RDF Instantiation of ISLE/MILE Lexical Entries

Nancy Ide
Department of Computer Science
Vassar College
Poughkeepsie, New York
USA 12604-0520
ide@cs.vassar.edu

Alessandro Lenci
Università di Pisa
Dipartimento di Linguistica
Via Santa Maria 36
56100 PISA
Italy
lenci@ilc.cnr.it

Nicoletta Calzolari
Istituto di Linguistica
Computazionale, CNR
Area della Ricerca
Via Moruzzi 1 – 56100 PISA
Italy
glottolo@ilc.cnr.it

Abstract

In this paper we describe the overall model for MILE lexical entries and provide an instantiation of the model in RDF/OWL. This work has been done with an eye toward the goal of creating a web-based registry of lexical data categories and enabling the description of lexical information by establishing relations among them, and/or using pre-defined objects that may reside at various locations on the web. It is also assumed that using OWL specifications to enhance specifications of the ontology of lexical objects will eventually enable the exploitation of inferencing engines to retrieve and possibly create lexical information on the fly, as suited to particular contexts. As such, the model and RDF instantiation provided here are in line with the goals of ISO TC37 SC4, and should be fully mappable to the proposed pivot.

1 Introduction

The eventual vision for computational lexicons is to enable universal access to sophisticated linguistic information, which in turn will serve as a central component for content-based information management on the web. This demands, first of all, some standardized means to represent complex lexical information while retaining the flexibility required to accommodate diverse approaches to lexicon organization and use. To this end, the ISLE[1] (*International Standards for Language Engineering*) Computational Lexicons Working Group (CLWG) has designed MILE (*Multilingual ISLE Lexical Entry*), a general schema for the encoding of multilingual lexical information intended as a meta-entry that can serve as a standardized representational layer for multilingual lexical resources. MILE consists of an incremental definition of object-oriented layers for lexical description that will enable open and distributed lexicons, with elements possibly residing in different sites of the web. The defined lexical objects are intended for use by lexicon and application developers to build and target lexical data at high level of abstraction.

The Resource Definition Framework (RDF) and the Ontology Web Language (OWL) recently developed by the World Wide Web Consortium (W3C) build upon the XML web infrastructure to enable the creation of a *Semantic Web*, wherein web objects can be classified according to their properties, and the semantics of their relations (links) to other web objects can be precisely defined. This in turn will enable powerful inferencing capabilities that can adapt language processing applications to particular contexts.

The MILE lexical entry is an ideal structure for rendering via RDF/OWL. It consists of a hierarchy of lexical objects that are built up in a layered fashion by combining atomic data categories via clearly defined relations. The overall architecture is modular and layered, as described in Atkins et al.

[1] ISLE Web Site URL: lingue.ilc.pi.cnr.it/EAGLES96/isle/ISLE_Home_Page.htm

(2002) and Calzolari *et al.* (2003). On the horizontal dimension, independent, linked modules target different dimensions of lexical entries. On the vertical dimension, the layered organization allows for varying degrees of granularity in lexical descriptions, allowing both "shallow" and "deep" lexical representations. RDF's class hierarchy mechanism, together with its capacity to specify named relations among objects in the various classes, provide a web-based means to represent this architecture.[2] Furthermore, because RDF allows for instantiating objects in any defined class and subsequently referring to them as the target of appropriate relations, lexical objects at any level of specificity can be pre-defined. This provides an important mechanism for standardization of lexical elements, since these elements may be pre-defined, organized in class hierarchies with inherited properties, and used "off-the-shelf" as needed.

