Reference number of working document: **ISO/TC 37/SC 4 N330 Rev.13**

Date: 2006-11-30

ISO CD 24613:2006

Committee identification: ISO/TC 37/SC 4

Secretariat: KATS

Language resource management—Lexical markup framework (LMF)

Warning

This document is not an ISO International Standard. It is distributed for review and comment. It is subject to change without notice and may not be referred to as an International Standard.

Recipients of this document are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

Document type: International standard
Document subtype: if applicable
Document stage: 30.00
Document language: en

Table of contents

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 3.

Draft International Standards adopted by the technical committees are circulated to the member bodies for voting. Publication as an International Standard requires approval by at least 75 % of the member bodies casting a vote.

International Standard 24613 was prepared by Technical Committee ISO/TC 37, *Terminology and other language resources*, Subcommittee SC 4, *Language resource management*.

ISO 24613 is designed to coordinate closely with ISO Draft Revision 12620, *Computer applications in terminology – Data categories –Data category registry,* and ISO DIS 16642, *Computer applications in terminology – TMF (Terminological Markup Framework)*.

Annexes A, C, E, G, I, K, M, O form an integral part of this International Standard.

Introduction

Optimizing the production, maintenance and extension of electronic lexical resources is one of the crucial aspects impacting human language technologies (HLT) in general and natural language processing (NLP) in particular, as well as human-oriented translation technologies. A second crucial aspect involves optimizing the process leading to their integration in applications. Lexical Markup Framework (LMF) is an abstract metamodel that provides a common, standardized framework for the construction of computational lexicons. LMF ensures the encoding of linguistic information in a way that enables reusability in different applications and for different tasks. LMF provides a common, shared representation of lexical objects, including morphological, syntactic, and semantic aspects.

The goals of LMF are to provide a common model for the creation and use of electronic lexical resources ranging from small to large in scale, to manage the exchange of data between and among these resources, and to facilitate the merging of large numbers of different individual electronic resources to form extensive global electronic resources. The ultimate goal of LMF is to create a modular structure that will facilitate true content interoperability across all aspects of electronic lexical resources.

The LMF core package describes the basic hierarchy of information included in a lexical entry. The core package is supplemented by various resources that are part of the definition of LMF. These resources include:

- Specific data categories used by the variety of resource types associated with LMF, both those data categories relevant to the metamodel itself, and those associated with the extensions to the core package;

- The constraints governing the relationship of these data categories to the metamodel and to its extensions;

- Standard procedures for expressing these categories and thus for anchoring them on the structural skeleton of LMF and relating them to the respective extension models;

- The vocabularies used by LMF to express related informational objects for describing how to extend LMF through linkage to a variety of specific resources (extensions) and methods for analyzing and designing such linked systems.

Extensions of the core package which are documented in this standard in annexes include:

- Machine Readable Dictionaries

- Natural Language Processing electronic lexical resources

LMF extensions are expressed in a framework that describes the reuse of the LMF core components (such as structures, data categories, and vocabularies) in conjunction with the additional components required for a specific resource.

Types of individual instantiations of LMF can include such electronic lexical resources as fairly simple lexical databases, NLP and machine-translation lexicons, as well as electronic monolingual, bilingual and multilingual lexical databases. LMF provides general structures and mechanisms for analyzing and designing new electronic lexical resources, but LMF does not specify the structures, data constraints, and vocabularies to be used in the design of specific electronic lexical resources. LMF also provides mechanisms for analyzing and describing existing resources using a common descriptive framework. For the purpose of both designing new lexical resources and describing existing lexical resources, LMF defines the conditions that allow the data expressed in any one lexical resource to be mapped to the LMF framework, and thus provides an intermediate format for lexical data exchange.

1 Scope

This International Standard describes the Lexical Markup Framework (LMF), a high level model for representing data in lexical databases used with monolingual and multilingual computer applications.

LMF provides mechanisms that allow the development and integration of a variety of electronic lexical resource types[1]. These mechanisms will present existing lexicons as far as possible. If this is impossible, problematic information will be identified and isolated.

This standard is designed to be used in close conjunction with the metamodel presented in ISO 16642:2003, *Terminology Markup Framework* and with ISO 12620, *Terminology and other language resources — Data categories.*

2 Normative references

The following normative documents contain provisions that, through reference in this text, constitute provisions of ISO 24613. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on ISO 24613 are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative document referred to applies. Members of ISO and IEC maintain registers of currently valid International Standards.

ISO 639-1:2002, Codes for the representation of names of languages – Part 1: Alpha-2 Code.

ISO 639-2:1998, Code for the representation of languages – Part 2: Alpha-3 Code.

ISO DIS 639-3:2005, Codes for the representation of languages – Part 3: Alpha-3 Code for comprehensive coverage of languages.

ISO 1087-1:2000, Terminology – Vocabulary – Part 1: Theory and application.

ISO 1087-2:1999, Terminology – Vocabulary – Part 2: Computer application.

ISO/IEC 10646-1:2003, Information technology – Universal Multiple-Octet Coded Character Set (UCS).

ISO/IEC 11179-3:2003, Information Technology – Data management and interchange – Metadata Registries (MDR) – Part 3: Registry Metamodel (MDR3)

ISO 15924:2004, Information and documentation – Code for the representation of names of scripts.

ISO 16642:2003, Computer applications in terminology – TMF (Terminological Markup Framework).

[1] It supports existing lexical resource models such as the Genelex [9], the EAGLES International Standards for Language Engineering (ISLE) [5] and Multilingual ISLE Lexical Entry (MILE) models [6].

ISO 12620:200?, Terminology and other language and content resources – Data Categories – Specification of data categories and management of a data category registry for language resources.

ISO 24610:200?, Language resources management – Features structures – Part 1: Feature structure representation.

3 Terms and definitions

For the purposes of this International Standard, the terms and definitions given in ISO 1087-1, ISO 1087-2, ISO 12620:200? and the following apply:

3.1 abbreviated form

form resulting from the omission of any part of the **full form** of the same **lexeme**

3.2 adjunct

non-essential element associated with a verb as opposed to **syntactic arguments**

Example Alfred (syntactic argument) read a book (syntactic argument) today (adjunct)

NOTE Adverbs are possible adjuncts for a sentence.

3.3 affix

bound morpheme that may contribute to a form and which either serves as an inflectional or derivational element or participates in the process of agglutination or composition

NOTE Affixes function as prefixes (pre-positioned), suffixes (post-positioned) and infixes (inserted).

3.4 affixation

process in which an affix is added to a lexeme or a stem

3.5 agglutinated form

form that a **word** can take when used in a sentence or a phrase within an **agglutinating language**

3.6 agglutinating language

language where a word may consist of more than one morpheme but the boundaries between morphemes in the word are always clear-cut [16]

EXAMPLE Korean, Japanese, Hungarian and Turkish are agglutinating languages.

3.7 bound morpheme

morpheme that appears only together with one or several other **morphemes**

3.8 composition
compounding

word formation in which a new **word** is formed by adjoining at least two **lexemes**, in their original forms or with slight transformations

NOTE Composition should not be confused with agglutination and derivation, where bound morphemes are added to free ones.

3.9 compound

word built from two or more **lexemes**

NOTE For purposes of this standard, a compound is both a **word** and a **multiword expression**.

3.10 derivation

change in the **form** of a **word** to create a new **word**, usually by modifying the **stem** or by **affixation**

NOTE Sometimes derivation signals a change in part of speech, such as *nation* to *nationalize*. Sometimes the part of speech remains the same as in *nationalization* vs. *denationalization*.

