as we envision here would allow to distribute efforts in the sense that some parties could create a lexicon for a specific ontology and make the lexicon publicly available in a machine readable format. Other parties could customize their information extraction system to the ontology in question by searching and downloading an appropriate lexicon from the Web. To realize this vision, we need a sound, principled and declarative model by which we can represent and share ontology lexica. This is the goal we pursue with LexInfo.

LexInfo conceptually builds on three main components: the *LingInfo* [13,14] and *LexOnto* [16] models as well as the Lexical Markup Framework (LMF) [24]. LingInfo and LexOnto were developed independently from each other in previous work, but with similar goals and motivations. The LingInfo model provides a mechanism for modeling label-internal linguistic structure, i.e., lexical inflection and morphological or syntactic decomposition of ontology labels, interpreted as terms. The LexOnto model on the other hand enables the representation of label-external linguistic structure, i.e., predicate-argument structures that are projected by lexical heads of ontology labels and their mapping to corresponding ontological structures. While the two models have the same aim of providing more expressive lexicon models for ontologies, they have focused on rather complementary aspects. The LexInfo model presented in this article combines aspects of both (LingInfo and LexOnto) to yield a rich lexicon model for associating linguistic information with the elements defined in an ontology. The LexInfo model builds on the *Lexical Markup Framework*

[2] What we mean by 'declarative' in this context is that the specification of linguistic descriptions associated to ontology elements is independent of any specific applications or systems.

(LMF) [3] (see [24,27]) as a core and extends it appropriately to accommodate the essential aspects of the LingInfo and LexOnto models. From a more general perspective, we hope that this can provide a basis for future discussion on standardization of lexicon models for OWL ontologies.

The paper is structured as follows. In Sect. 2 we provide an extensive motivation for the work discussed here as well as a comparison with related work. In Sect. 3, we discuss in detail the three building blocks of LexInfo: the LingInfo and LexOnto models as well as the Lexical Markup Framework (LMF). In Sect. 4 we present our merged LexInfo model that combines aspects of both models, using LMF as a glue to bring these models together. Furthermore, in Sect. 5 we present the first implementation of an API for the LexInfo model and discuss both the tool support that is currently available to create ontology lexicons according to the LexInfo model as well as preliminary results from applying the LexInfo model. Finally, in Sect. 6 we draw some conclusions of the work presented and discuss ideas for future work.

2 Motivations and Related Work

In the following, we argue firstly that labels such as specified by the RDFS and OWL standards are not sufficient for the purpose of associating linguistic information with ontologies. Secondly, we also argue that existing related work on the association of linguistic information with ontologies still fails to address the need for richer models to capture the lexicon-ontology interface. In addition, we also formulate a number of requirements which should be fulfilled by models for linguistic grounding of ontologies and discuss how far the different proposals fulfill these.

2.1 Separation Between Lexical and Ontological Layer

RDF(S) and OWL allow us to represent what could be termed a verbal 'anchor' for concepts, properties, individuals, etc. by way of the `rdfs:label` property, which is defined for `Resource` as domain and `Literal` as range. We could use this to specify that the class `River` is typically expressed in natural language with the word *'river'*:

```
<rdfs:Class about="#River">
  <rdfs:label>river</rdfs:label>
</rdfs:Class>
```

[3] The Lexical Markup Framework has been recently accepted as an ISO standard under ISO-24613:2008.

To allow for multilingual representation, the `rdfs:label` property enables the representation of labels with an indication of language, e.g., for English (as above) and for German using language tags:

```
<rdfs:Class about="#River">
  <rdfs:label xml:lang="en">river</rdfs:label>
  <rdfs:label xml:lang="de">Fluss</rdfs:label>
</rdfs:Class>
```

If we additionally want to represent linguistic variants of *'river'*, e.g., the plural *'rivers'*, the RDF data model gives us only one choice, i.e., adding a further independent label, i.e.

```
<rdfs:Class about="#River">
  <rdfs:label xml:lang="en">river</rdfs:label>
  <rdfs:label xml:lang="en">rivers</rdfs:label>
  <rdfs:label xml:lang="de">Fluss</rdfs:label>
  <rdfs:label xml:lang="de">FlÃ¼sse</rdfs:label>
</rdfs:Class>
```

Although RDF(S) thus supports the representation of variants across languages, the way this is done is very unsatisfactory for the following reasons:

- The model does not allow us to capture relations between different labels, e.g., morphological relations such as the fact that *'rivers'* is the plural of *'river'*. Certainly, we could ask ourselves if we want to represent such morphological relations explicitly in an ontology, but this raises the question why we should represent linguistic and morphological variants at all, as these are not ontologically relevant but rather should be attached to a base entry (e.g., the lemma) in a lexicon rather than included in the ontology itself. This is exactly the approach we have realized in LexInfo where a separate ontology structured according to the LexInfo model is used to model the linguistic information.
- When attempting to extend the ontology with such simple information as to which syntactic category (part of speech) a label belongs to, we will fail as the labeling system only allows literals to be attached without further information. This is a serious limitation and could also be overcome by separating the lexical and ontological levels, having lexical entries in a lexicon which point to the ontology.
- The RDF label system rules out the possibility that completely different lexica for a given ontology co-exist. The RDF label system in particular ties the labels very closely to the ontology elements. A modular approach in which the ontology and the lexicon are separated would clearly allow us to have different lexica for a given ontology.

Models such as SKOS that tie the conceptual and the lexical layer together more tightly do not solve the above issues in any way. In fact, SKOS only introduces additional typology for the labels, distinguishing between *preferred*, *alternative* and *hidden* labels. Certainly, we could specify that *"rivers"* is a hidden label for the class `River`, but this does not solve any of the above mentioned problems.

The principled solution in our view is to separate the ontological and lexical information into two different domains of discourse. First we would have the *ontology domain of discourse*, talking about classes, properties, individuals, etc., and then we would have the *lexical domain of discourse* talking about lexical elements as "first-class citizens". This allows us to add linguistic information with respect to any level of linguistic description and expressivity required by applications to the lexical elements in the lexical layer. Hereby, the lexical layer is clearly separated from the ontology domain of discourse except for referencing its elements. A crucial question we address here is how the lexicon layer can be structured as a principled model. LexInfo is our answer to this question.

2.2 More Flexible Coupling Between Ontological and the Linguistic System

The label property for RDF(S) and OWL in essence specifies an $n : m$ relation in the sense that one class, property or individual, etc. can be associated to many labels and on the other hand one label can be ambiguous and refer to many ontology elements (class, property or individual). In essence, the relation between ontology elements O_E of an Ontology O and a set of labels L is specified by the following two functions (specific to a given language S):

- $f_S : L \rightarrow 2^{O_E}$ (i.e., a label can denote different ontological entities)
- $f_S^{-1} : O_E \rightarrow 2^L$ (i.e., a given ontological entity can be represented by various labels)

The sets in the ranges of the above described functions, i.e., f_s and its inverse f_s^{-1}, have what could be termed a *disjunctive interpretation* in the sense that a label l can denote some member of $f_s(l)$ (but not a composition of these), while the same holds for f_s^{-1}, i.e., an ontological element e can be realized lexically as any member of $f_s^{-1}(e)$. Thus, the labeling system in RDFS relies on the fact that there is always a counterpart at the ontological level for each label specified in the lexicon, without allowing a more complex correspondence between a class or property on one side and a "syntagmatic" [4] composition of

[4] Syntagmatic relations are between words in a sentence in sequence, whereas "paradigmatic" relations are between words according to meaning, i.e., between synonyms.

several labels on the other.

The reason a more complex correspondence is needed may be explained with the following example. Let us consider a composite term like the German *'Autobahnkreuz'* (*highway interchange*). We have the following possibilities to associate this term with ontological elements:

- There might be a class `HighwayInterchange` to which *'Autobahnkreuz'* can directly refer to.
- There might be a composite class $Interchange \sqcap \exists locatedAt.Highway$ to which *'Autobahnkreuz'* can point.
- There might be simply the general class `Interchange`, in which case we want to specify that only the second part of the composite term *'highway interchange'*, i.e., *'interchange'* refers to the class `Interchange`.
- There might be both classes `Highway` and `Interchange` represented, in which case we want to specify that the second part of the composite term (*'kreuz'*) refers to the class `Interchange` and the first part (*'Autobahn'*) to the class `Highway`.

It thus seems that we require a flexible system to associate terms to concepts that is sensitive to the way concepts or properties have been modeled and allowing us to assign them to the whole term or to individual parts of a term. Further, we see it as a requirement that this model does not assume that the linguistic and ontological levels are "fully synchronized".[5] Thus, we do not need to add a class just because we want to include the term in the lexicon nor the other way round as in RDFS. It is in this sense that synchronization is not required. For this we need appropriate means to represent the decomposition of terms and to associate ontological entities to terms and their sub-structure. Obviously, this is out of the scope of the RDFS label system, as it does not allow modeling of any of the semantic implications of the morphosyntactic structure of complex labels (i.e., composite terms). Clearly, an approach based only on the use of `rdfs:label` does not allow the semantic implications of such label-internal morphological (in the case of German) or syntactic (in the case of English) structure to be modeled.

2.3 Capturing Syntactic Behaviour

When we speak, we certainly do not do so in telegraphic style using only content words (or labels). Words have a clear syntactic behaviour which to a great extent is determined by their syntactic category (verb, noun, adjective,

[5] Hirst [30] even argues they cannot be synchronized as there are ontological distinctions that are never lexicalized and linguistic distinctions that are ontologically irrelevant.

etc.) The way that sentences are composed crucially depends on the syntactic behaviour of the different words that make up the sentence.

When analyzing language, interpreting it with respect to an ontology (e.g., in information extraction or question answering) or generating language output on the basis of the ontology, it is crucial to have access to information about the linguistic behaviour of words. To illustrate this, let us look at the following properties:

```
<rdf:Property about="#capital">
  <rdfs:domain rdf:resource="#Country"/>
  <rdfs:range rdf:resource="#City"/>
  <rdfs:label xml:lang="en">capital</rdfs:label>
</rdf:Property>

<rdf:Property about="#flowThrough">
  <rdfs:domain rdf:resource="#River"/>
  <rdfs:range rdf:resource="#City"/>
  <rdfs:label xml:lang="en">flow through</rdfs:label>
</rdf:Property>

<rdf:Property about="#locatedAt">
  <rdfs:domain rdf:resource="#City"/>
  <rdfs:range rdf:resource="#Highway"/>
  <rdfs:label xml:lang="en">located at/rdfs:label>
</rdf:Property>
```

Although each property in these examples has been associated with meaningful labels (*'capital'*, *'flow through'*, *'located at'*) this is not sufficient for various reasons:

- Lack of linguistic information about the part-of-speech of the lexical item expressed by the label. Consider, e.g., the `capital` property and assume we want to generate a natural language description for the triple `(Germany,capital,Berlin)`. To prevent a system from generating a sentence like *"Germany capitals Berlin."*, it needs to know that *capital* is a noun and cannot be used as a verb. Capturing part-of-speech information (defining if it expresses a noun, verb, etc.) for labels is thus essential.
- Lack of deeper linguistic knowledge on subcategorization frames [6] that constrain the linguistic constructions in which such labels may appear. Consider the case that we want to generate a natural language description of the triple

[6] A subcategorization frame of a word specifies the number and types of syntactic arguments (subject, direct object, prepositional object, etc.) as well as their linguistic structure (nominal phrase, prepositional phrase, relative clause, etc.) that the word can possibly co-occur with in a sentence.

`(Rhein,flowThrough,Karlsruhe)`. Here we need to know that *flow* is an intransitive verb[7] that requires a prepositional phrase introduced by the preposition *'through'* in order to generate an appropriate sentence like *"The Rhein flows through Karlsruhe"* (provided we also specified morphological information about the verb *'flow'*, in particular that the 3rd person singular is *'flows'*, see the discussion on inflection above).

- Lack of ways for capturing the variation in relation expression, as there are many ways in which a certain relation or property can be expressed in language. Consider, for example, the `locatedAt` relation, which can be expressed by *"The A8 passes by Karlsruhe"*, *"The A8 connects Karlsruhe"*, *"The A8 goes through Karlsruhe"*, etc. Although we would not necessarily want to add *'pass'*, *'connect'* and *'go'* as labels to the *locatedAt* property, we may want to express that all of the corresponding verbal forms are valid ways of expressing the `locatedAt` property.
- Lack of ways for expressing how and in which order linguistic arguments of a certain verb map to corresponding semantic arguments of a predicate as modeled in the ontology. For example, given a transitive verb such as *connects*, we may want to specify that its linguistic subject maps to the range of the `locatedAt` property and its direct object to the domain, as in *[The A8: subject] connects [Karlsruhe: direct object]*, which would map to the triple `(Karlsruhe,locatedAt,A8)`.

We see also from our examples above that properties can be realized by various constructions, e.g., verbal constructions (*flow through*), by way of relational nouns (*capital of*), but also by way of participle constructions (*located at*). The linguistic information necessary for analysis and generation purposes obviously differs for the different syntactic categories. Capturing these differences and modeling the relevant information by introducing appropriate classes is an important issue for any model allowing us to associate linguistic information to ontologies.

2.4 Requirements

Given the above explanations, it is clear that more expressive models than those currently available are needed to associate linguistic information with ontology elements, particularly with properties. In general, we derive from the discussion above at least the following specific requirements on a richer model for grounding linguistic information in ontologies:

(1) We require that the model fosters a clear **separation** and **independence** between the ontological and linguistic levels. Separation is important to

[7] Transitive verbs (e.g., *'love'*) require both a subject and a (direct) object, while intransitive verbs do require only a subject but no direct object (e.g., *'sleeps'*).

allow different lexica for one ontology to co-exist, while independence is important to ensure that the different levels (ontological and linguistic) do not constrain nor restrict each other.

(2) We require a model that allows us to express **(structural) information about linguistic realization** with respect to any level of linguistic description and expressivity required by applications. For this we clearly need a separate domain of discourse where lexical entries are first-class citizens and arbitrary complex information can be attached to them. This includes part-of-speech, morphological information, etc.

(3) The model should be able to model the **morphological or syntactic decomposition** of complex terms, allowing the semantics of the single components to be specified with respect to ontological entities (classes, properties, etc.)

(4) As lexical elements never appear in language in isolation but interact with other words in a variety of ways (through syntagmatic relations), we need to capture also their **syntactic behaviour** as well as how this syntactic behaviour translates into ontological representations and structures defined in the ontology.

(5) It should allow **the meaning of linguistic constructions** to be specified with respect to an **arbitrary (domain) ontology**.

Further, a lexicon model for ontologies should fulfill also the following general requirements (compare [43]):

- **support for multilinguality**: allowing us to represent complex lexical entries for multiple languages,
- **accessibility**: supporting the querying, updating and navigation of the model,
- **interoperability**: building on standards that allow models to be shared.

Our standpoint here is that **accessibility** and **interoperability** will be fulfilled by any model building on some standard representation language (RDFS, OWL, UML, XML, etc.) having appropriate tool support. The requirement for multilingual representation can be accommodated by most of the models we discuss below, though it has not always been a focus. In the simplest case, multilinguality can be taken into account by attaching language information (e.g., by way of a language tag) to every lexical entry. LIR, a more elaborate model for representing multilingual information has been presented in [40].

2.5 Related Approaches

In what follows we briefly discuss some related approaches and state whether they fulfill the requirements we have defined. Table 1 summarizes this discus-

	1) separation and indep.	2) struct. ling. information	3) syntactic behaviour	4) morph. decomp.	5) arbitrary ontologies
RDF/OWL	No	No	No	No	Yes
SKOS	No	No	No	No	n.a.
LMF	No	Yes	Yes	Yes	No
LIR	Yes	Yes	No	No	Yes
LW	?	No	No	No	?
Penman GUM	Yes	Yes	Yes	?	Yes
OntoWordNet	No	Yes	No	No	No
LingInfo	Yes	Yes	No	Yes	Yes
LexOnto	Yes	No	Yes	No	Yes
LexInfo	Yes	Yes	Yes	Yes	Yes

Table 1
Requirements 1–5 fulfilled by the different models

sion:

- **SKOS**: The Simple Knowledge Organisation System (SKOS) essentially defines a formal data-model for informal concept schemas such as thesauri and taxonomies by use of the RDFS and OWL vocabularies [38,39]. The main focus of SKOS is on exploiting the RDFS and OWL data-models to model the relations that are typically used in such resources but lack a formal interpretation. For this purpose SKOS defines relations such as `skos:broader` and `skos:narrower` on the basis of the syntax and formal semantics of the RDFS and OWL vocabularies. Although the representation of (multilingual) terms is a shared objective, the aims of SKOS differ compared to ours as our aim is to design a model which allows us to associate linguistic information to arbitrary ontologies, while SKOS mainly uses RDF and OWL (as a data model) to represent classification schemas such as thesauri, technical vocabularies, etc. In addition to RDFS and OWL, SKOS allows labels to be modeled in different flavors, i.e., as a preferred label (`prefLabel`), as an alternative label (`altLabel`) or as a `hidden label`—used to capture information for text mining and not visible to the person inspecting the model for example. SKOS also incorporates multilingual support by allowing language tags to be assigned to the labels. In addition, it allows us to model the taxonomic structure of the resource in question using `narrower` and `broader` properties.