In this paper we describe the overall model for MILE lexical entries and provide an instantiation of the model in RDF/OWL. This work has been done with an eye toward the goal of creating a web-based registry of lexical data categories and enabling the description of lexical information by establishing relations among them, and/or using pre-defined objects that may reside at various locations on the web. It is also assumed that using OWL specifications to enhance specifications of the ontology of lexical objects will eventually enable the exploitation of inferencing engines to retrieve and possibly create lexical information on the fly, as suited to particular contexts. As such, the model and RDF instantiation provided here are in line with the goals of ISO TC37 SC4, and should be fully mappable to the proposed pivot.[3]

2 The MILE Lexical Model

The MILE Lexical Model (MLM) consists of two primary components: a mono-lingual component and a multi-lingual component. The mono-lingual component comprises three layers: morphological, syntactic, and semantic. The overall architecture is shown in Figure 1.

Within each of the MLM layers, two types of objects are defined:

1. *MILE Lexical Classes* (MLC): the main building blocks of lexical entries. They formalize the basic lexical notions for each layer defined in the ISLE project (Calzolari *et al.* 2003). The MLM defines each class by specifying its attributes and the relations among them. Classes represent notions like *syntactic feature*, *syntactic phrase*, *predicate*, *semantic relation*, *synset*, etc. Instances of MLCs are the *MILE Data Categories* (MDC). So for instance, NP and VP are data category instances of the class <Phrase>, and SUBJ and OBJ are data category instances of the class <Function>. Each MDC is identified by a URI. MDC can be either user- defined or reside in a shared repository.
2. *lexical operations*: special lexical entities which allow users to state conditions and perform complex operations over lexical entries. They will for instance allow lexicographers to establish multilingual conditions, link the slots within two different syntactic frames, link semantic arguments with syntactic slots, etc.

The MLM is described with Entity-Relationship (E-R) diagrams defining the entities of the lexical model and the way they can be combined to design an actual lexical entry. As such, the MLM does not correspond to a specific lexical entry, but is rather an *entry schema* corresponding to a *lexical meta-entry*. This means that different possible lexical entries can be designed as instances of the schema provided by the MLM. Instance entries might therefore differ for the *type* of information they include (e.g. morphological, syntactic, semantic, monolingual or multilingual, etc.), and for the depth of lexical description.

Figure 2 depicts the MLM classes and relations for the syntactic layer (SynU for "syntactic unit"). Full definitions for the MLM can be found in the ISLE document (Calzolari *et al.* 2003).

[2] It should be noted that this architecture is analogous to other data models, including ER diagrams and various knowledge representation schemes.

[3] We have in fact produced a version of the prototype ISLE lexical entry in an XML format instantiating the proposed ISO pivot format (Ide and Romary, Vassar/LORIA internal document).