3.11 derived form

form resulting from a **derivation**

3.12 form

sequence of morphemes or sequence of phonemes

3.13 free morpheme

morpheme that may stand by itself

EXAMPLE The English noun *boy*

3.14 full form

complete representation of a **word** for which there is an **abbreviated form**

3.15 grammatical feature

property induced from the **inflected, agglutinated, compound** or **derived form**

NOTE An example of a grammatical feature is: /grammatical gender/.

3.16 grapheme

atomic unit in a written language including letters, pictograms, ideograms, numerals, punctuation and other glyphs

3.17 human language technology

technology as applied to natural languages

3.18 inflected form

form that a **word** can take when used in a sentence or a phrase within an **inflectional language**

3.19 inflectional language

language where there is no clear-cut boundary between **morphemes** in that morphemes are generally fused together to yield a single, non-segmentable form [16]

EXAMPLE Spanish, Italian, French and English are inflectional languages.

3.20 interlingua

abstract intermediary language used in the machine translation of human languages

3.21 lemma
lemmatised form
citation form
headword

conventional **form** chosen to represent a **lexeme**

EXAMPLE In European languages, the **lemma** is usually the /singular/ if there is a variation in /number/, the /masculine/ **form** if there is a variation in /gender/ and the /infinitive/ for all verbs. In some languages, certain nouns are defective in the singular **form**, in which case, the /plural/ is chosen. In Arabic, for a verb, the lemma is usually considered as being the third person singular with the accomplished aspect.

3.22 lexeme

abstract unit generally associated with a set of **forms** sharing a common meaning

NOTE single words and multiword expressions are lexemes

3.23 lexical entry

container for managing one or several forms and possibly one or several meanings in order to describe a **lexeme**

3.24 lexical resource
lexical database

database consisting of one or several **lexicons**

3.25 lexicon

resource comprising **lexical entries** for a given language

NOTE A special language **lexicon** or a **lexicon** prepared for a specific NLP application can comprise a specific subset of language.

3.26 machine readable dictionary
MRD

electronic lexical resource designed to be consulted by human beings

NOTE Historically, MRDs were first computer representations of 'printed' dictionaries, that's why they are called *machine readable*.

3.27 machine translation lexicon

electronic **lexical resource** in which the individual **lexical entries** contain equivalents in two or more languages together with morphological, syntactic and semantic information to facilitate automatic or semi-automatic processing of lexical units during machine translation

3.28 morpheme

smallest unit of meaning expressed by a sequence of **phonemes** or a sequence of **graphemes**

EXAMPLE The word *boys* consists of two morphemes: *boy* and *s*.

3.29 morphology

description of the structure and formation of words.

3.30 multiword expression
MWE

lexeme made up of a sequence of lexemes that has properties that are not predictable from the properties of the individual lexemes or their normal mode of combination

NOTE An **MWE** can be a **compound**, a fragment of a sentence, or a sentence. The group of lexemes making up an MWE can be continuous or discontinuous. It is not always possible to mark an MWE with a **part of speech**.

EXAMPLE *to kick the bucket*, which means *to die* rather than *to hit a bucket with one's foot*.

3.31 natural language processing
NLP

field covering knowledge and techniques involved in the processing of linguistic data by a computer

3.32 orthography

way of spelling or writing **lexemes** that conforms to a conventionalized use

NOTE Aside from standardized spellings of alphabetical languages, such as standard UK or US English, or reformed German spelling, there can be variations such as transliterations of languages in non-native scripts, stenographic renderings, or representations in the International Phonetic Alphabet. In this regard, orthographic information in a **lexical entry** can describe a kind of transformation applied to the **form** that is the object of the entry. The specific value /native/ represents the absence of transformation.

3.33 paradigm class

set of **form** operations that build the various forms of a lexeme, possibly by **inflection, agglutination, compounding** or **derivation**

NOTE An inflectional paradigm class is not the explicit list of inflected forms. It usually references a prototypical class of inflectional forms, e.g., *ring,* as per *sing.*

3.34 part of speech lexical category grammatical category word class

category assigned to a **word** based on its grammatical and semantic properties

NOTE Typical parts of speech for European languages include: *noun, verb, adjective, adverb, preposition, etc.*

3.35 phoneme

smallest phonetic unit in a language

3.36 script

set of graphic characters used for the written **form** of one or more language (ISO/IEC 10646-1, definition 4.14)

NOTE The description of scripts ranges from a high level classification such as hieroglyphic or syllabic writing systems vs. alphabets to a more precise classification like Roman vs. Cyrillic. Scripts are defined by a list of values taken from ISO-15924. Examples are: Hiragana, Katakana, Latin and Cyrillic.

3.37 single word

word that does not contain any other **lexeme**

3.38 stem

linguistic unit whose form is smaller than or equal to the **form** of a single **lexeme** and that may be affected by an **inflectional, agglutinative, compositional** or **derivation** process

3.39 subcategorization frame valency

set of restrictions of a word indicating the properties of the **syntactic arguments** that can or must occur with this given word

3.40 support verb

verb that makes a generic semantic contribution to the context and that combines with a noun to form a lexicalised unit

EXAMPLES *take an exam* or *give an exam*. In these examples, *take* and *give* have only limited inherent meaning based on their semantics, but rather are used in a conventional, generic way to express a collocational conceptualization.

3.41 synset

linguistic element that links **synonyms**

NOTE The term stands for *synonym se*t and has been coined by the WordNet authors [8] but the notion was implemented and used long before WordNet.

3.42 syntactic argument

one of the essential and functional elements in a clause that identifies the participants in the process referred to by a verb

Example: Alfred (syntactic argument) read a book (syntactic argument) today (adjunct)

NOTE The subject, indirect object and direct object are possible syntactic arguments for a sentence.

3.43 transcription

form resulting from a coherent method of writing down speech sounds, to include converting speech sounds described in one writing system to an equivalent representation of the same speech sounds described in another writing system

3.44 transliteration

form resulting from the conversion of one writing system into another

3.45 variant

one of the alternative **forms** of a **word**

3.46 word

lexeme that has, as a minimal property, a **part of speech**

NOTE There are two types of words: **single words** and **compounds**. The description of a **word** can be more complete with more morphological information and/or syntactic and semantic information.

4 Key standards used by LMF

4.1 Unicode

LMF is Unicode compliant and presumes that all data are represented using Unicode character encodings.

4.2 ISO 12620 Data Category Registry (DCR)

The designers of an LMF conformant lexicon shall use data categories from the ISO 12620 Data Category Registry (DCR).

4.3 The ISO 639 family of standards

Language identifiers used in LMF-compliant resources shall conform to criteria specified in the ISO 639 family of standards. Some issues involving the combination of language and country codes, as well as the coordination of different parts of the ISO 639 standard have been addressed in external standards supported by the technology community. It is recommended that users should consult the current edition of IETF Best Common Practices (BCP) 47, *Tags for the Identification of Languages* in order to resolve issues involving choice of identifiers for use in electronic environments [1].

4.4 ISO 15924 ISO Codes for Script Identification

Script identifiers used in LMF-compliant resources shall conform to criteria specified in the ISO 15924 Codes for Script Identification.

4.5 Unified Modeling Language (UML)

LMF complies with the specifications and modeling principles of UML as defined by the Object Management Group (OMG) [2]. LMF uses a subset of UML that is relevant for linguistic description.

5 The LMF Model

5.1 Introduction

LMF models are represented by UML classes, associations among the classes, and a set of ISO 12620 data categories that function as UML attribute-value pairs. The data categories are used to adorn the UML diagrams that provide a high level view of the model. LMF specifications in the form of textual descriptions that describe the semantics of the modeling elements provide more complete information about classes, relationships, and extensions than can be included in UML diagrams.