  With respect to our criticism that plain labels without any further linguistic structure are not enough, SKOS does not add anything beyond RDFS and OWL. Clearly, SKOS fails on our requirements 2–5 as it was clearly not designed to specify the meaning of linguistic constructions with respect to an arbitrary ontology (req. 5). Neither does it aim for a clear separation between the knowledge representation and the linguistic levels (thus failing on req. 1).
- **LMF**: The *Lexical Markup Framework* (LMF) aims to provide a meta-model

as a standard framework for modeling and representing computational lexicons such as WordNet [21], the SIMPLE lexicon [33] and others, which is similar to the aims of SKOS to provide a standardized framework for modeling and representing thesauri. LMF clearly fulfills requirements 1–4, but fails on requirement 5 as it does not attempt to establish any connection with domain ontologies, but instead stops where the lexical semantics of words stops.

- **Linguistic Watermark:** The *Linguistic Watermark* (LW) (see [42]) is an *"ontological and software framework for describing, referring and managing heterogeneous linguistic resources and for using their content to enrich and document ontological objects"*. In essence, the LW framework includes a meta-model in order to describe linguistic resources *"from simple synonym dictionaries, to complex resources such as WordNet"*. In this sense it is quite similar to the LMF framework, striving for a uniform model for representing linguistic resources with the goal of interoperability. An aspect which clearly distinguishes the LW from the LMF is that the former clearly aims at connecting/integrating the ontological information with the linguistic one. This connection comes in two flavors: i) integration proper where parts of the linguistic resources are directly "imported" into the ontology, and ii) linking of concepts to so called `SemanticIndex`es (e.g., a WordNet synset) with the aim of "documenting" the concept's meaning. With respect to our requirements, it seems that the Linguistic Watermark clearly fails on requirements 2, 3 and 4. With respect to requirement 1, it is not clear in how far the linguistic and the ontological level are really separated as the LW Suite allow WordNet sub-trees to be imported into the ontology (thus clearly mixing both levels). Further, it can be expected that the LW allows arbitrary ontologies to be taken into account but as this is not completely clear we refrain from filling these fields in Table 1.
- **LIR**: The Linguistic Information Repository (LIR) [43] is a model inspired in the LMF model for associating lexical information to OWL ontologies. The main goal of LIR is to provide a model allowing to enrich the ontology with a lexico-cultural layer allowing to capture the language-specific terminology used to refer to certain concepts in the ontology as well as to capture variations for different languages. The LIR model has focused on multilingual aspects as well as on capturing specific variants of terms (such as abbreviations, short forms, acronyms, transliterations, etc.) which are all modeled as subclasses of the property `hasVariant`. To account for multilinguality, the classes `LexicalEntry, Lexicalization, Sense, Definition, Source` and `UsageContext` are all associated to a certain `Language` to model variants of expression across languages. It also allows to document the meaning of certain concepts in different cultural settings. LIR certainly fulfills requirements 1 and 5, but certainly not requirements 2, 3 and 4.
- **The Penman Generalized Upper Model (GUM)**: aims at simplifying the mapping between language and knowledge by introducing a level

of linguistically motivated knowledge organization (see [3]). The categories modeled in the Penman Upper model are linguistically motivated in the sense that they constrain the linguistic realizations of knowledge. It relies on a classification-based paradigm in which the classes, relations, etc., which are relevant to a given domain are assumed to be classified with respect to the linguistically motivated semantic categories of the Penman upper model. The knowledge organization level of the Penman upper model is thereby assumed to provide a domain-independent, reusable knowledge organization that is valid across domains. According to the rationale of the Penman project, domain experts should not be required to model linguistic expressions, but only link their own models to a general level of knowledge organization that is linguistically motivated but keeps the linguistic details hidden. As described in [4], the Penman upper model contains about 200 categories. The main goal of the Penman model is to ease the generation of text from knowledge models. While the Penman model seems to fulfill all requirements 1–5 in principle, it remains however unclear how variants of expression (relevant for analysis and generation) are specified in the Penman model. This is a crucial aspect of the LexInfo model.

- **OntoWordNet**: OntoWordNet [**?**] is a project that attempts to take the existing WordNet lexicon and make it into an ontology, by conforming it with the upper level model DOLCE [25]. In particular this consists of several tasks: identifying WordNet synsets as classes, individuals or relations; aligning the top of the WordNet hierachy to DOLCE; consistency checking and adding extra domain relations. This approach does not offer separation between the ontological and lexical layer and works only for a single ontology.

3 Building Blocks

In the following, we discuss the three main building blocks of LexInfo: LingInfo, LexOnto and the LMF.

3.1 LingInfo: Multilingual Terms and Morpho-Syntactic Information

3.1.1 Basic Idea

LingInfo [14,13] was developed as an ontology-based lexicon model that allows an integrated but modular approach to the representation of (multilingual) terminology for ontology classes in particular. LingInfo defines a lexicon model where terms can be represented as objects that include lexical information, morpho-syntactic decomposition and point to semantics as defined

by a (domain) ontology. Consider for instance the previously discussed example *'Autobahnkreuz'* (highway interchange). This term can be linguistically decomposed into the following morphological stems *'Autobahn'* (highway) and *'Kreuz'* (interchange), each of which can be linked to lexical information and semantics as expressed by a domain ontology class. Even more complex examples of this can be found for instance in medical terminology: *'muscular branch of lateral branch of dorsal branch of right third posterior intercostal artery'*.

This complex term corresponds to a complex *nominal phrase* and can be linguistically decomposed into the following sub-phrases, where each (sub-) phrase may in turn express an ontology class:

```
sub-phrase 1        muscular branch
sub-phrase 2        lateral branch
sub-phrase 3        dorsal branch
sub-phrase 4        right third posterior intercostal artery
```

The LingInfo model has been developed to represent this kind of morpho-syntactic information on (multilingual) terms for ontology classes and properties. Among the requirements listed in Sect. 2 above, the LingInfo model therefore clearly addresses requirements 1 and 2, as well as 4 and 5 as the meaning of decomposed terms will be represented with respect to a domain ontology, clearly separating linguistic knowledge on these terms. The LingInfo approach in effect integrates a domain-specific multilingual wordnet into the ontology, although importantly, the original WordNet model does not distinguish clearly between linguistic and semantic information whereas the LingInfo model is exactly based on this distinction.

3.1.2 Design

LingInfo supports the representation of linguistic information that is needed to handle the cases discussed above, which includes: *language-ID* (ISO-based unique language identifier), *part-of-speech* (of the head of the term), *morphological and syntactic decomposition*, and *statistical/grammatical context models* (linguistic context represented by N-grams, grammar rules, etc.). The LingInfo model supports the association of such information with ontology elements by way of a meta-class (`ClassWithLingInfo`) and a meta-property (`PropertyWithLingInfo`) which are instantiated by the class or property in question. This allows to link these classes and properties to instances of the class `LingInfo` which represents the linguistic features of the class or property. Figure 1 shows an overview of the model with example domain ontology classes and associated LingInfo instances. The domain ontology consists of the class `Highway` with parts `HighwayLane` and `HighwayInterchange`, each of

which are instances of the meta-class `ClassWithLingInfo` with the property `lingInfo` pointing s to the respective `LingInfo` objects.

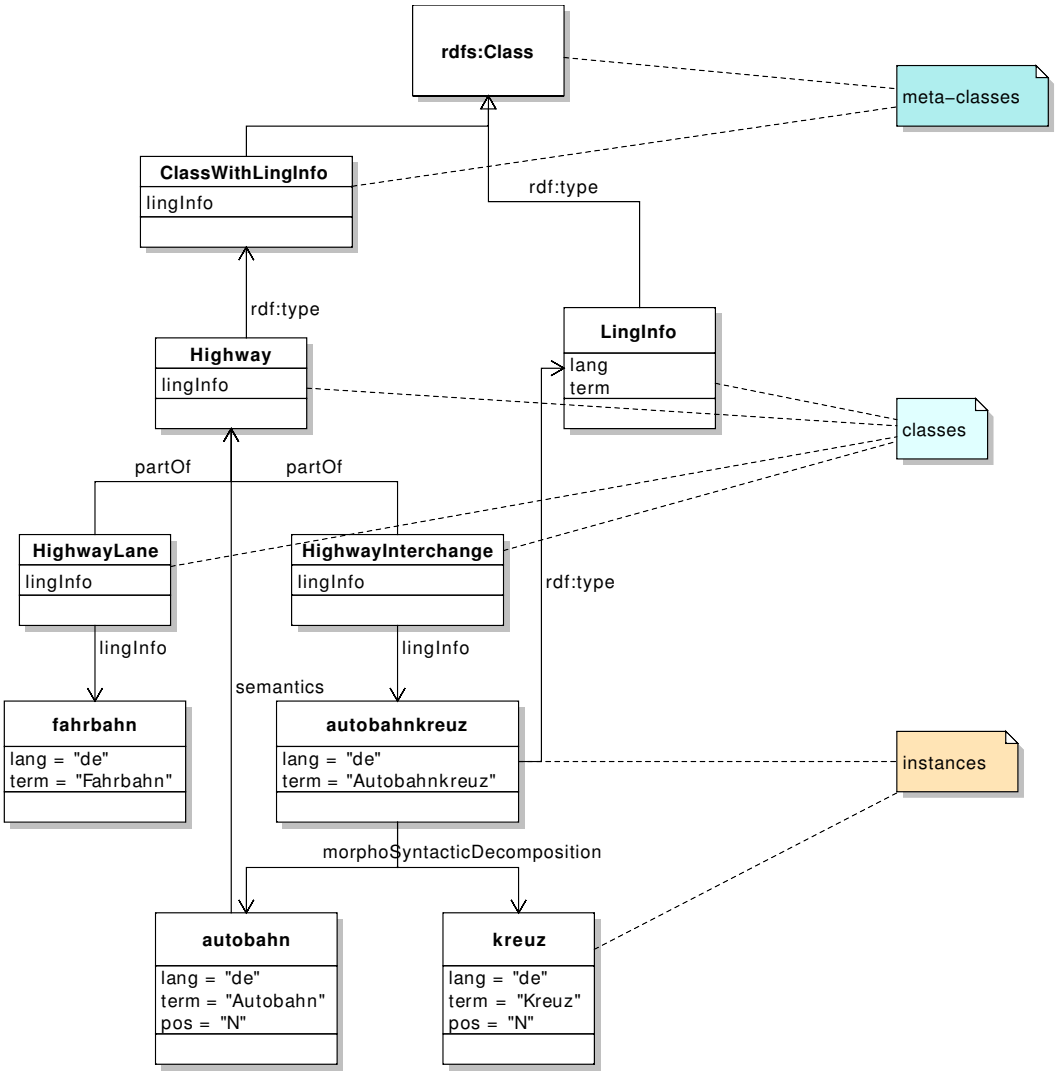

Fig. 1. LingInfo model with example domain ontology classes and LingInfo instances (simplified)

3.2 LexOnto: Representing Syntactic Behaviour

3.2.1 Basic Idea

As already discussed in previous sections, words do not appear in isolation in natural language but enter into a variety of syntagmatic relations with other words which constrain the sentences that can be constructed. When developing richer models allowing linguistic information to be associated with ontologies it is thus crucial to capture the syntactic behaviour of words and the relation between this behaviour and the ontology. This is the goal for

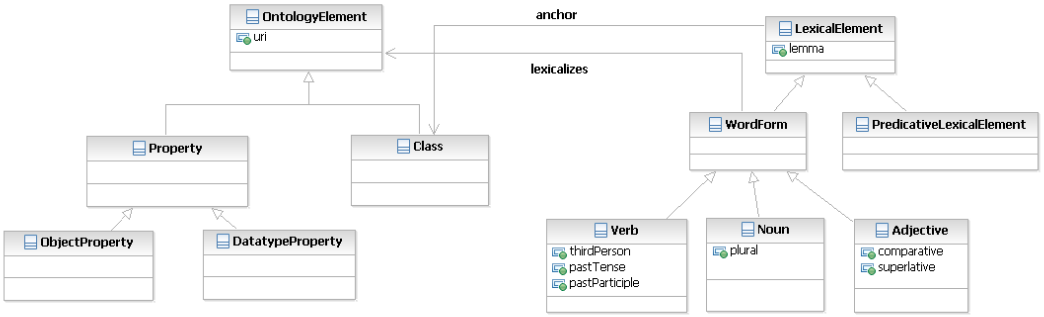

Fig. 2. Main Elements of LexOnto

which the LexOnto model was designed. LexOnto focuses in particular on the representation of the syntactic behaviour of nouns, verbs and adjectives and also on capturing their meaning with respect to a domain-specific ontology. More generally, as any lexicon, LexOnto clearly focuses on the representation of open-class words following the rationale described by Graeme Hirst:

"The words that are of interest [in a lexicon] are usually **open-class** *or* **content** *words, such as nouns, verbs and adjectives rather than* **closed-class** *or* **grammatical function** *words, such as articles, pronouns, and prepositions, whose behaviour is more tightly bound to the grammar of the language."* (see [31]).

3.2.2 Design

At an abstract level, the design of the LexOnto Model is conceptually very simple. The main class of the LexOnto model is the class `LexicalElement`, which has the subclasses `PredicativeLexicalElement` (PLE) and `WordForm` (see Fig. 2). `WordForm`s correspond to nouns, verbs and adjectives as plain words ignoring the predicate-argument structures they project. `PLE`s correspond to predicate-argument structures for verbs, nouns as well as lexical entries for adjectives (see Fig. 3). In order to simplify the representation of the mapping between lexical structures (`LexicalElement`s), there is one single relation `anchor` between a `LexicalElement` and a `Class` (understood as a frame representing the semantics of the `LexicalElement`). This allows the mapping from lexical elements to ontological structures to be represented in a uniform way for all types by pointing to a single class (the so called *anchor class*). For instance, the pattern *"X is capital of Y"* lexicalizes a structure anchored at the class `Country` (because this class is the domain of the `hasCapital` property). While the Y maps to the domain of the property `hasCapital` in the sense that it would fill the subject position of a corresponding triple, X maps to the range of the `hasCapital` property in the sense that the X position will represent the range of the `hasCapital` property. The verbs *write* and *flow (through)* lexicalize a structure anchored at the classes `Document` and `River`,

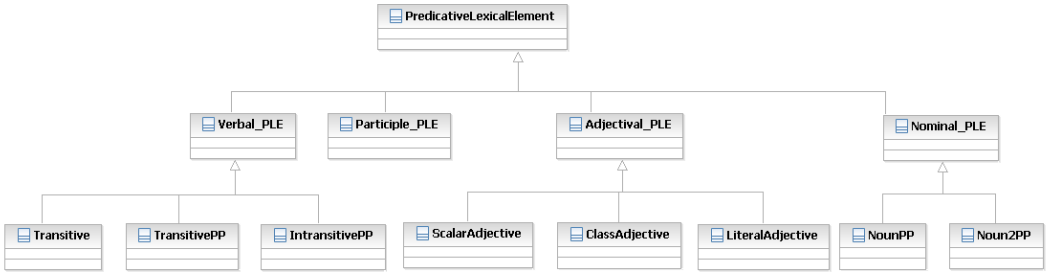

Fig. 3. Predicative Lexical Elements in LexOnto (showing only data-type but no object properties)

respectively (i.e., these are the domains of the properties they refer to, i.e., `hasAuthor` and `flowsThrough`).