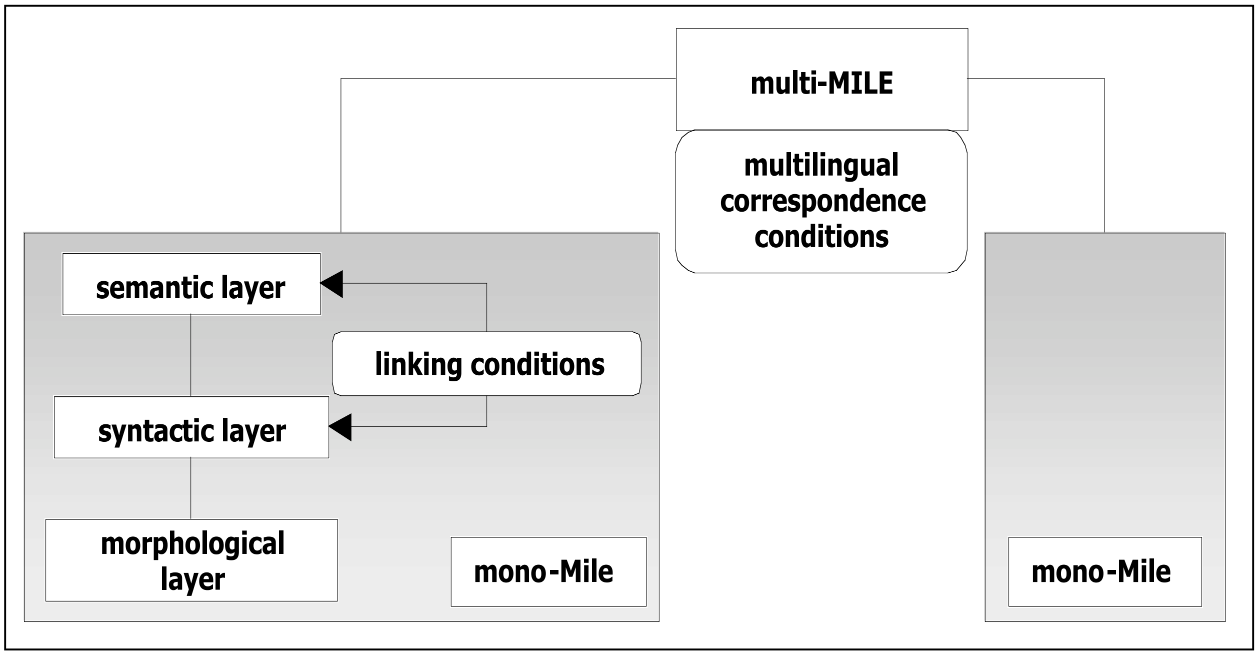

Figure 1. Overall MILE architecture

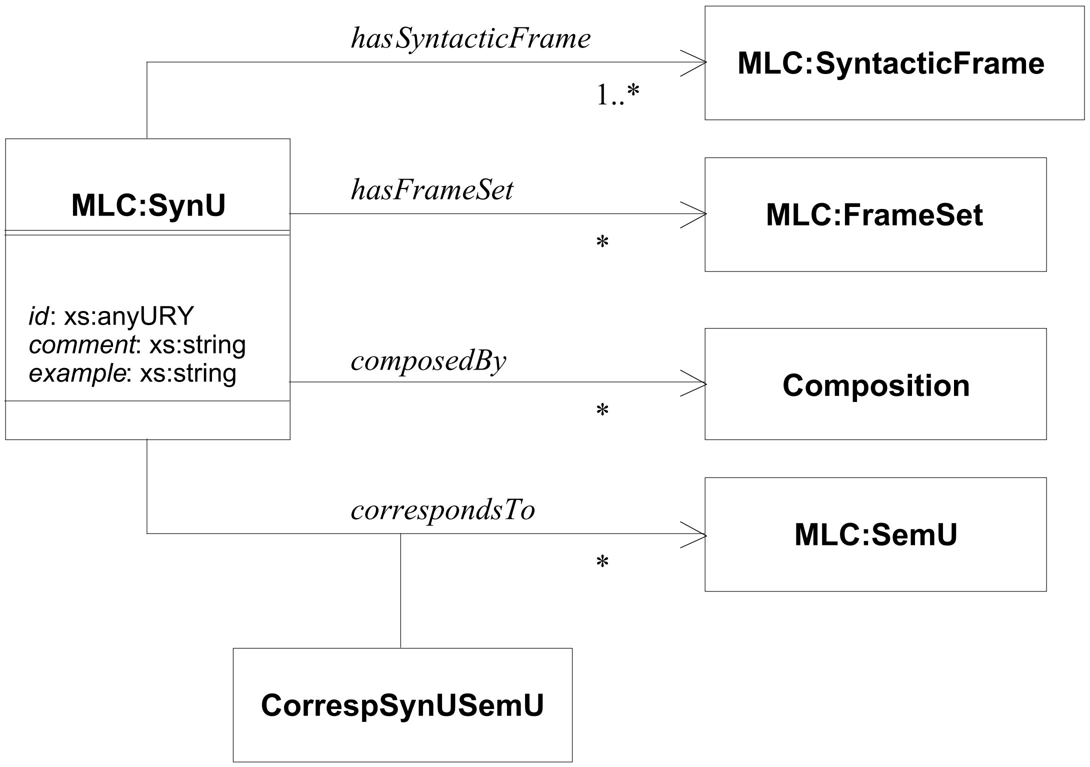

Figure 2. Lexical classes and their relations for the syntactic layer (SynU)

3 RDF instantiation

We have created an RDF schema for the syntactic layer of the ISLE/MILE lexical entry and instantiated one entry in several alternative forms to explore its potential as a representation for lexical data that can be integrated into the Semantic Web. The following describes the various components.