In this process, lexicon developers shall use the classes that are specified in the **LMF core package** (section 5.2). Additionally, developers can optionally use classes that are defined in the **LMF extensions** (relevant annexes). Developers shall define a data category selection (DCS) as specified for **LMF data category selection procedures** (section 5.4).

5.2 LMF Core Package

The LMF core package is a metamodel that provides a flexible basis for building LMF models and extensions.

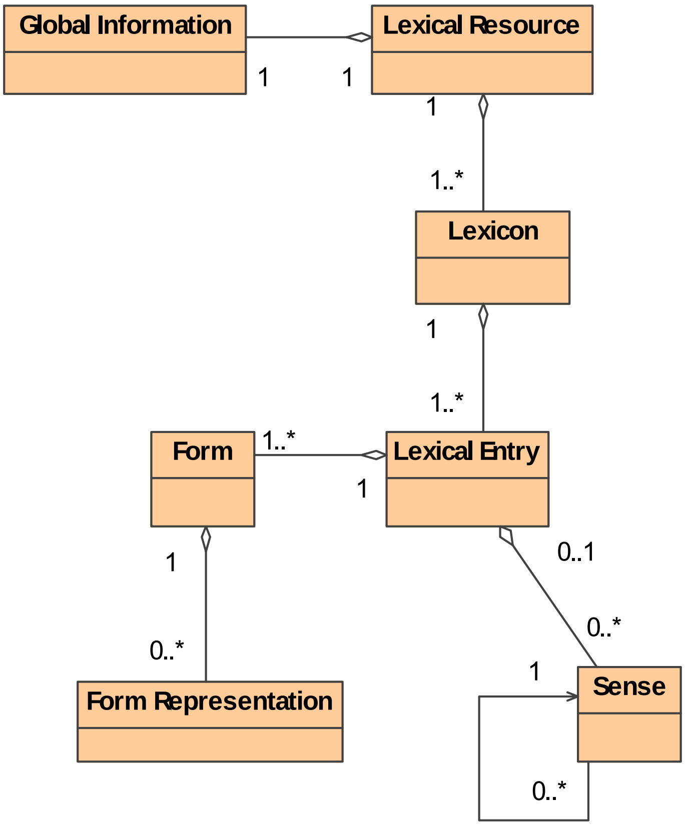


Figure 1: LMF core package

5.2.1 Lexical Resource class

Lexical Resource is a class representing the entire resource. *Lexical Resource* occurs once and only once. The *Lexical Resource* instance is a container for one or more lexicons.

5.2.2 Global Information class

Global Information is a class representing administrative information and other general attributes. There is an aggregation relationship between the *Lexical Resource* class and the *Global Information* class in that the latter describes the administrative information and general attributes of the entire resource. The *Global Information* class does not allow subclasses.

5.2.3 Lexicon class

Lexicon is a class containing all the lexical entries of a given language within the entire resource. A *Lexicon* instance must contain at least one lexical entry. The *Lexicon* class does not allow subclasses.

5.2.4 Lexical Entry class

Lexical Entry is a class representing a lexeme in a given language. The *Lexical Entry* is a container for managing the *Form* and *Sense* classes. Therefore, the *Lexical Entry* manages the relationship between the forms and their related senses. A *Lexical Entry* instance can contain one to many different forms, and can have from zero to many different senses. The *Lexical Entry* class does not allow subclasses.

5.2.5 Form class

Form class is a class representing one lexical variant of the written or spoken form of the lexical entry. A *Form* contains a Unicode string that represents the word form, as well as data categories that describe the attributes of the word form. The *Form* class itself may contain more than one orthographic variant (e.g. lemma, pronunciation, syllabification). The *Form* class allows subclasses.

5.2.6 Form Representation class

Form Representation is a class representing multiple orthographies. The *Form Representation* class supports the management of unique attribute-value sets describing an orthography when there is more than one orthography represented for the form (e.g. transliteration, pronunciation, or variant spelling), the *Form* instance may be associated with a *Form Representation* instance. A *Form Representation* instance contains a specific orthography and one to many data categories that describe the attributes of that orthography.

5.2.7 Sense Class

Sense is a class representing one meaning of a lexical entry. The *Sense* class allows subclasses. The *Sense* class allows for hierarchical senses in that a sense may be more specific than an other sense of the same lexical entry.

5.3 LMF Extension Use

All extensions conform to the LMF core package in the sense that each extension is anchored in a subset of the core package classes. An extension cannot be used to represent lexical data independently of the core package. Depending on the kind of linguistic data involved, an extension can depend on another extension. From the point of view of UML, an extension is a UML package. The dependencies of the various extensions are specified in the following diagram.

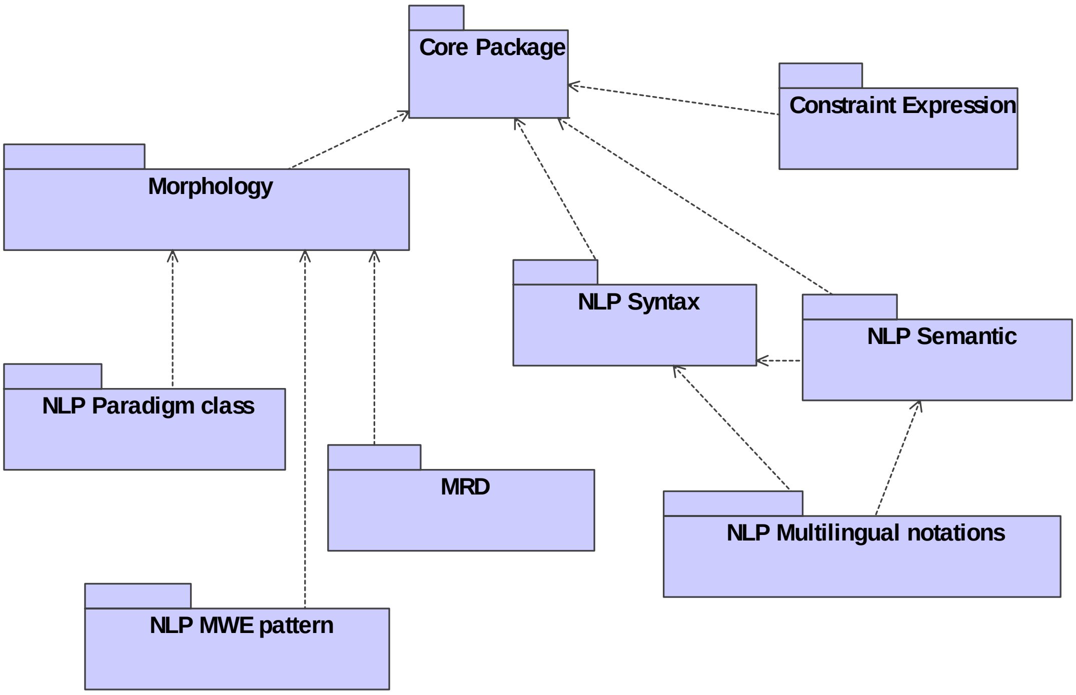


Figure 2: Dependencies between the LMF core and extension packages

Additional extensions may be developed over time. A new extension may either be based on the LMF core package itself or on an existing extension to the core package, or may be a combination of extension mechanisms from the core package and existing extensions.

The extension mechanisms include:

- the creation of subclasses based on UML modeling principles
- the addition of new classes
- constraints on the cardinality and type of associations
- specification of different anchor points for associations
- data category selections (DCSs)

The current LMF extensions are described in the annexes of this International Standard. Creators of lexicons should select the subsets of these possible extensions that are relevant to their needs.