`PredicativeLexicalElement`s are in all cases linked to a `WordForm`, e.g., a verbal predicate-argument structure is linked to its head verb, a nominal predicate-argument structure to its head noun etc. As the treatment of verbs, nouns and adjectives in LexOnto has been basically projected to the LexInfo model, we discuss the details in Sect. 4.

3.3 Lexical Markup Framework

The Lexical Markup Framework is a meta-model that provides a standardized framework for the creation and use of computational lexicons, allowing interoperability and reusability across applications and tasks [27]. As the lexicon for an ontology is a special type of computational lexicon, we build on the LMF framework to describe lexica for ontologies. The LMF meta-model is organized in a number of packages (depicted in Fig. 4). The *core package* contains the basic elements of the model and their dependencies (depicted in UML-style notation in Fig. 5).

The central entity in the LMF meta-model is the `Lexical Resource`, which has an associated `Global Information` object capturing administrative details and information related to encoding. A `Lexical Resource` consists of several language-specific `Lexicon`s. A `Lexicon` then comprises of `Lexical Entries` (i.e., words, multi-word entities such as terms and idioms, etc.) which are realized in different `Form`s and can have different meanings or `Sense`s.

Other packages which are of relevance to our work here are:

(1) **morphology extension**: provides a mechanism for describing the morphological structure of lexical entries (extensionally, i.e. for specific examples without supporting the definition of general patterns or rules)
(2) **NLP syntax extension**: allows the syntactic behaviour and properties

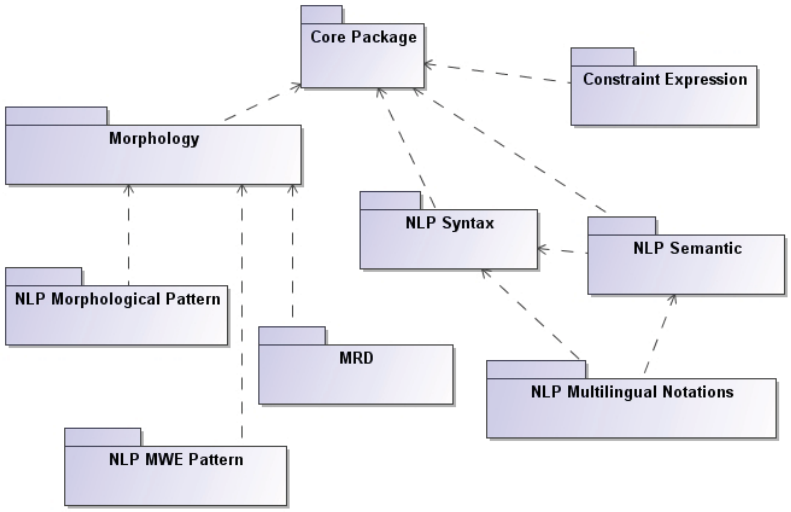

Fig. 4. Package structure of LMF taken from [27]

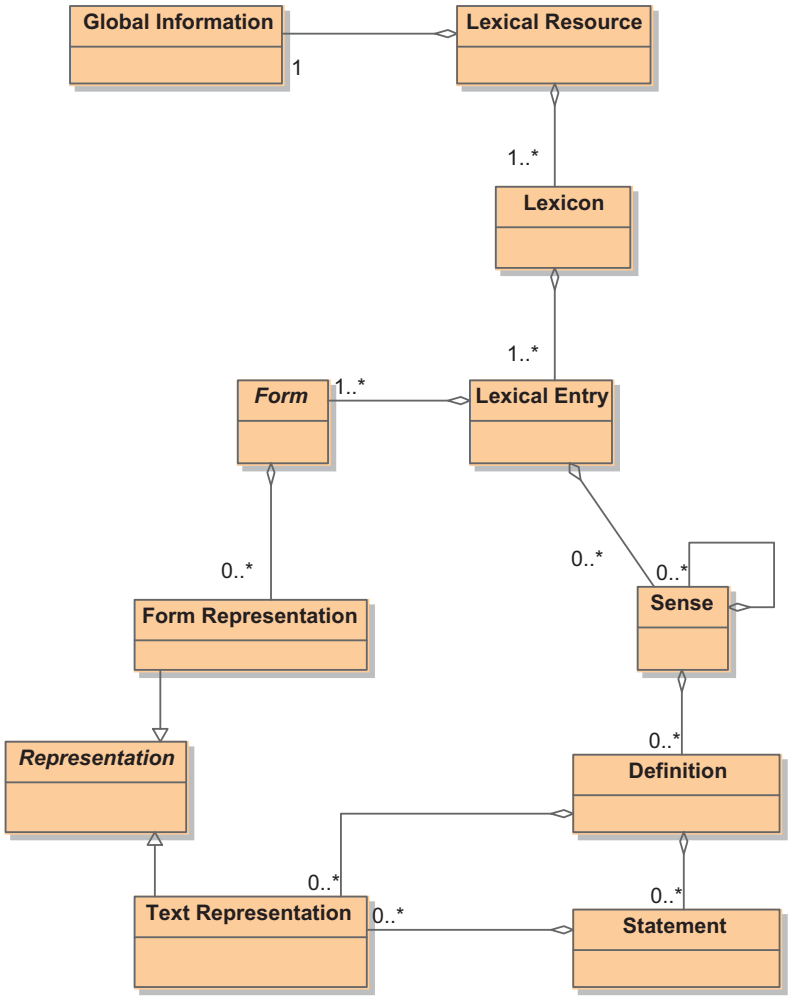

Fig. 5. Core package model taken from [27]

of a lexical entry to be described, in particular the subcategorization frame structure for predicative elements such as verbs etc.

(3) **NLP semantics extension**: provides a way to associate semantic representation structures to syntactic structures, which clearly has a strong relation with the syntax package, allowing semantic predicates to be defined and their semantic arguments to be associated with syntactic arguments of a subcategorization frame.

The other LMF packages are not that important for our current purposes as they cover: i) the intensional definition of patterns for morphological operations (*NLP Morphological Pattern extension*), ii) the representation of data stored in machine readable dictionaries (*MRD extension*), iii) the representation of sense and syntactic behaviour equivalents across languages (*NLP multilingual notations extension*), and iv) the description of the internal structure of a multi-word entity (MWE) (*NLP MWE Pattern extension*). However, it might be the case that these packages become relevant at a later stage of LexInfo development. [8]

At first sight, the LMF model serves our purposes of linguistically grounding ontologies as it clearly distinguishes between the syntactic and semantic levels of description (NLP Syntax vs. NLP Semantic packages). In our case, the NLP Semantic package reflects classes, properties and other ontological structures as described in a (domain) ontology with which we associate the computational lexicon(s). As shown in Fig. 6 and 7 however, both levels are clearly interlinked as described below.

In the syntactic extension, we can model i) the syntactic behaviour (`lmf:SyntacticBehaviour`) of lexical entries in the form of subcategorization frames (`lmf:SubcategorizationFrame`) and their corresponding syntactic arguments (`lmf:SyntacticArgument`), such as subject, object etc. The `lmf:SynSemArgMap`, which is located in the semantic extension package, is the key entity allowing us to associate syntactic arguments to semantic ones. From the semantic point of view, entities of type `lmf:LexicalEntry`, `lmf:SyntacticBehaviour`, and `lmf:PredicativeRepresentation` are all associated to a `lmf:Sense` which captures their (lexical) semantics. For our purposes, the crucial entities are `lmf:PredicativeRepresentation`, `lmf:SemanticPredicate` and `lmf:SemanticArgument`, which will be refined in terms of specific subclasses which allow us to connect to predicates defined in the ontology (e.g., classes and their properties). The essential class for mapping syntactic to semantic arguments is the `lmf:SynSemCorrespondence`

[8] In fact, at the time of writing we are working on integrating the NLP Morphological Pattern extension into the LexInfo API in order to represent generative inflection patterns and avoid representing inflectional variants for all lexical entries explicitly. However, this part of the work is not completed. The LexInfo API is currently still under development and a description of all features is out of the scope of this article.

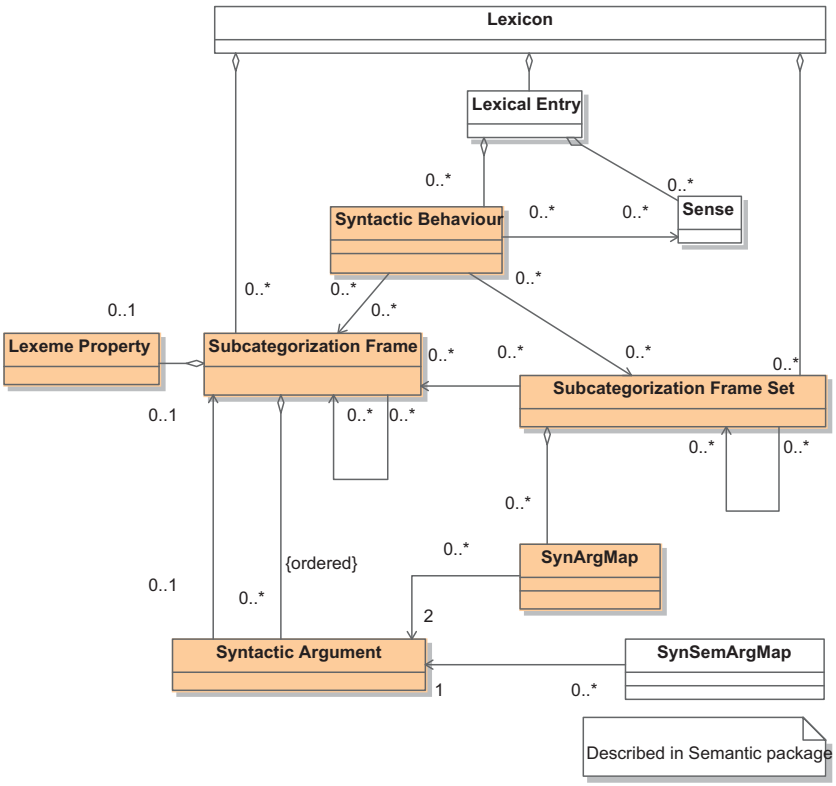

Fig. 6. Structure of the NLP Syntax extension, taken from [27]

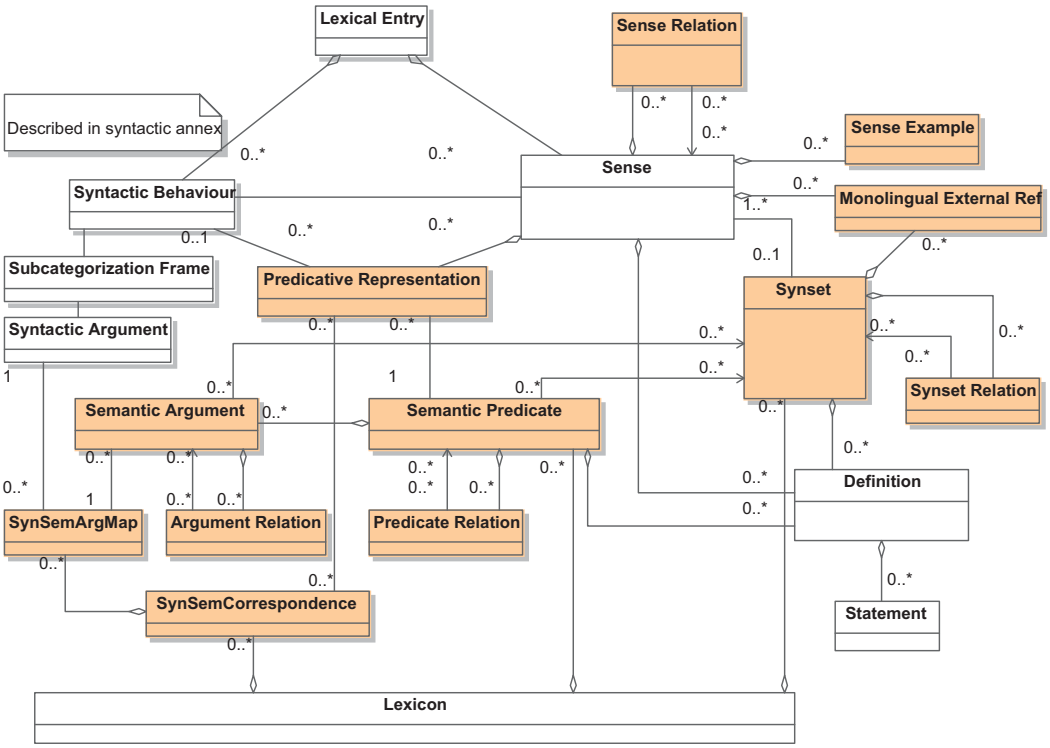

Fig. 7. Structure of the NLP Semantics extension, taken from [27]

class, which consists of a number of `SynSemArgMaps` for mapping specific syntactic arguments of a subcategorization frame to semantic arguments of a `lmf:SemanticPredicate`. In the following section we will describe how the LMF model has been extended to support the association of ontological structures to lexical entries and, conversely, of linguistic information to ontological classes and properties.

4 LexInfo: Enriching Ontologies With Linguistic Information

The starting point for our unifying model for linguistically grounded ontologies are the LingInfo, LexOnto and LMF models discussed in the previous sections 3.1, 3.2 and 3.3, respectively. The glue for the three frameworks will essentially be provided by the LMF model. We proceeded as follows to arrive at our unifying model:

- We downloaded the OWL version of the LMF model available at `http://www.lexicalmarkupframework.org/`.
- As the ontology has been originally created starting from an UML model and only uses the properties `isAssociated`, `isPartOf` and `isAdorned`, we have introduced appropriate subproperties for most of the associations between entities described in the LMF Specification [24]. Further, we have commented most of the ontology classes on the basis of the descriptions of the LMF Specification [27]. The resulting ontology is available for download at `http://lexinfo.net/lmf`.
- Then, we created a new ontology `LexInfo` importing the corrected LMF ontology, introducing our monotonic extensions on top of it. The LexInfo ontology can be downloaded here: `http://lexinfo.net/lexinfo`.

In what follows we discuss how LexInfo meets our requirements 1–5.

4.1 Separation Between Linguistic and Ontological Levels

The separation between the ontological and linguistic levels is achieved in LexInfo by introducing two separate domains of discourse by way of using different name spaces (and different ontologies). On the one hand we have the domain ontology with its own namespace defining the relevant classes, properties and individuals in the given domain. And on the other hand the lexical information is factored into a separate domain of discourse, which is structured according to the LexInfo model. The main entities in the lexical domain of discourse are instances of the class `LexicalEntry`. Figure 8 shows the subclass hierarchy of lexical elements. Figure 9 shows how a `LexicalElement` representing the

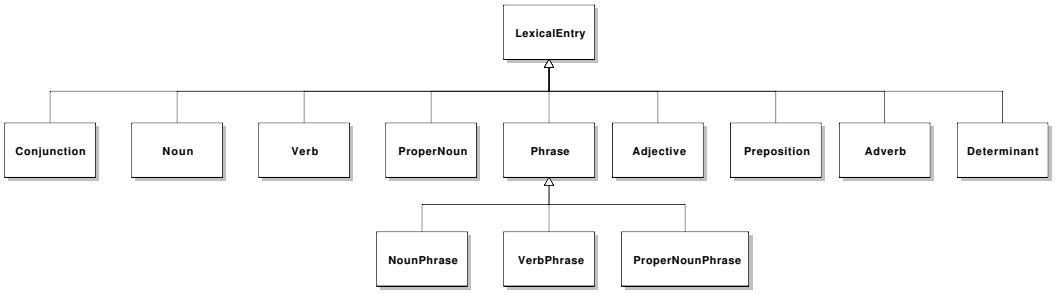

Fig. 8. Subclasses of LexicalEntry

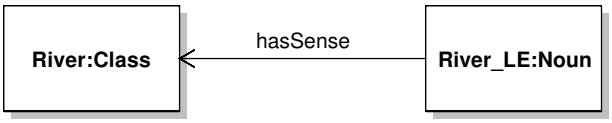

Fig. 9. The Lexical Entry for *'river'* points to the class `River` through the `hasSense` property. The lexical and ontological levels are clearly separated but linked to each other.

`Noun` *'river'* is linked to the class `River` defined in the ontology by pointing via the `hasSense` [9] property to an individual of the OWL2 meta-ontology standing proxy for the class. The ontology elements are reified as by building on a meta-mode of OWL [51] so that we can refer to them as individuals.

