Literal Reification

http://ontologydesignpatterns.org/wiki/Submissions:Literal_Reification

Aldo Gangemi
ISTC-CNR
aldo.gangemi@cnr.it

Silvio Peroni
University of Bologna
speroni@cs.unibo.it

Fabio Vitali
University of Bologna
fabio@cs.unibo.it

ABSTRACT

In this paper we introduce the pattern *literal reification*, a modelling technique to address scenarios, in which we need to bless particular literals, usually when applying data properties, in order to use them as subjects and/or full-fledged objects of semantic assertions.

Keywords

OWL, SWRL, literal reification

1. INTRODUCTION

Recently within the Semantic Web community a new topic has been actively discussed: whether and how to allow literals to be subjects of RDF statements[1]. This discussions failed to provide a unique and clear indication of how to proceed in that regard.

Although one of the suggestions coming out of the discussion was to explicitly include the proposal in a (future) specification of RDF, this topic is not actually new, particularly in ontology modelling. The needs to describe "typical" literals (specially strings) as individuals of a particular class has been addressed by a lot of models in past, such as Common Tag[2] (through the class *Tag*), SIOC[3] (through the classes *Category* and *Tag*), SKOS-XL[4] (through the class *Label*), and LMM[5] (through the class *Expression*). After considering the above-mentioned models and other related and inspiring ones, we have developed a pattern called *literal reification* to address this issue. It allows to express literal values as proper ontological individuals so as to use them as subject/object of any assertion within OWL models.

The rest of the paper follows this structure: in Section 2 and Section 3 we respectively introduce a high level and detailed description of the pattern; in Section 4 we discuss two particular application scenarios that we use to demonstrate all the capabilities of the pattern.

2. GENERAL DESCRIPTION

Extending the pattern *region*[6], the pattern *literal reification* promotes any literal as "first class object" in OWL by reifying it as a proper individual of the class *litre:Literal*. Individuals of this class express literal values through the functional data property *litre:hasLiteralValue* and can be connected to other individuals that share the same literal value by using the property *litre:hasSameLiteralValueAs*. Moreover, a literal may refer to, and may be referred by any OWL individual through *litre:isLiteralOf* and *litre:hasLiteral* respectively.

Note that the pattern defines also a SWRL rule that allows to infer the (not explicitly asserted) literal value of a particular literal individual when it is connected to another literal individual via *litre:hasSameLiteralValueAs*:

```
litre:hasSameLiteralValueAs(x,y) ,
litre:hasLiteralValue(y,v)
  -> litre:hasLiteralValue(x,v)
```

This pattern allows to use each reified literal as subject or object of any assertion, and it is able to address scenarios described, for example, by the following competency questions:

- What is the context in which entities refer to a particular literal value?
- What is the meaning of a particular value considering the context in which it is used?

Plausible scenarios of its application include:

- modelling domains concerning descriptive tags, in which each tag may have more than one meaning depending on the context in which it is used;
- extending quickly the capabilities of a model by adding the possibility to make assertions on values, previously referred through data properties, without modifying it.

3. ELEMENTS

As shown in Fig. 1, the pattern *literal reification* is composed by a class, a data property and three object properties, described as follows:

- Class *litre:Literal*. It describes reified literals, where the literal value they represent is speficied through the property *litre:hasLiteralValue*. Each individual of this class must always have a specified value.
- Data property *litre:hasLiteralValue*. It is used to specify the literal value that an individual of *litre:Literal* represents.

[1] http://www.w3.org/2001/sw/wiki/Literals_as_Subjects.
[2] http://www.commontag.org
[3] http://rdfs.org/sioc/spec
[4] http://www.w3.org/TR/skos-reference/#xl
[5] http://www.ontologydesignpatterns.org/ont/lmm/LMM_L1.owl
[6] http://ontologydesignpatterns.org/wiki/Submissions:Region

- Object property *litre:hasSameLiteralValueAs*. It relates the reified literal to another one that has the same literal value.
- Object property *litre:hasLiteral*. It connects individuals of any class to a reified literal.
- Object property *litre:isLiteralOf*. It connects the reified literal to the individuals that are using it.