5.4 LMF data category selection procedures

5.4.1 LMF Attributes

UML models such as LMF are adorned or further described by UML attributes, which provide information about specific properties or characteristics associated with the model. All LMF attributes are complex data categories. For a given class, all attributes are different. Each value of an attribute is either a simple data category or a Unicode string. Each attribute has only one value.

5.4.2 Data Category Registry (DCR)

The Data Category Registry (DCR) is a set of data category specifications defined by ISO 12620 [18] [19] [20]. The designers of any specific LMF lexicon shall rely on the DCR when creating their own data category selection.

5.4.3 Data Category Selection (DCS)

In the broadest sense, a data category selection can comprise all the data categories used by a given domain in the field of language resources. A DCS can also list and describe the set of data categories that can be used in a given LMF lexicon. The DCS also describes constraints on how the data categories are mapped to specific classes.

5.4.4 User-defined data categories

Lexicon creators can define a set of new data categories to cover data category concepts that are needed and that are not available in the DCR. This supplemental set of data categories shall be registered with the DCR Registration Authority and managed in conformance with ISO 12620.

5.4.5 Lexicon comparison

When two LMF conformant lexicons are based on two different DCSs, comparison of the DCS in each lexicon provides a framework for identifying what information can be exchanged between one format and the other, or what will be lost during a conversion. When LMF is used to describe an existing resource, it will be necessary to map the existing resource to corresponding data categories in the DCR.

5.5 LMF process

LMF shall be used according to the following steps.

Step 1: Define an LMF conformant lexicon

Step 2: Populate this lexicon

An LMF conformant lexicon is defined as the combination of an LMF core package, zero to many lexical extensions and a set of data categories. The combination of all these elements is described in the following UML activity diagram:

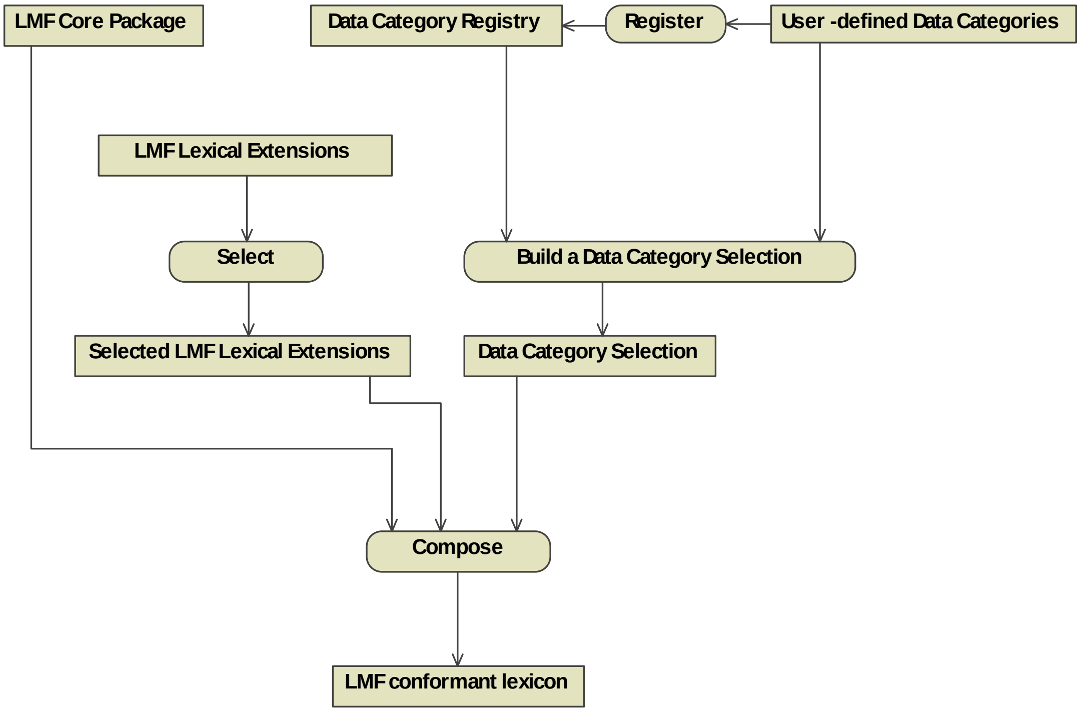


Figure 3: LMF Process

Annex A (normative) Morphology extension

A.1 Objectives

The purpose is to provide the mechanisms to support the development of lexicons that have an **extensiona**l description of the morphology of lexical entries.

Example: when applied to an inflectional languages, "extensional" means that all inflected forms will be explicitly described within one *Lexicon* instance.

Note: the mechanisms for an **intensional** description of the morphology are specified in the Paradigm class annex.