4.2 Linguistic expressivity

The separation between the linguistic and the ontological levels also enables linguistic information to be attached to lexical elements for any level of required linguistic expressivity. In particular, LexInfo allows for a high degree of linguistic expressivity by attaching part-of-speech and morphological information to lexical entries. This is in contrast not possible in RDF(S), OWL or SKOS, which restrict the range of the `label` property to `Literal`. Part-of-speech information is attached to lexical entries by specifying the lexical entry as an instance of classes such as verb, noun, adjective, etc. This is done according to the subclass hierarchy of `LexicalEntry` (see Fig. 8).

Morphological information and relations between the different morphological variants are captured by directly building on the structure provided by LMF.

[9] A reviewer has pointed us to the fact that we are using the term *Sense* here in an unorthodox fashion (from a lexical semantics point of view. However, let us note that in standard sense-enumerating lexicons (the ones criticized by Pustejovsky in his theory of the Generative Lexicon [**?**] the sense of a word is defined by pointing to one ore more elements in the inventory of sense, as for instance in WordNet. We are using *sense* in this line but assuming that sense inventory is provided by a given domain ontology.

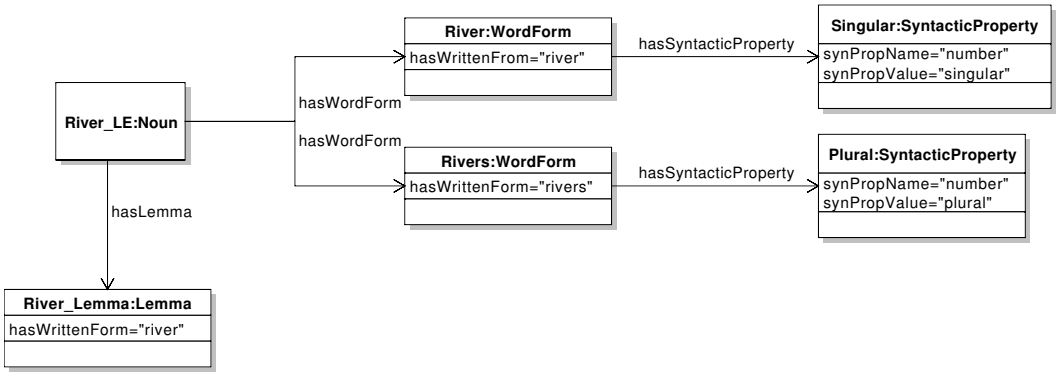

Fig. 10. Modeling *'river'* and its morphological variations using the LMF machinery. Here both the plural form and singular form are represented with appropriate syntactic properties.

Figure 10 shows how information about morphological variants of a word (here *'river'*) are modeled through `WordForms`. In the figure we see how the fact that the plural of *'river'* is *'rivers'* is specified through an instance of `WordForm` which has the syntactic property `Plural` and `hasWrittenForm` value of *'rivers'*.

4.3 Morphological or Syntactic Decomposition of Composite Terms and Multi-word Expressions

The morphological decomposition of terms is done in LexInfo by building on the morphological extension package of LMF, which essentially allows us to associate a `ListOfComponents` with a `LexicalEntry`, which has an ordered list of components (with a minimum of 2) (see [27]). We have modeled this in OWL by introducing an additional data-type property `order` specifying the absolute order of a `Component` within a `ListOfComponents`. Components then point to `LexicalEntries` which can again be composite, thus allowing for recursion. In order to capture how the parts of a compound are associated to the ontology, we build on the general mechanism of LMF allowing LexicalEntries to be associated with a `Sense`, of which `owl2:Entity` is a subclass. [10] In this way, we are able to state that *'Autobahnkreuz'* is composed of two lexical entries where the first refers to the class `Highway` and the second to the class `Interchange` (see Fig. 11). In this sense the LexInfo model thus captures the relevant aspects of the LingInfo model, allowing the morphological decomposition of terms to be modeled and thus fulfilling requirement 4.

Multi-word expressions are modeled in a similar manner by way of the subclass `Phrase` of `LexicalEntry` indicating that the lexical entry is actually a

[10] We build on the OWL2 meta-ontology for this purpose: `http://owlodm.ontoware.org/OWL2`.

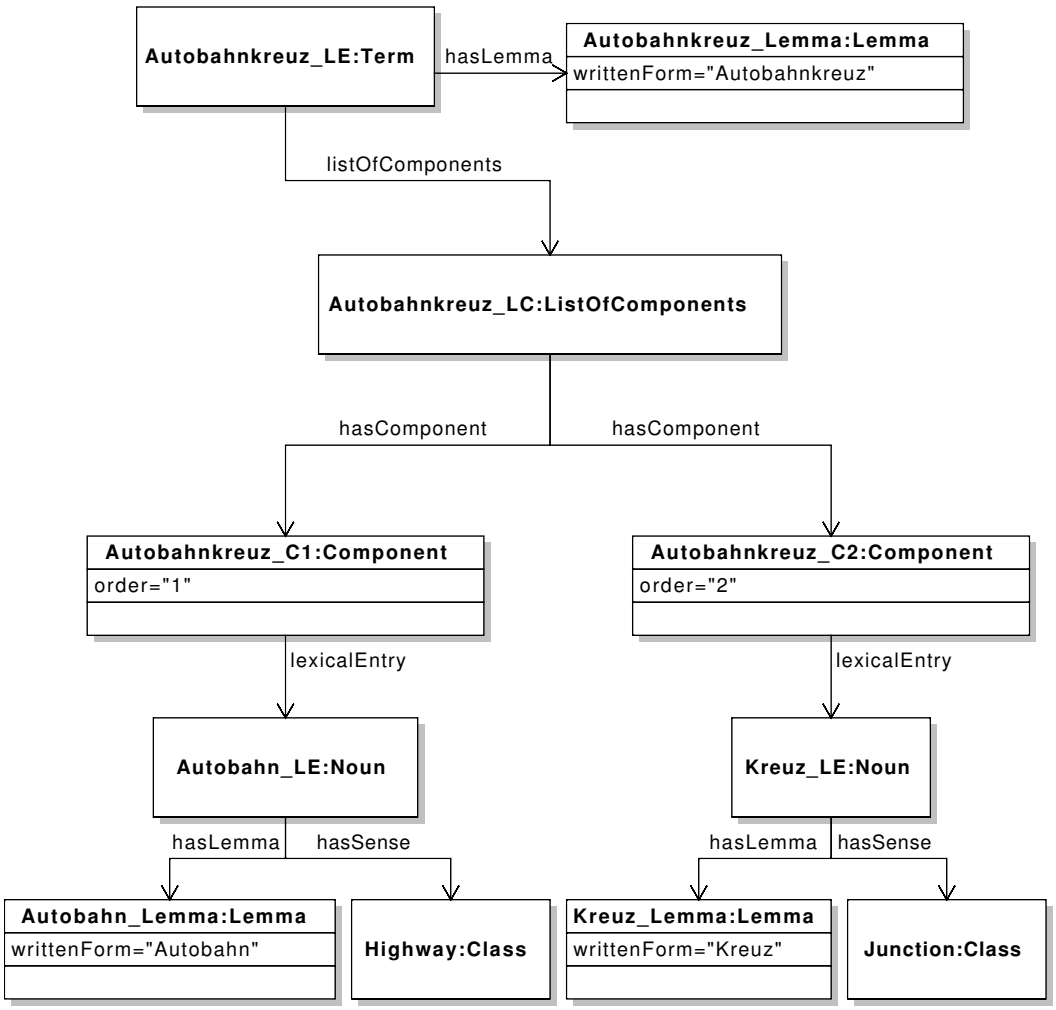

Fig. 11. Example of decomposition (*'Autobahnkreuz'*) with linking to ontology concepts (LexInfo extension). Here we see the lexical entry is decomposed into two components, both of which are lexical entries with their own lemmas.

complex expression. `Phrase` has a number of subclasses, such as `NounPhrase` or `VerbPhrase`, representing a phrase with the head being a noun or verb, respectively. Each phrase is connected to a LexicalEntry representing the head via the property `head`. Each component of the phrase is then modeled in the same manner as for composite terms (see Fig. 12).

4.4 Syntactic Behavior

We have argued already that the representation of syntactic behavior is crucial for any model linking linguistic information to ontology elements. For this purpose, we have extended the LMF model by reusing its classes but refining them in the LexInfo model, in particular introducing the following subclasses of LMF classes:

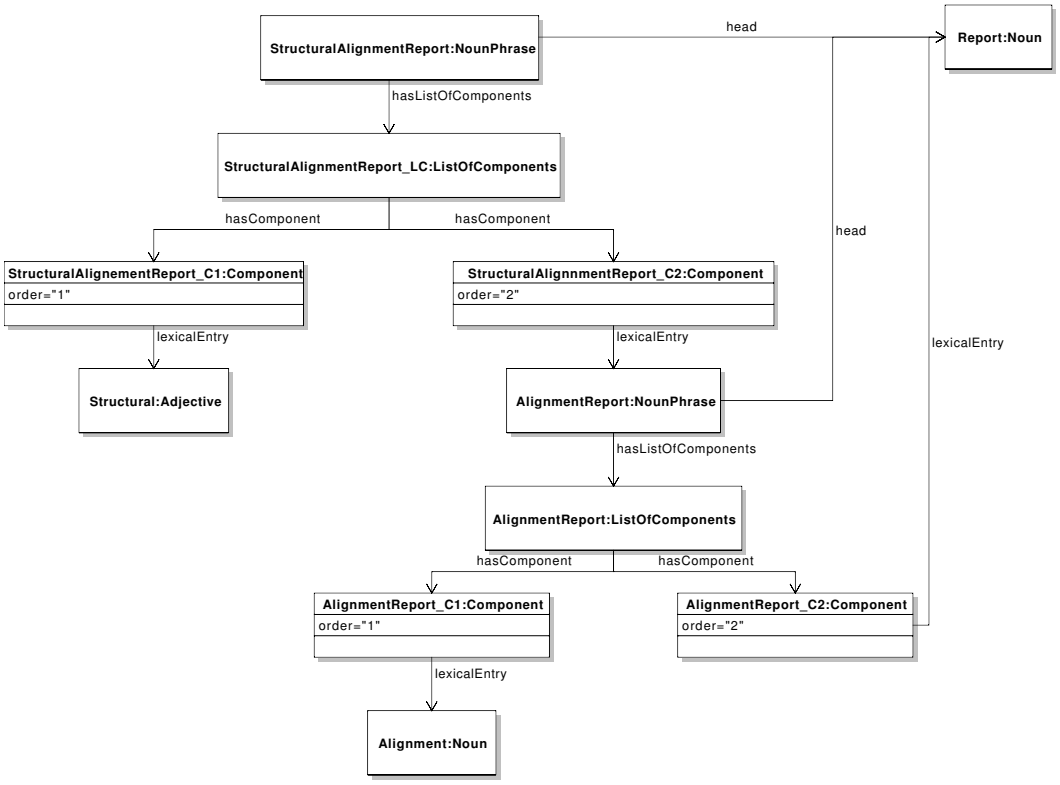

Fig. 12. Example of phrase decomposition of "structural alignment report". Here we see that a tree is created by first decomposing the lexical entry into "structural" + NP, and then the noun phrase is decomposed into "alignment" + NP and the final noun phrase into "report."

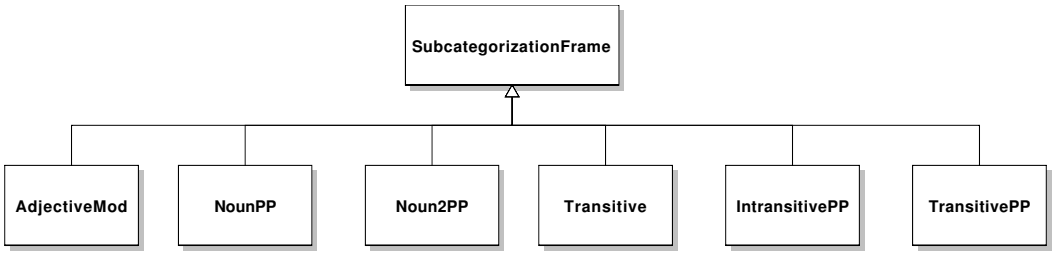

Fig. 13. Subclasses of SubcategorizationFrame

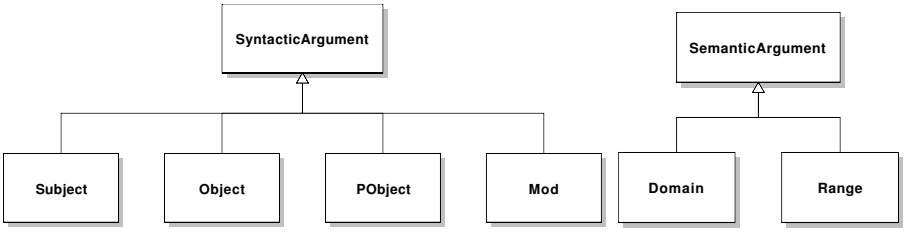

Fig. 14. Subclasses of SyntacticArgument and SemanticArgument

(1) Subclasses of `lmf:LexicalEntry`, i.e., `lexinfo:Verb`, `lexinfo:Noun` etc., which are distinguished by way of attributes in the LMF model (see Fig. 8).

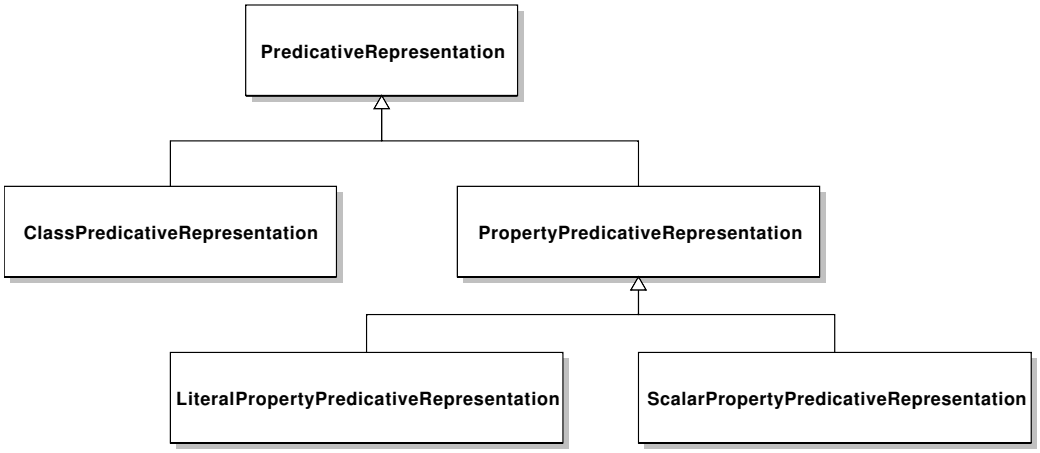

Fig. 15. Subclasses of PredicativeRepresentation

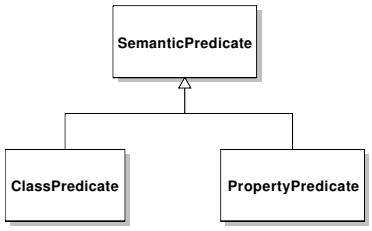

Fig. 16. Subclasses of SemanticPredicate

(2) Subclasses of `lmf:SubcategorizationFrame`, i.e., `lexinfo:Transitive, lexinfo:IntransitivePP`, etc. (see Fig. 13).
(3) Subclasses of `lmf:SyntacticArgument`, i.e., `lexinfo:Subject, lexinfo:Object, lexinfo:PObject`), etc. (see Fig. 14).
(4) Subclasses of the `lmf:PredicativeRepresentation` and `lmf:SemanticPredicate` classes, e.g., the classes `lexinfo:Class-PredicativeRepresentation` and `lexinfo:ClassPredicate` as well as `lexinfo:PropertyPredicativeRepresentation` and `lexinfo:PropertyPredicate` allowing us to refer to a class or property (as predicate), respectively (see Fig. 15 and Fig. 16). In addition, there are subclasses (`ScalarPropertyPredicativeRepresentation` and `LiteralPropertyPredicativeRepresentation`) for representing the behavior of the predicate if the range is valued (i.e., integer, string, etc.).
(5) Subclasses of the `lmf:SemanticArgument` class, i.e., `lexinfo:Domain`, `lexinfo:Range`, etc., as well as appropriate subclasses allowing the semantic arguments of a class to be specified (where properties are understood as slots of the frame represented by the class) (see Fig. 14).