3.1.1 RDF schema for ISLE lexical entries

An RDF schema defines classes of objects and their relations to other objects. It does not in itself comprise an instance of these objects, but simply specifies the properties and constraints applicable to objects that conform to it.

The RDF schema for the syntactic layer of ISLE lexical entries can be accessed at http://www.cs.vassar.edu/~ide/rdf/isle-schema-v.6. The classes and relations (properties) defined in the schema correspond to the ER diagrams in Calzolari *et al.* (2003). The schema indicates that there is class of objects called **Entry**; a property declaration indicates that the relation *hasSynU* holds between **Entry** objects and **SynU** objects. Note that classes can be defined to be subclasses of other classes, in which case properties associated with the parent class are inherited. In the ISLE schema, for example, the objects **Self** and **SlotRealization** are defined to be sub-classes of **PhraseElement**, and the *hasPhrase* property holds between any object of type **PhraseElement** (including its sub-classes) and objects of type **Phrase**.

The ISLE RDF schema and entries have been validated using the ICS-FORTH Validating RDF Parser (VRP v2.1), which analyzes the syntax of a given RDF/ XML file according to the RDF Model and Syntax Specification[4] and checks whether the statements contained in both RDF schemas and resource descriptions satisfy the semantic constraints derived by the RDF Schema Specification.[5]

[4] http://www.w3.org/TR/rdf-syntax-grammar/

[5] http://www.w3.org/TR/rdf-schema/

4 ISLE Lexical Entries and the Data Category Registry

Appendix A contains three versions of the **SynU** description for "eat", instantiated as RDF objects. The first is a "full" version in which all of the information is specified, including atomic values (strings) at the leaves of the tree structure. The second two versions, rather than specifying all information explicitly, rely on the existence of a *Data Category Registry* (DCR) in which pre-defined lexical objects are instantiated and may be included in the entry by a direct reference.

The potential to develop a Data Category Registry in which lexical objects are instantiated in RDF is one of the most important for the creation of multi-lingual, reusable lexicons. It allows for the following:

1. specification of a universally accessible, standard set of morphological, syntactic, and semantic information that can serve as a reference for lexicons creators;
2. a fully modular specification of lexical entities that enables use of all or parts of the lexical information in the repository as desired or appropriate, to build more complex lexical information modules;
3. a template for data category description that lexicon creators can use to create their own data categories at any level of granularity;
4. means to reuse lexical specifications in entries sharing common properties, thereby eliminating redundancy as well as providing direct means to identify lexical entries or sub-entries with shared properties;
5. a universally accessible set of lexical information categories that may be used in applications or resources other than lexicons.

Note that the existence of a repository of lexical objects, instantiated and specified at different levels of complexity, does not imply that these objects must be used by lexicon creators. Rather, it provides a set of "off the shelf" lexical objects which either may be used as is, or which provide a departure point for the definition of new or modified categories.

The examples in Appendix A provide a general idea of how a repository of RDF-instantiated lexical objects can be used. Sample repositories at three different levels of granularity, corresponding to the examples in Appendix A, are given in Appendix B:

1. a repository of *enumerated classes* for lexical objects at the lowest level of granularity; this comprises a definition of sets of possible values for various lexical objects. Any object of this type must be instantiated with one of the listed values.
2. a repository of *phrase classes* which instantiate common phrase types, e.g., NP, VP, etc.
3. a repository of *constructions* containing instantiations of common syntactic constructions (e.g., for verbs which are both transitive and intransitive, as shown in the example).