A.2 Class diagram

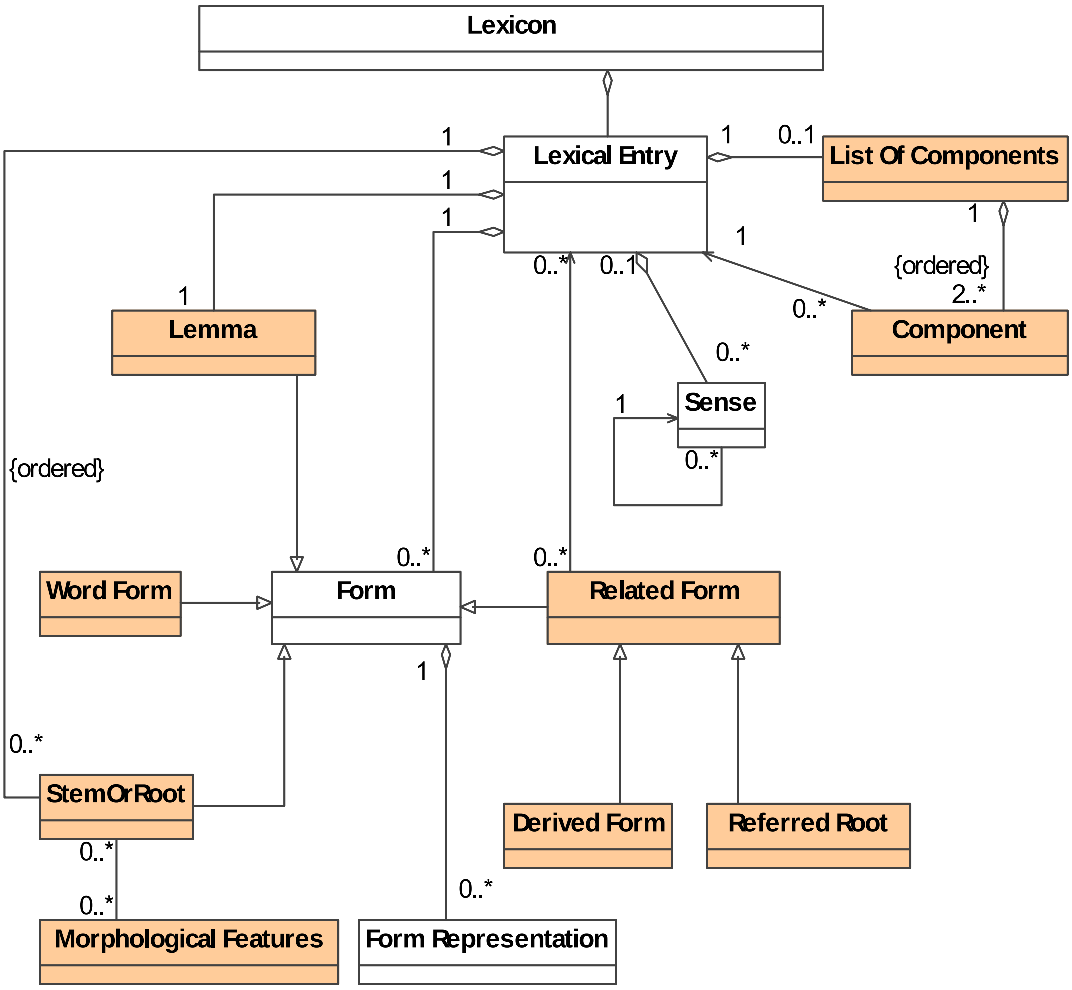


Figure A.1: Morphology class model

A.3 Description of morphology model

The morphology model manages two categories of *Form* subclasses: *Form* subclasses that represent sets of grammatical variants that make up the abstract lexeme, and *Form* subclasses representing words, morphemes, and MWEs that can be related to a form in another Lexical Entry. The former classes include the *Lemma*, *Word Form*, and *StemOrRoot*. The latter classes include the *Related Form* and its subclasses. The *Lexical Entry* is constrained on the Part of Speech.

A.3.1 Form subclasses

A.3.1.1 Lemma Class

Lemma is a *Form* subclass representing a word form chosen by convention to designate the *Lexical Entry*. The *Lemma* class is in a one to one aggregate association with the *Lexical Entry* that overrides the multiplicity inherited from the *Form* class. The lemma is usually equivalent to one of the inflected forms, the root or stem, or MWE, e.g. compound, idiomatic phrase. The convention for selecting the lemma can vary by language, language family, or editorial choice.

A.3.1.2 Word Form Class

Word Form is a Form subclass representing a word of type inflected or agglutinated.

A.3.1.3 StemOrRoot Class

StemOrRoot is a Form subclass representing a morpheme of type stem or root. The aggregation association between a *Stem* and *Lexical Entry* is ordered.

A.3.1.4 Related Form Class

Related Form is a *Form* subclass representing a word form or morpheme that can be related to the Lexical Entry in one of a variety of ways (e.g. derivation, root). The Related Form can be typed and has the following subclasses: Derivation Class and Referred Root Class. There is no assumption that the Related Form is associated with the Sense class in the Lexical Entry.

A.3.1.5 Derived Form Class

Derived Form is a *Related Form* subclass representing a word form of type derivation.

A.3.1.6 Referred root Class

Referred Root is a *Related Form* subclass representing a morpheme of type root. The *Referred Root* class specifically represents a root that is managed by a different Lexical Entry instance and is shared by two or more other Lexical Entry instances.

A.3.2 List Of Components Class

List Of Component is a class representing the aggregative aspect of a multiword expression. The *List Of Components* class is in a zero or one aggregate relationship with the *Lexical Entry* class. Each *List Of Component* instance should have at least two components.

The mechanism can also be applied recursively, that is a multiword expression may be comprised of components that are themselves multiword expressions. *List Of Components* class is used in MWE pattern package.

A.3.3 Component class

Component is a class representing a reference to a lexical entry when this latest one is an element of *List Of Component* class.

A.3.4 Morphological Features class

Morphological Features is a class representing an unordered combination of grammatical features.

Annex B (informative) Morphology examples

B.1 Introduction

This extension provides examples of how to develop models for MRD and NLP Morphology lexicons.

B.2 Example of class adornment

Classes may be adorned with the following attributes:

Class name	Example of attributes	Comment
Lemma	writtenForm phoneticForm geographicalVariant scheme	/writtenForm/ and /phoneticForm/ take Unicode strings as values.
Word Form	writtenForm phoneticForm hyphenation grammaticalNumber grammaticalGender grammaticalTense person	When /writtenForm/ is valued as "kitten", /hyphenation/ will be valued as "kit ten". /grammaticalNumber may be valued by /plural/ for instance.
StemOrRoot	writtenForm phoneticForm	
Derived Form	writtenForm phoneticForm	
Related Form	geographicalVariant	
Derived Form		
Referred Form	writtenForm	
Component Form		
List Of Components		
Morphological Features	grammaticalNumber grammaticalGender grammaticalTense person	

B.3 Example of word description

B.3.1 Example of a simple morphology

In the following example, the lexical entry is associated with a lemma *clergyman* and two inflected forms *clergyman* and *clergymen*.

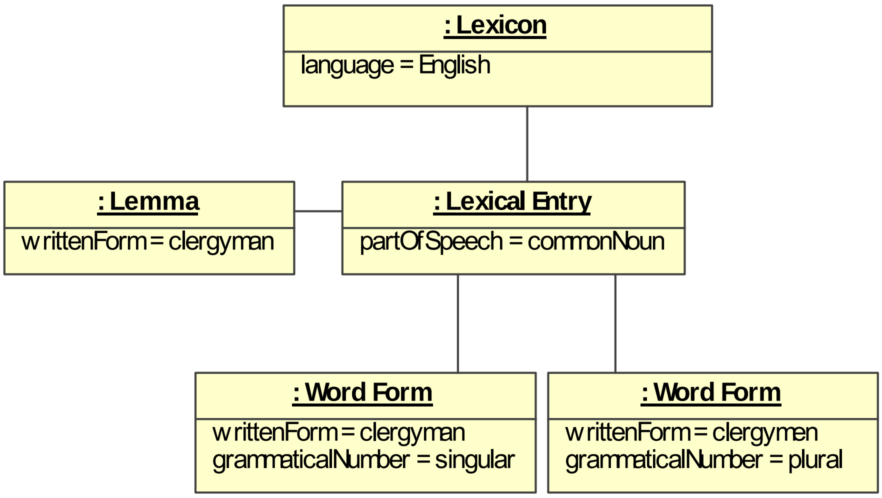


Figure B.1: Instance diagram for a simple example

The same data can be expressed by the following XML fragment:

```
<Lexicon>
   <DC att="language" val="English"/>
   <LexicalEntry>
      <DC att="partOfSpeech" val="commonNoun"/>
      <Lemma>
         <DC att="writtenForm" val="clergyman"/>
      </Lemma>
      <WordForm>
          <DC att="writtenForm" val="clergyman"/>
          <DC att="grammaticalNumber="singular"/>
      </WordForm>
      <WordForm>
         <DC att="writtenForm" val="clergymen"/>
         <DC att="grammaticalNumber="plural"/>
      </WordForm>
   </LexicalEntry>
</Lexicon>
```

It is also possible to precise the type of *Word Form* by adding a specific attribute /lexicalType/ as in the following instance diagram:

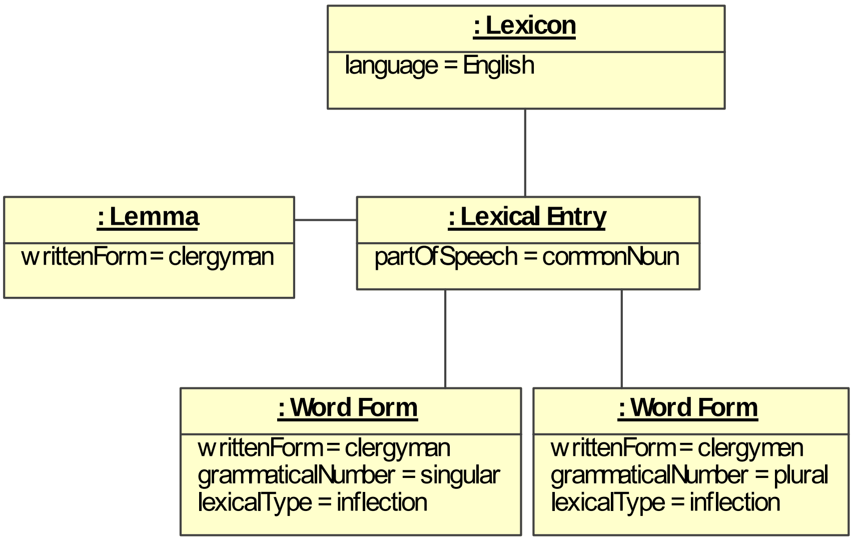


Figure B.2: highly specified Word Form example

B.3.2 Example of multiple scripts and orthographies

In the following example, the lexical entry is associated with a lemma with three different ways to express the word form [22]. The lexical entry is also associated with an inflected form with three different ways to express the word form.