It is important to note that LMF also distinguishes between different types of subcategorization frames. However, the distinction is encoded as an attribute, i.e., *'regularSVO'* for a transitive verb for instance. The advantage of modeling the different subcategorization frames as subclasses (as we have done) is that this allows us to formulate additional axioms, requiring for example that a

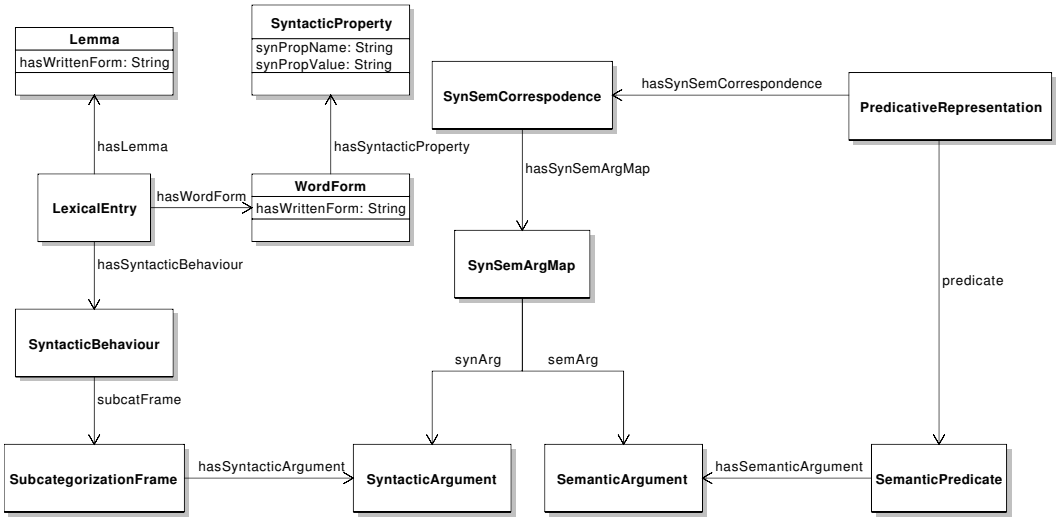

Fig. 17. General Schema of SynSemCorrespondence. In this diagram the syn-sem argument map is used to link the syntactic argument of the subcategorization with the semantic argument of ontology predicate. The mechanisms on the left show the lexical part of the lexicon with the entry, lemma, word forms and syntactic properties. On the right side the semantic mapping includes the predicative representation and the predicate. The `SynSemCorrespondence` has the function of gluing together all the mappings from syntactic to semantic arguments for the given subcategorization frame.

`Transitive` subcategorization frame has exactly two syntactic arguments: a subject and an object. Such general axioms allowing the lexicons to be checked for consistency are clearly not possible in the original LMF model.

There are a number of classes in the LexInfo model that stem from the LMF model and allow us to represent syntactic structures together with their mapping to ontological structures. The main classes defining the syntactic behavior of lexical elements is `SyntacticBehaviour` (linked to `LexicalEntry` through the property `hasSyntacticBehaviour`). Each instance of `SyntacticBehaviour` is associated to an instance of `Subcategorization Frame` (through the property `subcatFrame` which specifies the type of syntactic structure (transitive verb, intransitive verb, etc.) as well as the syntactic arguments it requires to be realized linguistically. This is accomplished through the `hasSyntacticArgument` property linking to various instances of the class `SyntacticArgument` representing the `Subject`, `Object` etc. of the syntactic construction. At the semantic side, an analogous specification of the predicate in question and its semantic arguments is assumed. `PredicativeRepresentation` stands proxy for a class or property defined in the ontology. This proxy relation is captured through the property `predicate` which links a `PredicativeRepresentation` to a `SemanticPredicate` which can be either a `PropertyPredicate` or a `ClassPredicate` pointing to the appropriate instance of the OWL2 meta-ontology representing the corresponding class or property.

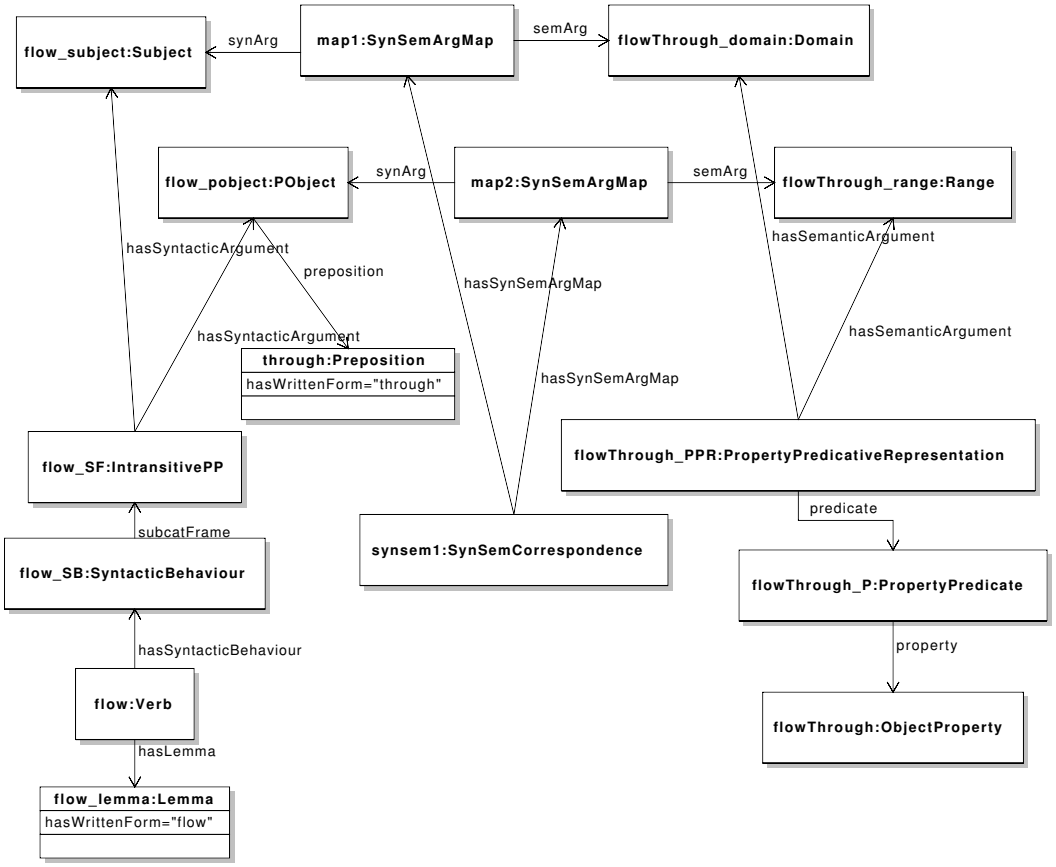

Fig. 18. Modeling of the syntactic behavior of the `LexicalEntry` (`Verb`) *'flow'* together with the mapping to the ontological predicate `flowsThrough` via the `SynSemCorrespondence` class.

The crucial link between the syntactic and semantic levels is accomplished through the class `SynSemCorrespondence`, which is related to one or more `SynSemArgMap`s which establish the correspondence between a syntactic argument in the subcategorization frame (through the property `synarg`) and a semantic argument of the `PredicativeRepresentation` (through `semArg`).

Figure 17 depicts all of the above schematically. We describe how this scheme is instantiated for the different types of part of speech (verbs, nouns, adjectives) in the subsections below.

4.5 Verbs

Verbs inherently express relations between their arguments and project different predicate-argument structures which we can not capture if we only model verbs as plain strings (in the form of labels). In the LexInfo model we thus explicitly represent verb argument structures (so called subcategorization frames) and the way they map to corresponding (domain-specific) ontology

structures (especially properties). In particular, in the model we consider the following verb classes (so far):

- **transitive verbs** (`Transitive`): these are those requiring a subject and a direct object, e.g., *'love'*, *'write'*, etc.
- **intransitive verbs subcategorizing a prepositional complement** (`IntransitivePP`): these are those requiring a subject and a prepositional complement headed by a specific preposition, e.g., *'flow through'*, *'pass by'* etc.
- **transitive verbs subcategorizing a prepositional complement** (`TransitivePP`): those requiring a subject, an object and a prepositional complement, e.g., *'write an article about sth.'*.

The hierarchy for verbal (and nominal) subcategorization frames is shown in Fig. 13. We illustrate the interplay between the different classes on the basis of the example of the subcategorization frame for the verb *'flows'* which requires a subject (i.e., that what flows) as well as a prepositional object introduced by the preposition *'through'*. Obviously, it is possible that a verb projects different subcategorization frames and syntactic behaviors. Figure 18 shows how the link between the syntactic behavior of the verb flows (through) and the ontology is modeled. First of all we have a `LexicalEntry` (a `Verb` more specifically) representing the verb itself. The verb has a base form or lemma *'flow'*. Further, the lexical entry for *'flow'* is related to an instance of `SyntacticBehaviour` representing one of its possible syntactic realizations. The instance of `SyntacticBehaviour` is then linked to an instance of type `SubcategorizationFrame` (an `IntransitivePP` in this particular case). The modeling makes explicit that this subcategorization frame requires two syntactic arguments: a `Subject` and a `PObject`. At the semantic/ontological level we introduce an instance of the class `PredicativeRepresentation` which will stand proxy for the object property `flowsThrough` with a `River` as `Domain` and a `Location` as `Range`. The instance of `PropertyPredicativeRepresentation` is linked to an instance of `PropertyPredicate` (through the property `predicate`), which in turn points to an object property defined in the domain ontology via the property `property`. The `PropertyPredicativeRepresentation` has two arguments: the `Domain` and the `Range` of the property. The link between syntax and semantics is achieved via an instance of the `SynSemCorrespondence` class which is related to two `SynSemArgMap`s via the `hasSynSemArgMap` property. One instance of `SynSemArgMap` links the subject of the subcategorization frame to the domain of the property `flowsThrough`, while another instance of `SynSemArgMap` links the object syntactic argument to the range of the predicative representation of `flowsThrough`.

In principle, there is no limit to the classes considered and the model can be extended to cover new types at any time. However, so far we have limited

the model to the most frequently occurring ones. The model will be certainly extended in the future in line with needs of future applications. Further, we intend to hook up the model to existing linguistic categories and resources as our goal is not to reproduce work on standardization of linguistic categories conducted in the computational linguistics community in particular under the auspices of ISO [11] .

4.6 Nouns

Nouns have been noted in general to correspond to ontology classes, but noun phrases can have an internal structure and subcategorize one (or more) prepositional phrases: e.g., *the mother of*, *the capital of*, *the flight from A to B*, *the distance between X and Y*, etc. Thus, nouns subcategorizing for a prepositional complement are modeled in LexInfo by associating it to a corresponding subcategorization frame. Such relational nouns thus represent a relation or a property rather than a single class. We have shown in Fig. 9 how simple nouns representing a class are represented in LexInfo. In this section we discuss in detail how relational nouns which actually represent a property are modeled. Relational nouns are modeled in LexInfo by way of nominal subcategorization frames that require a prepositional object as well as an external subject over which the property is predicated. For example in the case of the syntactic construction *'X is the capital of Y'*, the head of the prepositional object, i.e., Y, represents the domain of the property `capital`, while the X is realized here as the subject of a copula construct [12] and represents the range of the property `capital`.

4.7 Adjectives

Adjectives subcategorize syntactically the noun they modify and semantically the entity which the property expressed by the adjective is predicated over. Each adjective is assumed to have the subcategorization frame `AdjectiveMod` indicating that it modifies another lexical entry and the modified syntactic argument is denoted by the argument type `Mod`. We model the semantics of adjectives relative to properties, normally by defining a set of permissible object values for the adjective, i.e., we may say that 'big' can be applied to all cities with more than 100,000 inhabitants. We then split the semantic behavior into different forms by having subclasses of `PredicativeRepresentation` and

[11] These works are called in a data category register called ISOcat and available at `http://www.isocat.org/`

[12] In linguistics, a copula is a word used to link the subject of a sentence with a predicate, e.g., John is nice.

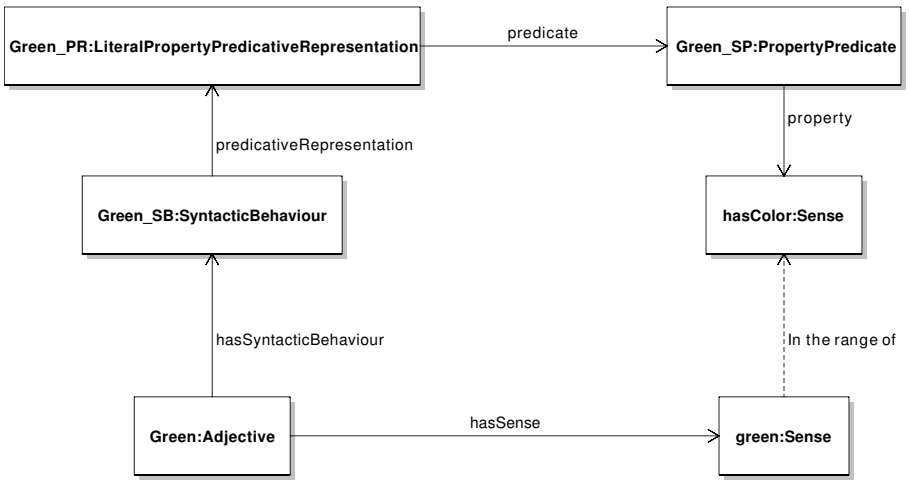

Fig. 19. Modeling the adjective 'green' and its relation to the ontology. Here we see that the literal predicate matches to a value green which is an instance of the range of the property hasGreen, represented here as pseudo-property "in the range of".

`SemanticPredicate`. In particular we define the following specific types of adjectives:

- **literal adjective**: This is used when the adjective models the value of an object property. The standard example are colors, for example if there was a property `hasColor` with a range defined as the class `Color`, with `green` as an instance of this class, then we would model *'green'* as corresponding to this individual when used in the given relation. We model this as a `LiteralPropertyPredicativeRepresentation` and a semantic predicate `PropertyPredicate`, and we associate the `LexicalEntry` for *'green'* with the individual `green` using the `hasSense` property and the `PropertyPredicate` with the property `hasColor` using the `property` property. This modeling is shown in Fig. 19. We have to admit that alternative formalisations of the semantics of colour adjectives are possible, but this is not important here. In fact, LexInfo is agnostic with respect to which semantic theory and modelling to adopt, as long as the corresponding predicates are defined in the corresponding ontology. The important point that we want to stress is that LexInfo has been designed in such a way that different formalisations of the semantic of adjectives can be represented. The specific choice of semantic representation is certainly left to the designers of the ontology and/or lexicon.
- **class adjective**: A class adjective is associated to a class in the ontology. This is modeled by way of the predicate representation `ClassPredicativeRepresentation` and the semantic predicate `ClassPredicate`. An example here is the adjective *"female"* which expresses the membership in the class `Female`. In essence class adjectives are equivalent to literal predicates, but the class membership predicate (e.g., `rdf:type`) is used in place of an object property. As such we can say that

the semantics of *'female dog'* is that it is a `Dog` that is also in the class `Female`.

- **scalar adjective**: A scalar adjective models a data-type property, whose range is totally ordered. Ideally if we have a data-type property like `size` then we would like to define *'large'* relative to it as subset of the property range, i.e., a city is large if it has `size` in the set $\{x : x \geq 100,000\}$. As with literal adjectives we have `PropertyPredicate` that refers to the data-type property and we have a `ScalarPropertyPredicativeRepresentation`. For open domain subsets, e.g., $x \geq 100,000$, we add two properties to the `ScalarPropertyPredicativeRepresentation`, namely `dataValue` indicating the minimum or maximum value of this range, and `polarity` indicating if this value is the minimum or maximum. This means that we also have a natural modeling of comparative and superlatives as if the `polarity` is `positive` this mean *'larger'* indicates an increases in the data-type property value. We also allow an optional `contextSense` to be defined, which states that this particular adjective only holds if the subject of the data-type property belongs to a given class. In the example above it is clear that the data-type property `size` may apply to several domains, but that the range corresponding to `'large dogs'` is not the same as the range corresponding to `'large buildings'`.