The example entries demonstrate three different possibilities for the use of information in the repositories:

1. Entry 1 uses only the enumerated classes in the LDCR for **SynFeatureName** and **SynFeatureValue**. Note that in this case, the LDCR only provides a closed list of possible values, from which the assigned value in the entry must be chosen.
2. Entry 2 refers to instances of *phrase objects* in the LDCR rather than including them in the entry; this enables referring to a complex phrase (**Vauxhave** in the example) rather than including it directly in the entry, and provides the potential to reuse the same instance by reference in the same or other entries (this is done with **NP** in the example).
3. Entry 3 takes advantage of *construction instances* in the LDCR, thus eliminating the full specification in the entry and, again, allowing for reuse in other entries.

5 Summary

This exercise is intended to exemplify how RDF may be used to instantiate lexical objects at various levels of granularity, which can be used and reused to create lexical entries within a single lexicon as well as across lexicons. By relying on the developing standardized technologies underlying the Semantic Web, we ensure universal accessibility and commonality. Ultimately, lexical objects defined in this way can be used not only for lexicons, but also in language processing and other applications.

This example serves primarily as a proof of concept that may be refined and modified as we consider in more depth the exact RDF representation that would best serve the needs of lexicon creation. However, the potential of exploiting the developments in the Semantic Web world for lexicon development should be clear. More importantly, by situating our work in the context of W3 standards, we are in step with ISO TC37/SC4 vision of a Linguistic Annotation Framework that includes a Data Category Registry of the type we describe here.

References

Atkins, S., Bel, N., Bertagna, F., Bouillon, P., Calzolari, N., Fellbaum, C., Grishman, R., Lenci, A., MacLeod, C., Palmer, M., Thurmair, G., Villegas, M., Zampolli A., 2002. “From Resources to Applications. Designing the Multilingual ISLE Lexical Entry”, *Proceedings of LREC 2002*, Las Palmas, Canary Islands, Spain: 687-693.

Calzolari, N., Bertagna, F., Lenci, A., Monachini, M., 2003. *Standards and best Practice for Multilingual Computational Lexicons and MILE (Multilingual ISLE Lexical Entry)*, ISLE Computational Lexicon Working Group deliverables D2.2 – D3.2, Pisa.

Appendix A: Sample Entries

ENTRY 1 : Full entry

Highlighted lines refer to objects whose values are constrained in DCR definitions (Appendix B).

```
<?xml version="1.0"?>
<!-- Sample ISLE lexical Entry for EAT (transitive), SynU only
     Abbreviated syntax version using no pre-defined objects
     2002/10/23 Author: Nancy Ide -->
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
         xmlns:mlc="http://www.cs.vassar.edu/~ide/rdf/isle-schema-v.6#"
         xmlns="http://www.cs.vassar.edu/~ide/rdf/isle-schema-v.6#">
<Entry rdf:ID="eat1">
   <!-- The SynU for eat1 -->
   <hasSynu rdf:parseType="Resource">
      <SynU rdf:ID="eat1-SynU">
         <example>John ate the cake</example>
         <hasSyntacticFrame>
            <SyntacticFrame rdf:ID="eat1SynFrame">
               <hasSelf>
                  <Self rdf:ID="eat1Self">
                     <headedBy>
                        <Phrase rdf:ID="Vauxhave">
                           <hasSynFeature>
                              <SynFeature>
                                 <hasSynFeatureName rdf:value="aux"/>
                                 <hasSynFeatureValue rdf:value="have"/>
              </SynFeature></hasSynFeature></Phrase></headedBy></Self></hasSelf>
               <hasConstruction>
                  <Construction rdf:ID="eat1Const">
                     <slot>
                        <SlotRealization rdf:ID="NPsubj">
                            <hasFunction rdf:value="Subj"/>
                            <filledBy rdf:value="NP"/>
                     </SlotRealization></slot>
                     <slot>
                        <SlotRealization rdf:ID="NPobj">
                             <hasFunction rdf:value="Obj"/>
                             <filledBy rdf:value="NP"/>
              </SlotRealization></slot></Construction></hasConstruction>
                <hasFrequency rdf:value="8788" mlc:corpus="PAROLE"/>
</SyntacticFrame></hasSyntacticFrame></SynU></hasSynu></Entry></rdf:RDF>
```