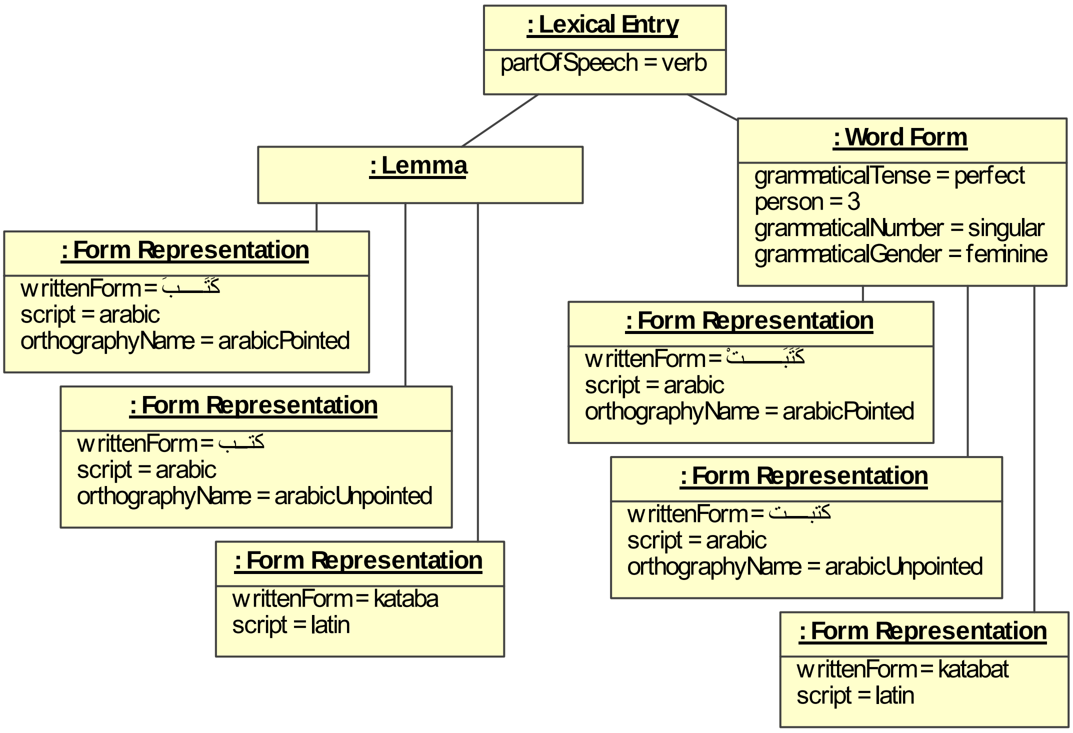


Figure B.3: example of multiple scripts and orthographies

It's worth noting that this strategy is not the only possible option in Arabic. Another strategy is to describe the Arabic pointed script forms in the lexicon and to provide an external mechanism to compute automatically the Arabic unpointed script forms and transliterations. In this case, Form Representation instances are not needed.

B.3.3 Example of Regional Variants

Regional variants can be modeled using the Form Representation class as follows:

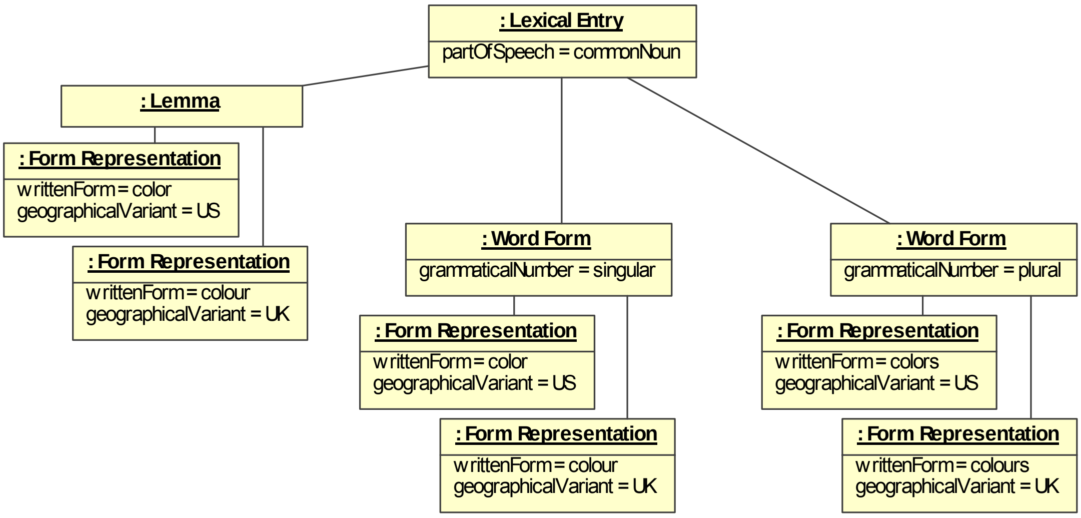


Figure B.4: example of regional variants using Form Representation

B.3.4 Example of Arabic root management

Arabic root is represented by a shared *Referred Root* instance. In the following instance diagram, the verb *kataba* and the noun *maktabatun* are both associated with the *Referred Root* instance ktb. Let's note that due to the fact that *Referred Root* class is a subclass of Form, a set of multiple representations may be recorded as in the previous Arabic example by the means of *Form Representation* instances.