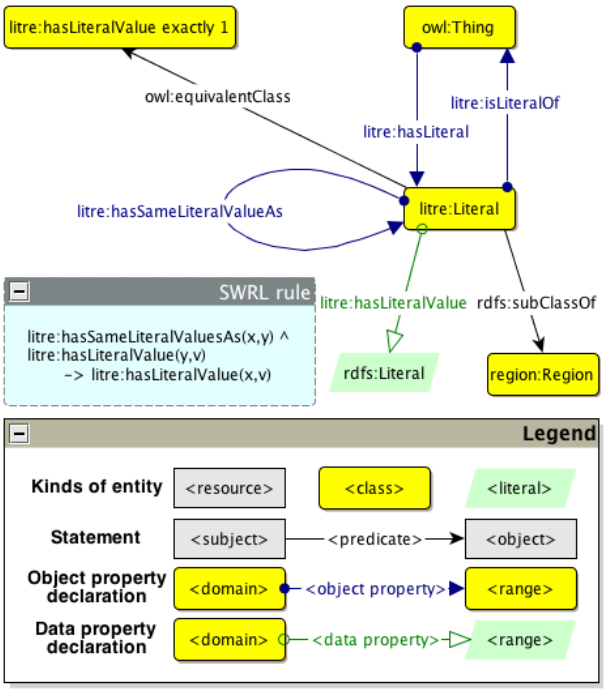

Figure 1: A figure summarizing the pattern.

4. SCENARIOS

4.1 Same tag, different meanings

Used frequently in the Web 2.0, descriptive tags such as the ones used in folksonomies are keywords (e.g., strings) assigned to a particular resource, such as a web document, with the intent to describe it. Just like words in any natural language, tags may have different meanings depending on the context in which they are used.

For instance, the word "Paris" may be either a name of a city or a first name of a person. Here, it is clear that the act of tagging with "Paris" both the Wikipedia pages about the Eiffel Tower and the one about Paris Hilton hides two different intents: in the former case, "Paris" denotes the city in which the tower stands; in the latter case, "Paris" denotes a particular person, i.e., Paris Hilton.

Using the literal reification pattern it is possible to express descriptive tags as first class objects in OWL, by considering them as proper individuals of the class *litre:Literal*. Different individuals may thus represent different meanings even if their literal values are identical[7]:

```
<http://en.wikipedia.org/wiki/Eiffel_Tower>
  a foaf:Document
  ; prism:keyword :parisTag1 .

<http://en.wikipedia.org/wiki/Paris_Hilton>
  a foaf:Document
  ; prism:keyword :parisTag2 .

:parisTag1 a litre:Literal
    , [ a skos:Concept
      ; skos:definition "A name associated
        to a city"@en ]
  ; litre:hasLiteralValue "Paris"
  ; lmm:denotes dbpedia:Paris .

:parisTag2 a litre:Literal
    , [ a skos:Concept
      ; skos:definition "A first name of
        a person"@en ]
  ; litre:hasSameLiteralValueOf :parisTag1
  ; lmm:denotes dbpedia:Paris_Hilton .
```

4.2 Keeping track of name changes

NameHistory3.0 is a (fictional) institution that keeps track of all the names of people, and stores them as an ABox of the FOAF ontology. In particular, each person is stored as an individual of the class *foaf:Person* with a specific first name (data property *foaf:givenName*) and family name (data property *foaf:familyName*).

On 24/09/2010, Bruce Wayne formally applied for changing his first name to Jack. Since NameHistory3.0 has to keep track of everything concerning names of people, on that date "Jack" was added as Mr. Wayne's first name. It was then that NameHistory3.0 noticed that, without any additional information, it is not possible to know which of the two first names are legally valid at any given point in time.

A solution to that scenario, which avoids any modification of the ontology model and consequently of the entire triple store (operation that is obviously time-consuming and error-prone), is to use the literal reification pattern in combination with the new expressivity for *punning* in OWL 2. Through them, it is possible to define a literal individual as also belonging to the class *foaf:givenName* – that is actually defined as a data property, but may be additionally be metamodelled as a class. We can now associate a particular time interval to each literal, so as to represent when the literal itself, i.e., the given name, is legally valid[8]:

```
:mr_wayne a foaf:Person
  ; foaf:familyName "Wayne"
  ; litre:hasLiteral
    [ a litre:Literal , foaf:givenName
      ; litre:hasLiteralValue "Bruce"
      ; dcterms:valid
        [ a ti:TimeInterval
          ; ti:hasIntervalStartDate
            "1983-01-15"
          ; ti:hasIntervalEndDate
            "2010-09-24"] ]
  ; litre:hasLiteral
    [ a litre:Literal , foaf:givenName
      ; litre:hasLiteralValue "Jack"
      ; dcterms:valid
        [ a ti:TimeInterval
          ; ti:hasIntervalStartDate
            "2010-09-24" ] ] .
```

[7] http://www.essepuntato.it/2010/06/sc1.ttl

[8] http://www.essepuntato.it/2010/06/sc2.ttl