ENTRY 2 : Using DCR categories for PHRASE

The highlighted lines refer to pre-instantiated lexical objects. A portion of the LDCR for Phrases is given in Appendix C. The URL reference is to the actual web address where the object is instantiated.

```
<?xml version="1.0"?>
<!--
     Sample ISLE lexical Entry for EAT (transitive), SynU only
     Abbreviated syntax version using no pre-defined objects
     2002/10/23 Author: Nancy Ide -->
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
         xmlns:mlc="http://www.cs.vassar.edu/~ide/rdf/isle-schema-v.6#"
         xmlns="http://www.cs.vassar.edu/~ide/rdf/isle-schema-v.6#">
<Entry rdf:ID="eat1">
   <!-- The SynU for eat1 -->
   <hasSynu rdf:parseType="Resource">
      <SynU rdf:ID="eat1-SynU">
         <example>John ate the cake</example>
         <hasSyntacticFrame>
```

```
            <SyntacticFrame rdf:ID="eat1SynFrame">
               <hasSelf>
                  <Self rdf:ID="eat1Self">
                    <headedBy rdf:resource=
                    "http://www.cs.vassar.edu/~ide/rdf/isle-datcats/Phrases#Vauxhave"/>
               </Self></hasSelf>
               <hasConstruction>
                  <Construction rdf:ID="eat1Const">
                     <slot>
                       <SlotRealization rdf:ID="NPsubj">
                         <hasFunction rdf:value="Subj"/>
                         <filledBy rdf:resource=
                         "http://www.cs.vassar.edu/~ide/rdf/isle-datcats/Phrases#NP"/>
                     </SlotRealization></slot>
                     <slot>
                       <SlotRealization rdf:ID="NPobj">
                         <hasFunction rdf:value="Obj"/>
                         <filledBy rdf:resource=
                         "http://www.cs.vassar.edu/~ide/rdf/isle-datcats/Phrases#NP"/>
                </SlotRealization></slot></Construction></hasConstruction>
                <hasFrequency rdf:value="8788" mlc:corpus="PAROLE"/>
</SyntacticFrame></hasSyntacticFrame></SynU></hasSynu></Entry></rdf:RDF>
```

ENTRY 3 : Using DCR categories for CONSTRUCTION

The highlighted lines refer to a pre-instantiated **Construction** object. A portion of the DCR for Constructions is given in Appendix B. The URL reference is to the actual web address where the object is instantiated.

```
<?xml version="1.0"?>
<!-- Sample ISLE lexical Entry for EAT (transitive)
     Abbreviated syntax version using pre-defined construction
     2002/10/23 Author: Nancy Ide -->
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
         xmlns:mlc="http://www.cs.vassar.edu/~ide/rdf/isle-schema-v.6#"
         xmlns="http://www.cs.vassar.edu/~ide/rdf/isle-schema-v.6#">
<Entry rdf:ID="eat1">
   <!-- The SynU for eat1 -->
   <hasSynu rdf:parseType="Resource">
      <SynU rdf:ID="eat1-SynU">
         <example>John ate the cake</example>
         <hasSyntacticFrame>
            <SyntacticFrame rdf:ID="eat1SynFrame">
               <hasSelf>
                  <Self rdf:ID="eat1Self">
                     <headedBy rdf:resource=
                    "http://www.cs.vassar.edu/~ide/rdf/isle-datcats/Phrases#Vauxhave"/>
                  </Self></hasSelf>
               <hasConstruction rdf:resource=
          "http://www.cs.vassar.edu/~ide/rdf/isle-datcats/Constructions#TransIntrans"/>
               <hasFrequency rdf:value="8788" mlc:corpus="PAROLE"/>
            </SyntacticFrame></hasSyntacticFrame></SynU></hasSynu></Entry></rdf:RDF>
```