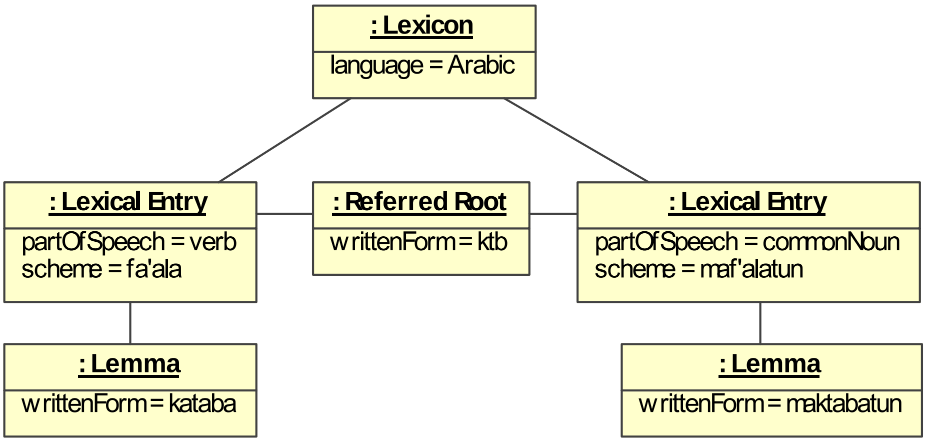


Figure B.5: example of Arabic root management

Annex C (normative) Machine Readable Dictionary extension

C.1 General Objectives

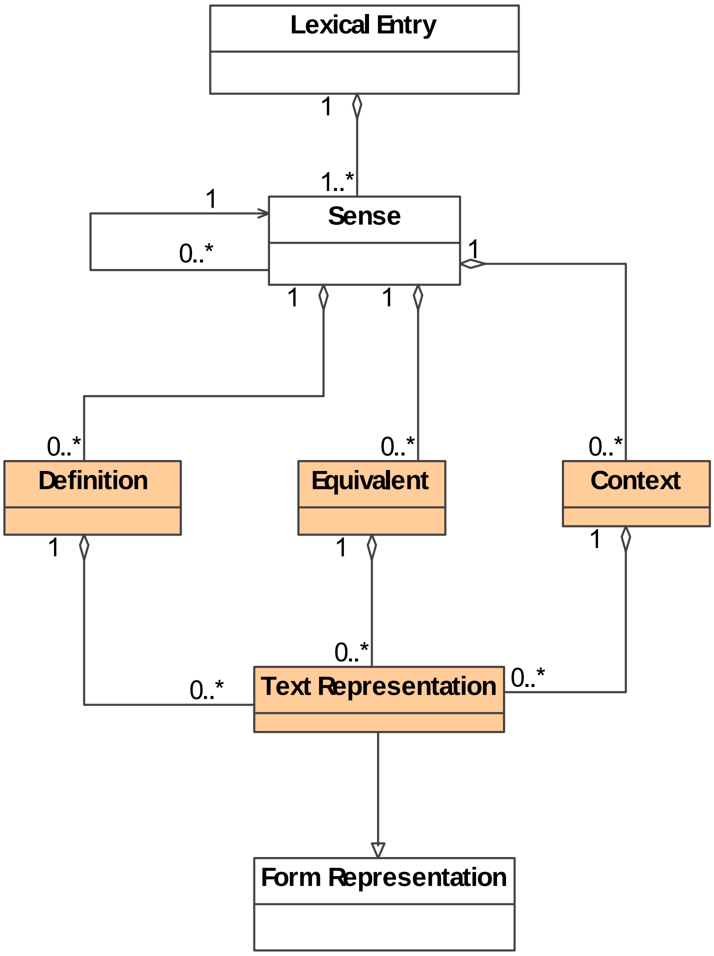
The Machine Readable Dictionary (MRD) extension provides metamodel packages for representing data stored in machine readable dictionaries. The extension supports electronic machine readable dictionary access for both human use and machine processing. Since the MRD extension is based on the LMF core package, it is designed to interchange data with other LMF extensions where applicable.

C.2 Dual Use MRD Package

C.2.1 Objectives

The objectives of the Dual Use MRD Package are to provide monolingual and bi-lingual dictionary support for human translators, support enterprise systems covering multiple languages and language families, support the preparation of lexical data for use in NLP systems, and directly support NLP systems (e.g. lexical data for named entity extraction).

C.2.2 Class Diagram



C.2.3 Description of the Dual Use MRD Metamodel

The dual use MRD metamodel is based on the NLP Morphological Extension with the following modifications:

- The dual use MRD package relaxes all constraints on the Related Form class. [NOTE: For example, the Related Form class can be typed or admit subclasses for the full range of word forms related to the Lexical Entry, e.g. synonym, antonym, abbreviation].
- The package requires a Sense class (1 to 1…* multiplicity).
- The package provides additional classes for the Sense class

C.2.4 Definition Class

The Definition class represents one definition of the word form managed by the Lemma class. The Definition class may be associated with zero to many Text Representation classes which manage the text definition in more than one language or script.

C.2.5 Equivalent Class

In a bilingual MRD, the Equivalent class represents the translation equivalent of the word form managed by the Lemma class. The Equivalent class is in a zero to many aggregate association with the Sense class, which allows the developer to omit the Equivalent class from a monolingual dictionary.

C.2.6 Context Class

The Context class represents a text string that provides authentic context for the use of the word form managed by the Lemma. The Context class is in a zero to many aggregate association with the Sense class and may be associated with zero to many Text Representation classes which manage the representation of the translation equivalent in more than one script or orthography.

Note: The context may use an inflected form of the Lemma.

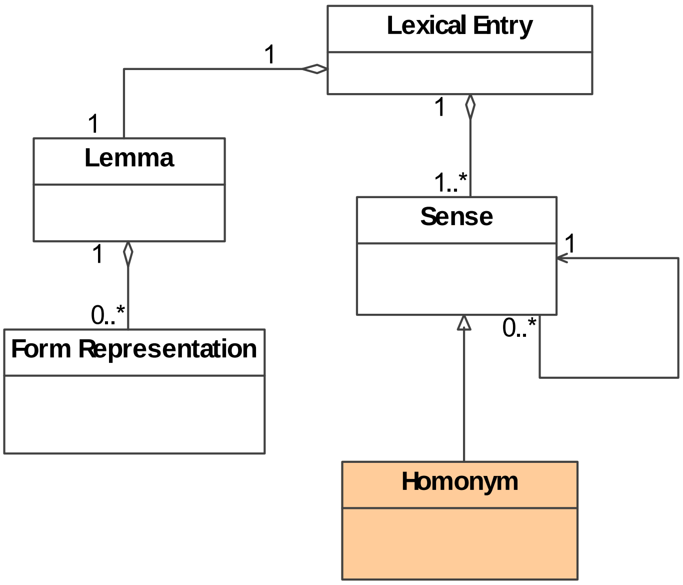
C.2.7 Text Representation Class

Text Representation is a subclass of the Form Representation class. A Text Representation class is associated with the child classes of the Sense class, not the Lexical Entry. A Text Representation class instance contains a specific orthography and one to many data categories that describe the attributes of that orthography. [Note: A developer is more likely to allocate language and script attributes to a Text Representation class than to a Form Representation class.]

C.3 MRD Package for POS Homonymy

The objective of the MRD Package for POS Homonmy is to allow the description of several parts of speech that are associated with one Lexical Entry instance.

C.4 Class diagram



C.5 Description of the POS homonymy package

The POS homonymy package is based on the NLP Morphology Extension with the following modifications:

- The Lemma is the only Form subclass managed by the Lexical Entry, since the Lemma represents homonyms with different parts of speech, each with different inflections, derivations, and related forms.

- The Sense class has a homonym subclass which allows the management of two or more homonyms related to the word form managed by the Lemma class.

- The Part of Speech constraint is relaxed allowing the Part of Speech to be allocated to the Homoym class.

- The Sense class may use the classes managed by the Sense class in the Dual Use MRD package.

C.5.1 Homonym Class

Homonym is a subclass of the Sense class. A Lexical Entry can manage zero to many Homonym classes and each Homoym class can manage zero to many Sense classes.

Example: In English *book* as noun and *book* as verb may be represented as two *POS Homonymy* instances that share a common *Lexical Entry* instance.

Annex D (informative) Machine Readable Dictionary examples

D.1 Dual Use MRD Example

D.1.1 Example of a Bilingual MRD with Multiple Representations

D.1.1.1 Introduction

The example of a bilingual MRD in Figure D.1 shows an entry containing the Arabic word ‘kitaab’ and two equivalents in English, ‘book’ (the most common meaning) and ‘credentials’. The transcriptions provide users more information about the pronunciation of the words and context than can be derived from the Arabic script. In this example, the Word Form class provides information about the form and pronunciation of the Arabic broken plural, which is an irregular inflection. The decision to include the *Form Representation* class is an editorial choice determined by the goals of the lexicon developer. If the goal were to produce an Arabic-English MRD that contained only Arabic script for the Arabic word forms, the inclusion of *Form Representation* class would not be necessary.

