Senses in the lexico-ontology model

Philipp Cimiano

October 31, 2012

Abstract

Input document for the discussion on senses in the ontolex telco on Nov. 2, 2012.

1 Preliminaries

This document makes a proposal on how to model senses in the lexicon-ontology model. This proposal is based on the discussions carried out via the ontolex mailing list. First of all, we try to give a definition of a *sense*.

Def: A sense represents the linguistic meaning of a word w when understood as referring to a certain concept c.

This does not specify *how* senses will be represented in the lexicon-ontology interface.

There are essentially two 'views' on senses:

- They are an 'object' (first-order constant) that reifies the pair of w and c, allowing to attach additional information, e.g. pragmatic information related to the usage of word w as referring to concept c.
- The sense can be understood as a subclass of c, capturing aspects of the concept c when verbalized through w. A sense is thus also a first-order property.

Both views are legitimate and might both be relevant for end-user applications involving the lexicon-ontology model.

2 Proposal

The concrete proposal would be to account for this dual nature of a sense and support two ways of conceptualizing and axiomatizing senses, i.e. senses as objects and senses as subclasses. I elaborate on these two roles below.

3 Senses as objects

According to this view, a sense would be a first-order constant, i.e. in OWL:

```
s rdf:type lexonto:Sense
```

It would reify the connection between a word w and the concept c. The proposal is to represent this through two OWL object properties *hasSense* and *representedBy*, with the following axiomatization:

$$\top \sqsubseteq \forall hasSense.Sense$$

$$\top \sqsubseteq \forall hasSense^{-}.Lex(icalEntry)$$

$$\top \sqsubseteq \forall representedBy^{-}.Sense$$

Further, hasSense and representedBy are inverse functional and functional respectively:

$$\top \sqsubseteq = 1 hasSense^{-}.Lex$$

$$\top \sqsubseteq = 1 representedBy.\top$$

Take the example of the word *lemon* and its two senses as *'A yellowish citrus fruit'* and *'a defective or inadequate item'*.

We would formalize this in our model as follows:

```
ex:lemon rdf:type ontolex:Lex.
ex:lemon ontolex:hasSense lemon_1.
lemon_1 ontolex:representedBy <http://dbpedia.org/page/Lemon>.

ex:lemon ontolex:hasSense lemon_2.
lemon_2 ontolex:representedBy ex:DefectiveItem.
```

4 Senses as subclasses

Following the example above, we could then also say the following:

```
ex:lemon rdf:type ontolex:Lex.
ex:lemon ontolex:hasSense lemon_1.
lemon_1 owl:subClassOf <http://dbpedia.org/page/Lemon>.

ex:lemon ontolex:hasSense lemon_2.
lemon_2 owl:subClassOf ex:DefectiveItem.
```

5 Shortcuts

Introducing shortcuts would indeed make the model more accessible for people that want to use a simplified version of the lexicon-ontology model. We could indeed introduce two properties lex and ref with the following axiomatization:

$$\top \sqsubseteq \forall lex.Lex$$

$$\top \sqsubseteq \forall ref^{-}.Lex$$

$$hasSense \circ representedBy \sqsubseteq ref$$

$$representedBy^{-} \circ hasSense^{-} \sqsubseteq lex$$

$$ref^{-} \equiv lex$$

And further (not expressible in OWL2 DL):

$$\forall x, y\ lex(x, y) \rightarrow \exists s\ Sense(s) \wedge hasSense(x, s) \wedge representedBy(s, y)$$

6 Discussion

With the above proposal, we can indeed capture the dual role of a sense as a reification of word and concepts as well as as subclass of some ontological concept, thus supporting:

- A dual view on senses, supporting also ontology reasoning over ontology and lexicon if needed.
- One model for different purposes and applications (corresponding to profiles in OWL2)
- LOD compliance in the sense of allowing to retrieve all lexicalizations of a given class in one step without requiring subsumption reasoning

A big question is certainly how to relate both views. I have no answer to this so far.

Concerning the shortcut: it is true that it blows up the complexity of the model. However, it simplifies the usage of the model and thus potentially increases its uptake. Essentially, in a simple usage mode, one would just use one class Lex, as well as two properties lex and ref. This is a very simple pattern.

The question is how to deal with the above axioms relating the longer chain to the shortcut in practice.

I think this is something that we will have to leave to the different implementations of the model. But we could give some support for this in the reference implementation of the ontolex API. When a user asks for *lex*s for a given class *c*, the API could make a query to the repository as follows:

```
SELECT ?lex
WHERE
{
  {c ontolex:lex ?lex.}
  UNION
  {?lex ontolex:hasSense ?sense. ?sense representedBy c}
}
```

And similar for the case where the user wants to retrieve the *ref*s for a lexical entry.

I think this can be handled effectively by query expansion (as above) in any implementation of an API for the lexicon-ontology model. Other implementations might want to rely on OWL2 DL reasoning instead of on query expansion. That's up to the implementation of the lexicon-ontology model.

The bigger problem I see is with queries for 'senses' of a word. In some cases we would need to infer a non-existing sense object. This can be done through SPARQL-construct, i.e materializing the senses out. However, this is problematic as this might lead to a proliferation of senses representing the same pair of word and concept.

For example, if I state:

```
ex:lemon rdf:type ontolex:Lex.
ex:lemon ontolex:hasSense lemon_1.
lemon_1 ontolex:representedBy <http://dbpedia.org/page/Lemon>.

ex:lemon ontolex:ref <http://dbpedia.org/page/Lemon>.
```

Then the question certainly is how many senses I get back with the query. Ideally, I would like to get one sense back.

Overall I propose not to worry too much about implementation aspects at this stage. There are different ways of implementing the right behaviour. Some are more cumbersome than others. We should talk about whether we think that the simplification of the model is worthwile having or not.

One question: do we want to model that for any pair of class and lex, there is at most one sense relating them? Can we do this in OWL?