Appendix B: DCR definitions

Sample DCR entries specifying enumerated values for **SynFeatureName**, etc. The specification uses the Ontology Web Language (OWL) to list valid values for objects of the defined class.

```
<!-- Enumerated classes for ISLE lexical entries v0.1 2002/10/23 Author: Nancy Ide  -->
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
         xmlns:owl ="http://www.w3.org/2002/07/owl#
         xmlns:isle ="http://www.cs.vassar.edu/~ide/rdf/isle-schema-v.6#">
```

```
<rdfs:Class rdf:about=
            "http://www.cs.vassar.edu/~ide/rdf/isle-enumerated-classes#FunctionType">
<owl:oneOf>
   <rdf:Seq>
      <rdf:li>Subj</rdf:li>
      <rdf:li>Obj</rdf:li>
      <rdf:li>Comp</rdf:li>
      <rdf:li>Arg</rdf:li>
      <rdf:li>Iobj</rdf:li>
</rdf:Seq></owl:oneOf></rdfs:Class>
<rdfs:Class rdf:about=
     "http://www.cs.vassar.edu/~ide/rdf/isle-enumerated-classes#SynFeatureName">
<owl:oneOf>
   <rdf:Seq>
      <rdf:li>tense</rdf:li>
      <rdf:li>gender</rdf:li>
      <rdf:li>control</rdf:li>
      <rdf:li>person</rdf:li>
      <rdf:li>aux</rdf:li>
</rdf:Seq></owl:oneOf> </rdfs:Class>
<rdfs:Class rdf:about=
     "http://www.cs.vassar.edu/~ide/rdf/isle-enumerated-classes#SynFeatureValue">
<owl:oneOf>
   <rdf:Seq>
      <rdf:li>have</rdf:li>
      <rdf:li>be</rdf:li>
      <rdf:li>subject_control</rdf:li>
      <rdf:li>object_control</rdf:li>
      <rdf:li>masculine</rdf:li>
      <rdf:li>feminine</rdf:li>
</rdf:Seq></owl:oneOf></rdfs:Class></rdf:RDF>
```

Sample LDCR entry for two Phrase objects

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
         xmlns:mlc="http://www.cs.vassar.edu/~ide/rdf/isle-schema-v.6#">

<Phrase rdf:ID="NP" rdfs:label="NP"/>

<Phrase rdf:ID="Vauxhave">
   <hasSynFeature>
     <SynFeature>
        <hasSynFeatureName rdf:value="aux"/>
        <hasSynFeatureValue rdf:value="have"/>
      </SynFeature></hasSynFeature></Phrase></rdf:RDF>
```

Sample LDCR entry for a Construction object

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
          xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
          xmlns="http://www.cs.vassar.edu/~ide/rdf/isle-schema-v.6#">
<Construction rdf:ID="TransIntrans">
    <slot>
       <SlotRealization rdf:ID="NPsubj">
          <hasFunction rdf:value="Subj"/>
          <filledBy rdf:resource=
                   "http://www.cs.vassar.edu/~ide/rdf/isle-datcats/Phrases#NP"/>
    </SlotRealization></slot>
    <slot>
       <SlotRealization rdf:ID="NPobj">
          <hasFunction rdf:value="Obj"/>
          <filledBy rdf:resource=
                  "http://www.cs.vassar.edu/~ide/rdf/isle-datcats/Phrases#NP"/>
</SlotRealization></slot></Construction></rdf:RDF>
```