4.8 Specification of Meaning w.r.t. Ontology

The requirement of allowing the meaning of linguistic constructs to be specified (terms, compounds, subcategorization frames) trivially follows from the way we have conceived the LexInfo model (see particularly Sect. 4.3 for how the semantics of terms and compounds are specified with respect to a domain ontology and Sect. 4.4 for how the ontological meaning of subcategorization frames is represented).

4.9 Multilinguality

The LexInfo model allows us to associate linguistic information in various languages to ontology elements. This support for multilinguality is a byproduct of the decision to build on the LMF as a basis for our model. The assumption in LMF is that lexicons are language-specific artifacts, such that there is one different lexicon for every language supported. While this is a reasonable assumption, other alternatives are possible. For example, we could assign a language tag to every single lexicon entry, having just one lexicon per ontology. By building on the LMF machinery we have however opted for having one lexicon per language, which does not constitute a restriction in any way.

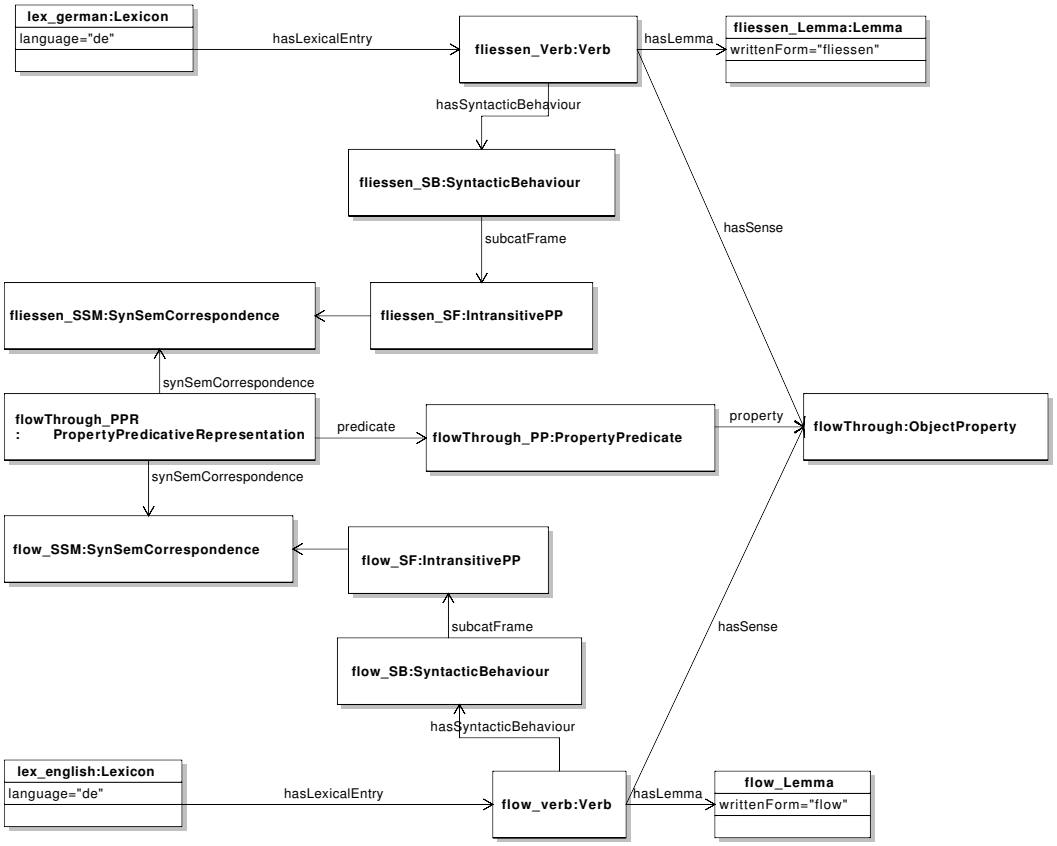

Fig. 20. Modeling multilinguality. On the top we have a lexical entry for the German verb "fliessen" in a German lexicon and on the bottom a lexical entry for the English verb "flow" in an English lexicon. The connection between the two entries is realized by sharing the same ontology sense and the same property predicate representation and semantic predicate.

Following the LexInfo model, the linguistic information for different languages (Lexical Entries, Subcategorization Frames, Syntactic Behavior etc.) is linked to certain ontology elements which are language independent. In this way, while the different variants for expressing a certain class or property are not directly related, they are linked via the ontology (see Fig. 20). The figure shows how realizations in different languages (*"flows through"* in English and *"fliesst durch"* in German) are linked via `SynSemCorrespondence` objects to the same `PropertyPredicativeRepresentation` object representing the `flowsThrough` object property. The fact that there is no direct link is reasonable as the exact relation between different linguistic realizations might be unclear (they will not always be translation equivalents), while they might certainly express the same meaning with respect to the vocabulary of the ontology.

While the LingInfo model was originally implemented in the RDF(S) language [14], LexOnto was already created using OWL-DL. The version of LexInfo described in this paper [13] has been implemented in OWL-DL. In this section we discuss the different possible choices for implementing LexInfo, together with their advantages and disadvantages:

- RDF(S): In RDF(S), Classes are themselves resources, so that we can attach linguistic information to classes by treating them as individuals. This is exactly the way that linguistic information was attached to classes and properties in the LingInfo model (by introducing the meta-classes `ClassWithLingFeat` that has other classes as instances, allowing the linguistic information to be attached). This modeling allows combined reasoning over the lexicon and the ontology, but is limited in expressivity as it is restricted to the expressiveness of RDFS.
- OWL-DL (with ontology meta-model): While it is forbidden to use the same URI to refer to an an individual and a class in OWL-DL, this can be done if we keep the lexical and ontological layers separate, thus not allowing reasoning over both ontologies together. This is exactly the way that linguistic information has been modeled in the LexOnto model and in the LexInfo model, building on an appropriate OWL meta-model to talk about properties and concepts in the lexicon ontology.
- OWL 2: in OWL2, a technique known as *punning* allows us to use the same URI to refer to an individual and a class. This is possible by treating the URIs logically as different symbols, clearly distinguishing between the individual and the class (see [18]). OWL2 thus allows combined reasoning over lexica and domain ontologies, overcoming the above mentioned limitations of OWL-DL while clearly going beyond the expressive power of RDF(S).
- OWL Full: As OWL Full is the only 'dialect' of OWL which subsumes RDF(S), it allows the same URI to be used to represent an individual and a class and is the most expressive ontology language available. While it therefore provides the necessary expressiveness to represent lexica and their association to ontology elements, the disadvantage is that it is undecidable and thus not relevant for practical implementations.

5 Tools and Applications

In this section we describe the tool support that is so far available to create, maintain and serialize LexInfo lexicons. We discuss on the one hand the

[13] available here: `http://lexinfo.net/lexinfo`

design of the LexInfo API but also discuss a plug-in for the NeOn toolkit [14] that supports the semi-automatic development of LexInfo lexicons. Further, we also present our observations from initial experiments in which several people had the task of creating LexInfo lexica. Finally, we briefly discuss the application of LexInfo in ontology-based question answering systems as well as in search applications which operate across languages, discussing in particular the CLOVA architecture.

5.1 The LexInfo API

5.1.1 Interfaces and Aggregates

The LexInfo Application Programming Interface (API) has been designed to be closely aligned with the LexInfo ontology, such that there are Java classes — interfaces and implementations (with appropriate methods) — for all the classes defined in the ontology.

One important design choice of the LexInfo API is to have three different representation levels: i) interfaces, ii) implementations and iii) so called *aggregates*.

For example, in order to represent subcategorization frames for transitive verbs, there is one interface (Transitive.java) corresponding to the class `Transitive` in the LexInfo ontology, an implementation thereof (TransitiveImpl.java) as well as a so called *aggregate* (TransitiveAggregate.java). Aggregates are introduced to facilitate the usage, querying and manipulation of lexicon entries and provide all relevant information for each lexical entry in one spot, accessible through appropriate methods. In this sense aggregates encapsulate from the user all individuals necessary to represent a particular lexical entry and make the manipulation of lexical entries easier by hiding all the modeling details. The current version of the API includes the following aggregates:

- `NounAggregate`: representing a single noun mapping to a class or individual in the ontology
- `NounPhraseAggregate`: as above, but representing a complex noun phrase
- `ProperNounAggregate`, `ProperNounPhraseAggregate`: representing named entities which are interpreted as individuals or classes with respect to the ontology
- `VerbAggregate`, `VerbPhraseAggregate`: representing a verb or verb phrase
- `ClassAdjectiveAggregate`, `ScalarAdjectiveAggregate`, `LiteralAdjectiveAggregate`: representing an adjective of the corre-

[14] `http://neon-toolkit.org`

sponding type

- `NounPPAggregate`, `NounPhrase_NounPPAggregate`: representing a noun which subcategorizes a prepositional object and is thus interpreted as a property with respect to the ontology
- `TransitiveAggregate`, `VerbPhrase_TransitiveAggregate`, `IntransitivePPAggregate`, `VerbPhrase_IntransitivePPAggregate`: a verb subcategorizing a corresponding number and type of arguments

Each of these aggregates are implemented as a class in the LexInfo API with a number of methods that make it easier to access the individuals that constitute this element. For Example `NounPPAggregate` has methods such as `getNoun()`, `getPreposition`, `getNounPP()` (the subcategorization frame), `getSubject()` (the syntactic argument), `getRange()` (the semantic argument) etc. There is also a special aggregate called `LexiconAggregate` which includes all aggregates which are associated with a single `Lexicon` individual in the lexicon (and hence all aggregates of a specific language).

5.1.2 Default Generation of Lexicons using LILAC

In order to enable the quick adaptation of domain ontologies to LexInfo lexicons we provide a method which can be used to automatically generate a lexicon from an existing ontology. This works by using either the `rdfs:label` annotation, the fragment of the URI in the domain ontology or if that is not available the local name of the URI. We then apply a simple tokenization procedure (e.g., for `http://www.mygrid.org.uk/ontology#has_identifier` we extract "has identifier" and for `http://dbpedia.org/resource/Shibuya%2C_Tokyo` we extract "Shibuya, Tokyo"). Once we have the label in text form we apply a POS tagger to obtain part-of-speech information for each token.

We developed a rule language called LILAC (LexInfo Label Analysis and Construction) to process the labels. The LILAC rules essentially define a phrase structure grammar which is used to parse labels (after they have been tokenized). Phrase structures for labels are then used to construct corresponding aggregates. The following LILAC rules for example states that if a label consists of a noun followed by a preposition (e.g., `capitalOf`), it will give raise to a NounPP aggregate in the lexicon:

```
NounPP : <noun> <preposition>
```

When the above rule matches (e.g., on the label *capitalOf*), it creates a `NounPPAggregate` as well as the appropriate lexical entries and uses the `setNoun(...)` and `setPreposition(...)` methods of the aggregate. However, this is not sufficient as it does not capture the mapping between the syntactic and semantic arguments. The LILAC rules also support the explicit specification of this mapping:

The following rule matches on labels consisting of a verb. The rule creates a `TransitiveAggregate` and assumes that the subject of the verb is interpreted as the domain and the object as the range.

```
Transitive : <verb> { subject -> domain, object -> range }
```

The above rule would for example match the label *'produce'*.

The mapping can also be reversed, for example in the case that we have a label such as *'isProducedBy'*:

```
Transitive : "is" <verb> "by"  { subject -> range,
                                 object -> domain }
```

The above rule would also create a `TransitiveAggregate` with reversed mappings.

These annotations allow LILAC to map the syntactic and semantic arguments correctly. The second rule above can be refined by adding an annotation to match the word form:

```
Transitive : "is" <verb> [ verbform="participle" ] "by"
   { subject -> range, object -> domain }
```

The above cases have assumed that the values of the aggregates (the head etc.) are always filled by words contained in the label. However, LILAC is not restricted to this case, allowing to include further words when generating the aggregate. The following LILAC rule which matches a label such as *'hasAddress'* illustrates this:

```
NounPP : "has" <noun> { subject -> range,
   pObject -> domain, preposition -> of }
```

The above constructs an aggregate of type `NounPP` for the head noun <noun> subcategorizing a prepositional object introduced by the preposition "of". This will allow us to interpret constructions such as *"The address of X is Y"* in terms of the property in question.

We allow for the construction of phrase expressions by allowing rules to match recursively, for example "structural alignment report" can generate a noun phrase as in Fig. 12 with the following rules

```
NounPhrase: <noun> <NounPhrase>
NounPhrase: <noun> <noun>
```

Overall, LILAC rules build on the observation of several authors (see [23], [29] and [37]) that labels in ontologies typically show some regularities.

We created the LILAC rules using a small test ontology we developed from common ontology labels and then further refined it by developing extra rules based on the SWRC ontology [49]. We then tested the rules on two further ontologies: the MyGrid ontology [48] and the WINE ontology (a W3C example ontology). We then developed a set of approximately 100 rules to create aggregates corresponding to the different types supported by the LexInfo API. These rules can be applied to generate a default LexInfo model for any ontology. Table 2 shows results for these three ontologies. Two different taggers were used before applying the LILAC rules to the labels: the Stanford POS Tagger [50] and TreeTagger [47]. Table 2 indicates the number of aggregates per type created for each ontology and part-of-speech tagger. In the last row (Total) the table indicates the percentage of labels for which at least one aggregate was created. These first results are certainly encouraging and show that we can indeed create lexica automatically for any ontology by processing the labels according to the LILAC rules. While there could be some overfitting here to the three ontologies considered we are quite confident after inspection that the rules obtained bear a reasonable degree of generality. [15]

5.2 Manually Creating and Editing LexInfo Lexicons

While the creation of Lexinfo lexica can be automated as described in the previous section, we can not expect that automatically generated lexica are sufficient for real applications. First of all, the analysis and aggregates created will not always be correct. Second, so far, we do not get any alternatives for expressing classes, properties, etc. LILAC rules stay actually very close to the label and are thus not able to capture linguistic variants.

In order to allow people to create new lexica or interact with already existing lexica (either created automatically or by other people), we have implemented a plug-in for the Neon toolkit [16] which allows people to select an ontology element (individual, class or property), see the existing lexical entries for it or create a new lexical entry by selecting an appropriate type and introducing basic information. Figure 21 shows a screen-shot of the plug-in. We see that the user has selected the concept `mountain` from the Geobase ontology. The part of the plug-in labeled with "1" shows the empty list of lexical entries that have been already modeled and associated to the concept `mountain`. The action buttons in part "2" of the main interface allow the user to i) delete existing entries, ii) save the current entry, iii) create a new entry, iv) save the whole lexicon and v) open an existing lexicon. The third part of the main interface shows the current lexicon entry the user is creating / editing. It

[15] Software for generating lexica is available at `http://www.lexinfo.net/`
[16] `http://neon-toolkit.org/wiki/Main_Page`

	Stanford			TreeTagger		
	MyGrid	SWRC	Wine	MyGrid	SWRC	Wine
Noun	45	38	50	20	37	39
NounPhrase	386	31	173	345	27	138
ProperNoun	0	0	0	27	14	9
ProperNounPhrase	0	0	2	0	1	25
Verb	15	0	1	14	0	0
ClassAdjective	2	1	0	2	2	0
ScalarAdjective	0	2	0	0	3	0
Adverb	0	0	2	0	0	0
Transitive	3	7	1	3	5	0
VP Transitive	0	1	0	0	1	1
IntransitivePP	1	7	4	1	6	4
VP IntransitivePP	0	2	0	0	1	0
NounPP	4	60	5	4	58	5
NP NounPP	0	14	4	0	14	4
Not Recognized	26	3	11	53	10	28
Total	94.6%	98.8%	95.7%	89.0%	93.9%	88.9%

Table 2
Results of Lexicon Generation using LILAC

shows that the user is currently creating a scalar adjective *"high"* associated to the `height` property of the class `mountain`. He has further specified that a mountain with a height greater than 750m counts as "high" and that the adjective is positively correlated with the scale represented by `height`. The plug-in is available at `http://code.google.com/p/lexiconmapper/`.

We report on the results of a small experiment in which different test subjects created LexInfo lexicons for the Geobase ontology [17]. In our experiment, we let five test persons model a LexInfo lexicon for the Geobase ontology using the NeOn Toolkit plug-in. All of these test subjects were computer scientists. Only two of them had knowledge about ontologies and only one person had knowledge about natural language processing. The session with each of these subjects lasted for 40 minutes. In the first 10 minutes they received instructions on the LexInfo model and the different lexical entry types supported.

[17] Available at `http://www.cimiano.org/philipp/home/Data/Orakel/geobase.owl`

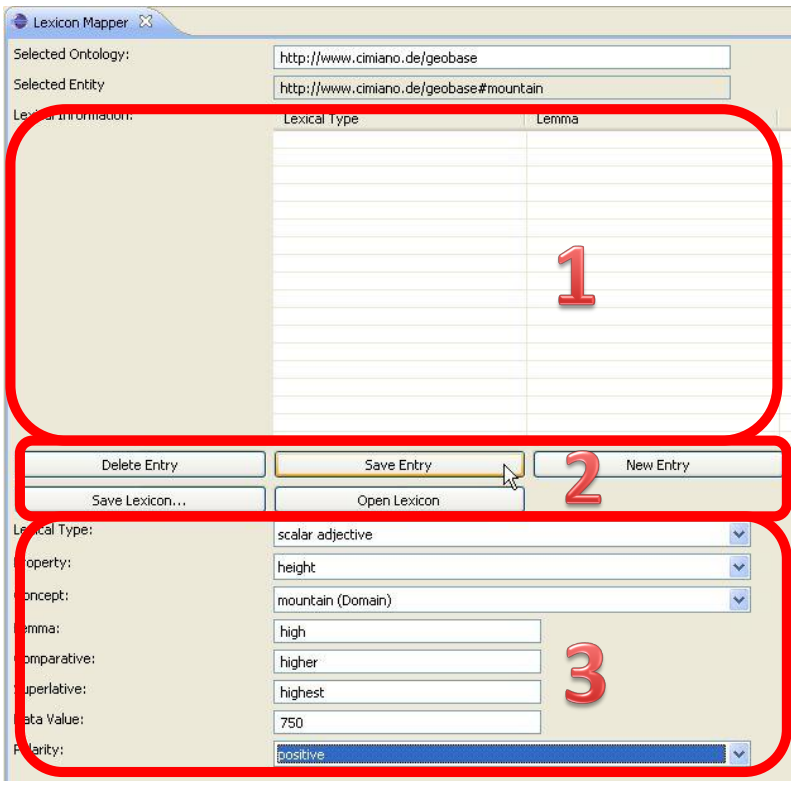

Fig. 21. Screen-shot of the NeOn toolkit plug-in

They received some written guidelines for this. They also received some basic instructions about how to use the NeOn Toolkit plug-in. In the remaining 30 minutes, the test subjects had to fulfill two tasks: creating their own lexicon from scratch as well as modifying an automatically created default lexicon.

None of the test subjects had any problems understanding the NeOn toolkit plug-in nor the LexInfo model and they created lexica containing between 21 and 32 lexical entries (manual mode) and 30 and 44 lexical entries (semi-automatic mode). Figure 22 shows the average number of lexical entries created for each type of lexical entry. We describe how such lexica can be used in question answering systems below.

5.3 Using LexInfo Lexicons

In the following we describe some applications were the LexInfo model has been deployed successfully. We briefly describe how LexInfo can be used in ontology-based question answering systems as well as in applications supporting cross-language access to semantic data.

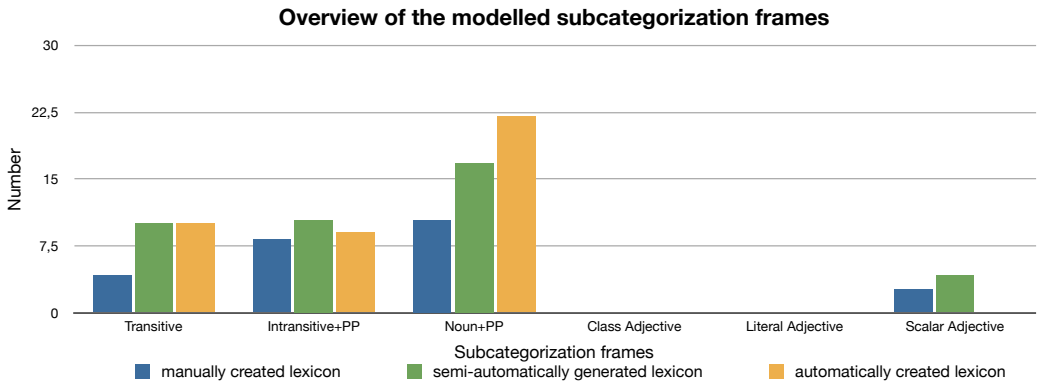

Fig. 22. Average number of modeled lexical entries

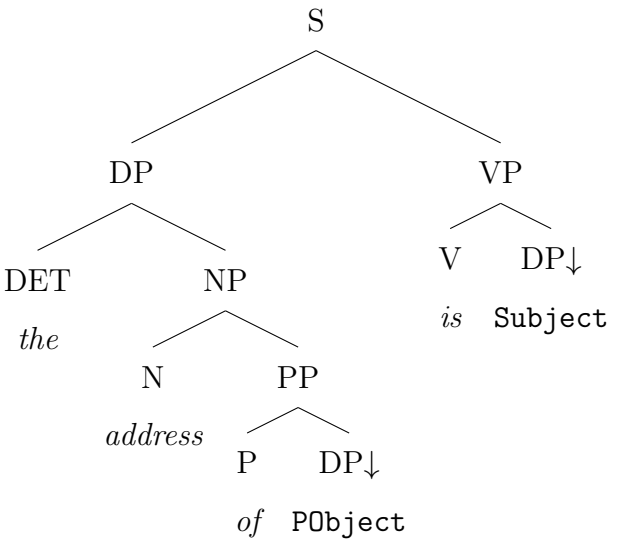

Fig. 23. LTAG tree for the *intransitive PP* aggregate "address of"

5.3.1 LexInfo for Question Answering

LexInfo can be used to partially generate lexicalized grammars – in particular Lexicalized Tree Adjoining Grammars (LTAG) – for question answering systems. We have implemented a generator that automatically generates LTAG elementary trees from the lexical entries in the LexInfo lexicon (cf. [**?**]). These elementary trees constitute a domain-specific lexicon, which can be used for parsing and semantic interpretation of questions with respect to an underlying ontology. For example, an elementary tree generated for the label `addressOf` is given in Fig. 23. In this tree, `PObject` and `Subject` refer to the semantic arguments corresponding to the subcategorization frame in the lexicon. In Fig. 24, we see that by using the *syn-sem argument maps* we are capable of linking this instance to a datatype property, and state that the `Subject` corresponds to the domain of the property and `PObject` to the range of the property.

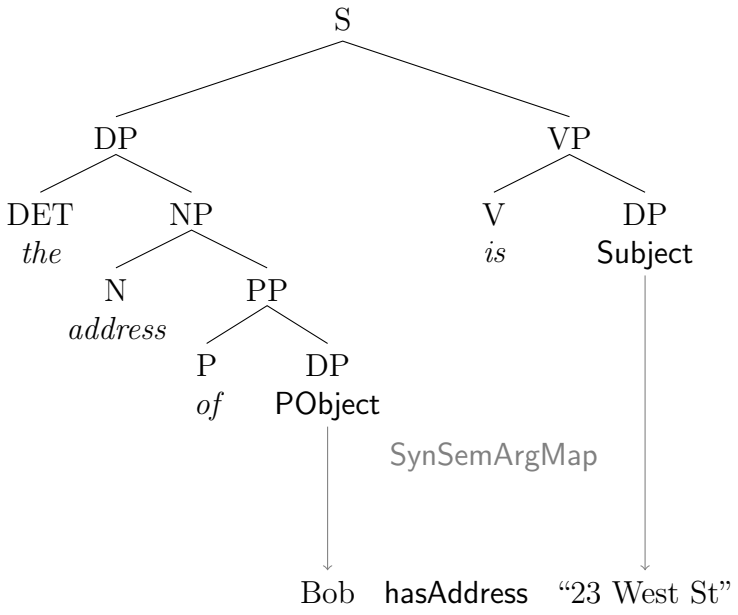

Fig. 24. Using LexInfo to select substitution into elementary trees

We applied this approach to an OWL version of Mooney's Geobase dataset. [18] The LexInfo model constructed for this ontology contains 762 lexical entries. 692 of them correspond to common nouns representing individuals, which are constructed automatically. The remaining 70 entries were built by hand, using the API that LexInfo provides. If we assume a reasonable effort of 2 minutes for building one such entry, the amount of work needed for lexicalizing the ontology amounts to a bit more than 2 hours. Then we used the generator to automatically generate 2785 grammar entries (LTAG trees together with a semantic representation) from those LexInfo entries. In addition, we manually specified 149 grammar entries for domain-independent elements such as determiners, wh-words, auxiliary verbs, and so on. Applied to a gold standard set of 859 questions and corresponding database queries, our system achieves a recall of 67% with 82% precision [**?**]. The main source of discrepancies between our system's output and the gold standard are due to the context-dependence of lexical elements and to a still required fine-tuning of the mechanism that translates semantic representations into queries.

5.3.2 LexInfo for Cross-Language Knowledge Retrieval

Data models and knowledge representation formalisms in the Semantic Web allow us to represent data independently of any natural language. In order to facilitate the interaction of human users with semantic data, supporting language-based interfaces seems crucial. As human users speak different languages, it is important to allow for the quick development of applications that

[18] `http://userweb.cs.utexas.edu/users/ml/geo.html`

allow to access semantic data in multiple languages. However, currently there is no principled approach supporting the access of semantic data across multiple languages. To fill this gap, we have designed an architecture called CLOVA (Cross-Lingual Ontology Visualization Architecture) designed for querying a semantic repository in multiple languages. A developer of a CLOVA application can define the search interface independently of any natural language by referring to ontological relations and classes within a *semantic form specification* (SFS) which is a declarative and conceptual representation of the search interface with respect to the ontology in a proprietary XML format. The search interface can then be automatically localized by the use of a lexicon ontology model such as LexInfo [12] which enables the system to automatically generate the form in the appropriate language. The queries to the semantic repository are generated on the basis of the information provided in the SFS and the results of the query can be localized using the same method as used for the localization of the search interface. The CLOVA framework is generic in the sense that it can be quickly customized to new scenarios, new ontologies and search forms. Additional languages can be added without changing the actual application, even at runtime if we desire.

CLOVA is a modular, reusable and extensible architecture implemented in Java which is fully configurable and easy to adapt to different data sources, user interfaces and localization tools. Figure 25 depicts the general architecture of CLOVA and its main modules. The *form displayer* is a module which translates the semantic form specification into a displayable format, e.g. HTML. Queries are executed by the *query manager* and the results are displayed to the user using the *output displayer* module. All of the modules use the *lexicalizer* module to convert the conceptual descriptions (i.e., URIs) to and from natural language. Each of these modules are implemented independently and can be exchanged or modified without affecting the other parts of the system. In what follows we briefly describe the main components of the system:

Semantic Form Specification: One of the most important aspects of CLOVA is the form specification, which consists of a list of fields which can be queried. Each field is associated to the URI of a property in the data source, an internal name, a usage property to determine if it can be included in the output and a so called *"property range"*. The property range defines semantically what values the property should take. Each property range is defined not by the data type of the range (e.g., integer) but by the set of queriable values it has. For example we may use an integer range to describe a property such as "year founded" but for a property for which it is not practical to be queried through a single value, e.g., "population", we specify the range as a *ranged integer* and for a property queried as a fixed set of brackets, e.g., "age" we specify it as a *segmented integer*. It is important to include these semantic distinctions in order to choose the right UI elements for the user to search the data source and to formulate the query correctly. CLOVA includes a num-

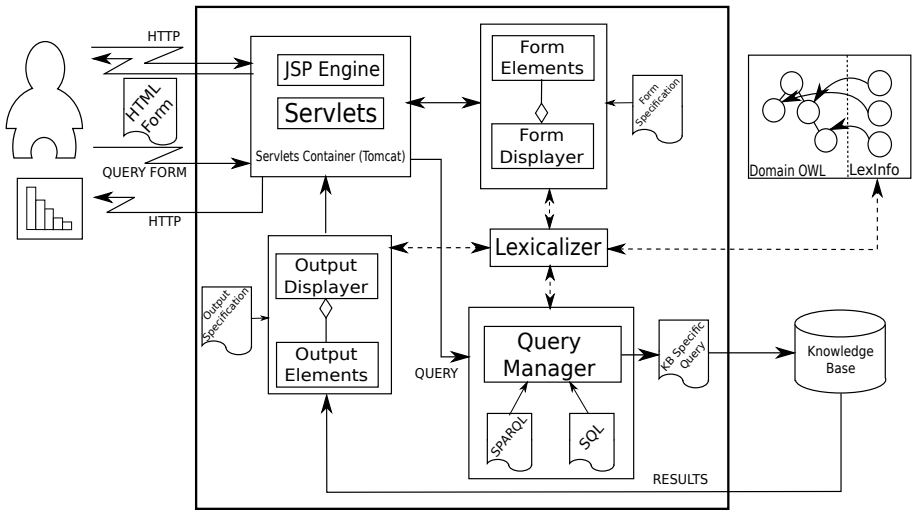

Fig. 25. CLOVA General Architecture

ber of property ranges and supports mechanisms for defining new property ranges and including them in the SFS. This document is in principle similar to the concept of a "lens" in the Fresnel display vocabulary [8], although our formalisms support additional information that is crucial when automatically generating a search interface given a conceptual specification of the relevant properties in the semantic form specification.

Query Manager: Once the form is presented to the user, he or she can fill the fields and select which properties he or she wishes to visualize in the results. When the query form is sent to the Query Manager, it is translated into a specific query for a particular knowledge base. We have provided modules to support the use of SQL queries using JDBC and SPARQL queries using Sesame [11]. We created an abstract query interface which can be used to specify the information required in a manner that is easy to convert to the appropriate query language allowing us to change the knowledge base, ontology and backend without major problems. The query also needs to be processed using the *lexicalizer*, due to the presence of language dependent terms introduced by the user which need to be converted to language-independent URIs.

Output Displayer: Once the query is evaluated, the results are processed by the *output displayer* to determine a proper visualization. The displayer has a display element for each different kind of output to represent, i.e., tables, bar charts, etc., which can be tested in order to see if it can display the data. The output displayer uses the lexicalizer to display the appropriate natural language labels in the same manner as the form displayer.

Lexicalizer: In order to lexicalize ontology URIs in different languages, CLOVA builds on the LexInfo Model and uses its API. In particular, CLOVA builds on the LILAC rule language in order to generate valid textual forms on the basis of the linguistic information contained in LexInfo. In fact, while

we have not mentioned this explicitly so far, LILAC can be also applied in reverse mode to generate lexicalizations for ontology elements (classes, properties, etc.) The lexicalization proceeds as follows: for each concept that needs to be lexicalized, the API is consulted to find the entry that has a sense with the same URI as the concept. Then appropriate LILAC rules to lexicalize the corresponding aggregate are selected. As this process requires only the URI of the concept, by changing the LexInfo model and providing a reusable set of LILAC rules, the language of the interface can be changed to any suitable form. This allows to move flexibly between languages as one has to add only a LexInfo lexicon for a given language without any further modification of the system.

Besides being used in the lexicalization of property URIs in different languages, LexInfo can also be used to resolve textual form queries into appropriate URIs that can be used to query the backend. For example if the user queried for *"food"* then the LexInfo model could be queried for all lexical entries that have either a lemma or word form matching this literal. The corresponding URIs referred to by this word can be used to query the knowledge base. This means that a user can query in their own language and expect the same results, for example the same concept for "food processing" will be returned by an English user querying "food" and a German user querying for *"Lebensmittel"* (part of the compound noun *"Lebensmittelverarbeitung"*).

We have realized the CLOVA architecture as a web application using HTML and JSP by implementing modules to translate the form specification into HTML. This involves creating a *form display element* for each property range containing the appropriate HTML code, and the *form displayer*, which organizes these into a single HTML form. Each of these elements depends on the *lexicalizer* which converts the ontology URIs to natural language. This form can then be incorporated into a JSP page and the layout and formatting of the pages can be easily changed by providing CSS stylesheets.

To demonstrate CLOVA we created a web portal for accessing information about companies. [19] We used a dataset that was a subset of the infobox data extracted from Wikipedia using the *Strict Infobox Ontology* for DBPedia 3.4 [1], reduced to the information relevant for companies. We have chosen this type of data because it is structured and measure units are normalized, which allows for more sophisticated querying. Also for most of the URIs there is a translation of the term provided in DBPedia, extracted from links between different language versions of Wikipedia, so we can quickly extract multilingual lexica. We used the LexInfo API to develop lexica for English, Spanish and German and created a form using several of the properties commonly used in the ontology.

[19] `http://www.sc.cit-ec.uni-bielefeld.de/clova/demo`

The bottom line is thus that LexInfo is providing a principled and declarative model that is used in the lexicalizer component of the CLOVA architecture to lexicalize ontology elements (properties in particular) in such a way that all the necessary information for accomplishing this is contained in the language-specific LexInfo models only. Thus, we can easily extend the languages covered by an application by adding an additional LexInfo ontology for the corresponding language without modification of the application code itself. This would even allow us to add new languages at runtime if necessary. LexInfo is thus contributing to defining a clear interface between an application supporting cross-language access to information and the linguistic information needed for this purpose. This leads to very modular architectures where lexica can be developed independently of an application and even shared without any implications for the application code using these models.

6 Conclusions and Future Work

The interface between language and knowledge as captured by ontologies is much richer and more complex than can be expressed by current ontology models, such as the labeling system of RDFS, OWL, and SKOS. Enhanced models that couple linguistic information with ontological structure are certainly required for tasks such as ontology learning and population from text, natural language generation from ontologies, etc. While there is no doubt that ad-hoc models for representing linguistic information might be suitable for individual systems and solutions to these problems, we have proposed a sound and principled model that is independent of any specific application (this is what we mean by 'declarative') and can be exchanged across systems. For this purpose we have clearly spelled out the requirements for such models and argued that many related proposals fall short of fulfilling these.

LexInfo fulfills the requirements stated and clearly details how the mapping from language to ontologies might work. In order to design LexInfo, we have built on two previously developed complementary models (LingInfo and LexOnto). In our current proposal we used the LMF meta-model to glue together the crucial ingredients of these models. LMF represents a solid and principled framework for representing computational lexica, of which we regard ontological lexicons as a special case. The LingInfo, LexOnto, LexInfo and LMF ontologies are available from the project website, [20] as well as a corresponding Java API with implementations for OWL API.

We have further shown how LexInfo can be applied in at least two applications with different needs. Concerning question answering applications, we

[20] `http://www.lexinfo.net`

have argued that LexInfo has the required expressivity to generate lexicalized grammars from it. LexInfo has also clear applications in cross-language semantic search applications. There LexInfo can be used to lexicalize both interfaces and query results to the language of choice of a user. The lexical information is captured in the LexInfo model only and can thus be exchanged in a modular fashion, allowing to include LexInfo lexica created by others and thus creating clear interfaces for the ontology-language interface. Finally, small experiments have been conducted to verify that users can indeed understand the model and are able to create LexInfo lexicons if appropriate GUIs are available.

In future work we intend to further develop the LexInfo API, extending it with more aggregate types as required by other applications. We are currently working on an extension of LexInfo to incorporate (generative) patterns that allow to generate morphological and inflectional variants such as the plural instead of storing them explicitly. Our mid-term goal is also to improve our default lexicon generation mechanisms by including information from linguistic resources (WordNet, CELEX, [21] COMLEX [22]) etc. as well as corpora such as Wikipedia that can give us information about frequency of usage of the aggregates in a lexicon. In this line we intend to develop a corpus-based approach which uses statistical measures defined on the corpus to assess the relevance and appropriateness of a certain entry for the domain in question. It is also our goal to extend our automatic lexicon creation methods to multiple languages. This will require the development of techniques for translating labels much as done in LabelTranslator [20]. It would also be interesting to investigate the relation between LexInfo and Linked Open Data. [23] LexInfo provides a formalism that can also be used to publish lexical information about data resources which could become itself part of the Web of data. People could thus create and publish their lexica much as they publish ontologies or data on the Web. This is a fascinating issue to explore which might allow us to add a "lexicon layer" crossing language boundaries to the Web of data, thus bringing it closer to the idea of a truly Multilingual Semantic Web. [24] Primarily however, we also hope to have provided a solid basis for any future discussion on standardization of lexicon models for OWL ontologies. At the time of preparing the final version of this manuscript, there is a new lexicon (meta-) model developed by some of the authors in the context of the Monnet project which brings together the LexInfo and the LIR models. In designing this new model – Lemon (Lexicon Model for Ontologies – we have attempted to provide a very minimal core vocabulary that can be extended with further modules as required by a specific application. In addition, lemon is linguistically agnos-

[21] `http://catalog.elra.info/product_info.php?products_id=439`
[22] `http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC98L21`
[23] `http://linkeddata.org/`
[24] `http://msw.deri.ie`

tic in the sense that it does not prescribe the usage of any specific linguistic categories. Instead, the model offers flexibility by enabling the definition of instantiations of lemon through the selection of specific data categories from a registry such as ISOcat[**?**].

Acknowledgements: This work is supported in part by the Science Foundation Ireland under Grant No. SFI/08/CE/I1380 (Lion-2) as well as by the European Union under Grant No. 248458 for the Monnet project. [25] and by the German Research Foundation (DFG) under the Multipla project (grant 38457858). We thank all those that have contributed in the last years to the development of LexInfo and its API, including (in alphabetical order): Hammad Afzal, Matthias Mantel, Thomas Wangler, and Tobias Wunner. Our thanks go also to Christina Unger for providing the examples related to grammar generation and to Peter Haase for his contributions to the design of LexInfo and its precursor LexOnto.

References

[1] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives. Dbpedia: A nucleus for a web of open data. *Lecture Notes in Computer Science*, 4825:722, 2007.

[2] C.F. Baker, C.J. Filmore, and J.B. Lowe. The berkeley framenet project. In *Proceedings of the International Conference on Computational Linguistics*, 1998.

[3] J. Bateman, R. Kasper, J. Moore, and R. Whitney. A general organization of knowledge for natural language processing: The penman upper model. Tech. Report. Information Sciences Institute, Marina del Rey, California, 1989.

[4] J.A. Bateman. Upper modeling: A general organization of knowledge for natural language processing. In *Proceedings of the Workshop on Standards for Knowledge Representation Systems*, 1990.

[5] J.A. Bateman. On the relationship between ontology construction and natural language: a socio-semiotic view. *International Journal of Human Computer Studies*, 43(5-6):929–944, 1995.

[6] S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D.L. McGuinees, P.F. Patel-Schneider, and L.A. Stein. OWL Web Ontology Language Reference, 2004.

[25] `http://www.monnet-project.eu/`

[7] A. Bernstein and E. Kaufmann. Gino - a guided input natural language ontology editor. In *Proceedings of the International Semantic Web Conference (ISWC)*, pages 144–157, 2006.

[8] C. Bizer, R. Lee, and E. Pietriga. Fresnel: A browser-independent presentation vocabulary for rdf. In *Proceedings of the Second International Workshop on Interaction Design and the Semantic Web, Galway, Ireland*. Citeseer, 2005.

[9] K. Bontcheva and B. Davis. *Semantic Knowledge Management: Integrating Ontology Management, Knowledge Discovery and Human Language Technology*, chapter Natural Language Generation from Ontologies, pages 117–130. Springer, 2009.

[10] D. Brickley and R.V. Guha. RDF Vocabulary Description Language 1.0: RDF Schema. Technical report, W3C, 2002.

[11] J. Broekstra, A. Kampman, and F. Van Harmelen. Sesame: A generic architecture for storing and querying rdf and rdf schema. *Lecture Notes in Computer Science*, pages 54–68, 2002.

[12] P. Buitelaar, P. Cimiano, P. Haase, and M. Sintek. Towards linguistically grounded ontologies. In *Proceedings of the European Semantic Web Conference (ESWC)*, pages 111–125, 2009.

[13] P. Buitelaar, T. Declerck, A. Frank, S. Racioppa, M. Kiesel, M. Sintek, R. Engel, M. Romanelli, D. Sonntag, B. Loos, V. Micelli, R. Porzel, and P. Cimiano. Linginfo: Design and applications of a model for the integration of linguistic information in ontologies. In *Proceedings of OntoLex06, a Workshop at LREC*, 2006.

[14] P. Buitelaar, M. Sintek, and M. Kiesel. A lexicon model for multilingual/multimedia ontologies. In *Proceedings of the 3rd European Semantic Web Conference (ESWC06)*, 2006.

[15] P. Cimiano, P. Haase, J. Heizmann, M. Mantel, and R. Studer. Towards portable natural language interfaces to knowledge bases - the case of the orakel system. *Data Knowledge Engineering*, 65(2):325–354, 2008.

[16] P. Cimiano, P. Haase, M. Herold, M. Mantel, and P. Buitelaar. Lexonto: A model for ontology lexicons for ontology-based NLP. In *Proc. of the OntoLex (From Text to Knowledge: The Lexicon/Ontology Interface) workshop at ISWC07 (International Semantic Web Conference)*, 2007.

[17] P. Cimiano and M. Minock. Natural language interfaces: What is the problem? - a data-driven quantitative analysis. In *Proceedings of the 14th International Conference on Applications of Natural Language to Information Systems*, 2009.

[18] B. Cuenca-Grau, I. Horrocks, B. Mortik, B. Parsia, P. Patel-Schneider, and U. Sattler. Owl2: The next step for owl. *Journal of Web Semantics: Science, Services and Agents on the World Wide Web*, 2008. in press.

[19] B. Davis, A. Ali Iqbak, A. Funk, V. Tablan, K. Bontcheva, H. Cunningham, and S. Handschuh. Rountrip ontology authoring. In *Proceedings of the International Semantic Web Conference*, pages 50–65, 2008.

[20] M. Espinoza, A. Gomez-Perez, and E. Montiel-Ponsoda. Multilingual and localization support for ontologies. In *Proceedings of the European Semantic Web Conference (ESWC)*, pages 821 – 825, 2009.

[21] C. Fellbaum. *WordNet, an electronic lexical database*. MIT Press, 1998.

[22] C. Fillmore. *Linguistics in the Morning Calm*, chapter Frame semantics, pages 111–137. Hanshin Publishing, 1982.

[23] G. Fliedl, C. Kop, and J. Vöhringer. From OWL class and property labels to human understandable natural language. In *Proceedings of the 12th International Conference on Applications of Natural Language to Information Systems (NLDB)*, volume 4592 of *Lecture Notes in Computer Science*, pages 156–167. Springer, 2007.

[24] G. Francopoulo, N. Bel, M. Georg, N. Calzolari, M. Monachini, M. Pet, and C. Soria. Lexical markup framework: ISO standard for semantic information in NLP lexicons. In *Proceedings of the Workshop of the GLDV Working Group on Lexicography at the Biennial Spring Conference of the GLDV*, 2007.

[25] A. Gangemi, N. Guarino, C. Masolo, and A. Oltramari. Sweetening WordNet with DOLCE. *AI Magazine*, 24(3):13–24, 2003.

[26] A. Gangemi, R. Navigli, and P. Velardi. The ontowordnet project: Extension and axiomatisation of conceptual relations in wordnet. In *Proceedings of the International Conference on Ontologies, Databases and Applications of SEmantics (ODBASE 2003)*, 2003.

[27] The LMF Working Group. Language resource management – lexical markup framework (LMF). Technical Report ISO/TC 37/SC 4 N453 (N330 Rev.16), ISO, 2008.

[28] S. Hartrumpf, H. Helbig, and R. Osswald. The semantically based computer lexicon HaGenLex – structure and technological environment. *Traitement automatique des langues*, 44(2):81–105, 2003.

[29] D. Hewlett, A. Kalyanpur, V. Kovlovski, and C. Halaschek-Wiener. Effective natural language paraphrasing of ontologies on the semantic web. In *Proceedings of the Workshop on End User Semantic Web Interaction, collocated with the International Semantic Web Conference (ISWC)*, 2005.

[30] G. Hirst. *Handbook on Ontologies*, chapter Ontology and the Lexicon, pages 209–229. Springer, 2004.

[31] G. Hirst. Ontologies and the lexicon. In Steffen Staab and Rudi Studer, editors, *Handbook on Ontologies*, International Handbooks on Information Systems, pages 209–229. Springer, 2004.

[32] O. Lassila and R. Swick. Resource description framework (RDF) model and syntax specification. Technical report, W3C, 1999.

[33] A. Lenci, N. Bell, F. Busa, N. Calzolari, E. Gola, M. Monachini, A. Ogonowsky, I. Peters, W. Peters, N. Ruimy, M. Villegas, and A. Zampolli. SIMPLE: A general framework for the development of multilignual lexicons. *International Journal of Lexicography*, 13(4):249–263, 2000.

[34] V. Lopez and E. Motta. Ontology-driven question answering in aqualog. In *Proceedings of the International Conference on Applications of Natural Language to Information Systems (NLDB)*, pages 89–102, 2004.

[35] C. Masolo, S. Borgo, A. Gangemi, N. Guarino, and A. Oltramari. Ontology library (final). WonderWeb deliverable D18, 2003.

[36] D.L. McGuiness and F. van Harmelen. OWL Web Ontology Language Overview. W3C Recommendation, February 2004.

[37] C. Mellish and X. Sun. The semantic web as a linguistic resource: Opportunities for natural language generation. *Knowl.-Based Systems*, 19(5):298–303, 2006.

[38] A. Miles and S. Bechofer. Skos simple knowledge organization system reference. W3c working draft, World Wide Web Consortium, 2008.

[39] A. Miles, B. Matthews, D. Beckett, D. Brickley, M. Wilson, and N. Rogers. Skos: A language to describe simple knowledge structures for the web. In *Proceedings of the XTech Conference*, 2005.

[40] E. Montiel-Ponsoda, W. Peters, G. Auguado de Cea, M. Espinoza, A. Gómez Pérez, and M. Sini. Multilingual and localization support for ontologies. Technical report, D2.4.2 Neon Project Deliverable, 2008.

[41] S. Nirenburg and V. Raskin. *Ontological Semantics*. MIT Press, 2004.

[42] A. Oltramari and A. Stellato. Enriching ontologies with linguistic content: An evaluation framework. In *Proceedings of OntoLex 2008 (Hosted by Sixth international conference on Language Resources and Evaluation*, 2008.

[43] W. Peters, E. Montiel-Ponsoda, G. Aguado de Cea, and A. Gómez-Pérez. Localizing ontologies in OWL. In *Proceedings of the ISWC OntoLex'07 Workshop*, 2007.

[44] C. Pollard and I.A. Sag. *Head-Driven Phrase Structure Grammar*. University of Chicago Press, 1994.

[45] J. Scheffczyk, C.F. Baker, and Srini Narayanan. Ontology-based reasoning about lexical resources. In *Ontology and Lexical Resources in Natural Language Processing*, 2006. to appear, earlier version presented at OntoLex 2006.

[46] J. Scheffczyk, A. Pease, and M. Ellsworth. Linking framenet to the suggested upper merged ontology. In *Proceedings of the International Conference on Formal Ontology in Information Systems*, 2006.

[47] H. Schmid. Improvements in part-of-speech tagging with an application to German. *Natural Language Processing Using Very Large Corpora*, 11:13–26, 1999.

[48] R.D. Stevens, A.J. Robinson, and C.A. Goble. myGrid: personalised bioinformatics on the information grid. *Bioinformatics-Oxford*, 19(1):302–304, 2003.

[49] Y. Sure, S. Bloehdorn, P. Haase, J. Hartmann, and D. Oberle. The SWRC ontology-semantic web for research communities. *Lecture notes in computer science*, 3808:218, 2005.

[50] K. Toutanova and C.D. Manning. Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In *EMNLP/VLC 2000*, pages 63–70, 2000.

[51] D. Vrandecic, J. Völker, P. Haase, T. Duc, and P. Cimiano. A metamodel for annotations of ontology elements in OWL-DL. In *Proceedings of the 2nd Workshop on Ontologies and Meta-Modeling*. GI Gesellschaft für Informatik, 2006.

[52] Y. Wilks. The Semantic Web: Apotheosis of annotation, but what are its semantics? *IEEE Intelligent Systems*, 23(3):41–49, 2008.