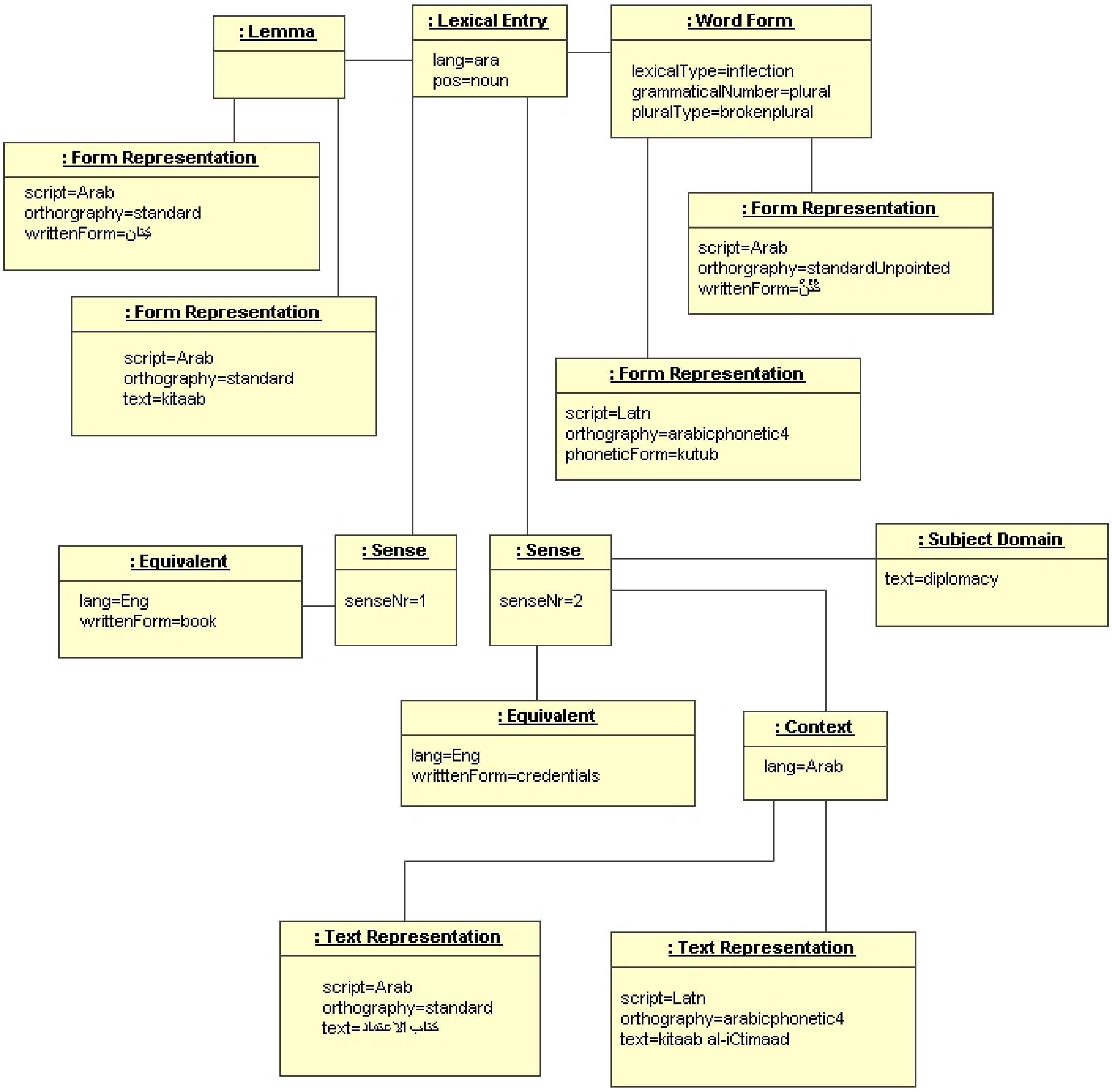


Figure D.1: Instantiation example for a bilingual MRD

D.2 MRD Example with POS Homonymy

The example of a monolingual MRD in Figure D.2 shows an entry containing the English word ‘bank’ as a noun and as a verb using the Homonym subclass of the Sense class. The example of the noun illustrates how the Sense superclass and the Homonym subclass are used in conjunction with each other in order to manage close polysemy.

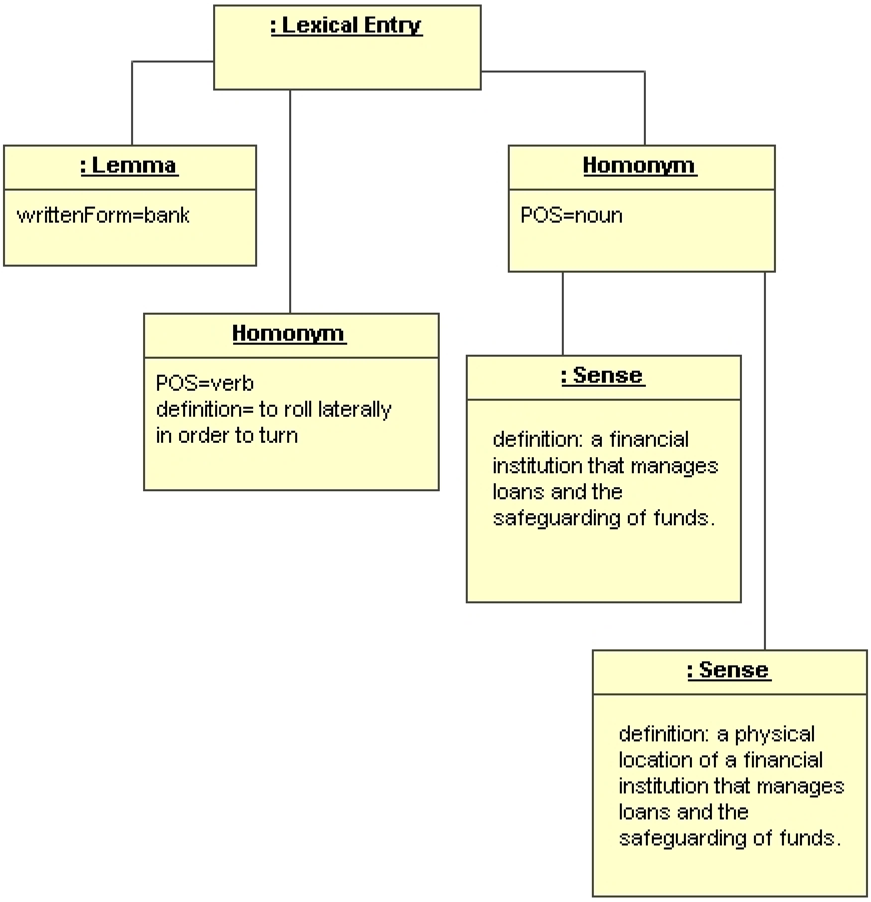

: Lexical Entry
: Lemma
writtenForm=bank
Homonym
POS=noun
Homonym
POS=verb
definition= to roll laterally in order to turn
: Sense
definition: a financial institution that manages loans and the safeguarding of funds.
: Sense
definition: a physical location of a financial institution that manages loans and the safeguarding of funds.

Figure D.2: Instantiation example for POS Homonymy

Annex E (normative) NLP syntax extension

E.1 Objectives

The purpose of this annex is to describe the properties of a word when combined with other words in a sentence. When recorded in a lexicon, the syntactic properties make up the syntactic description of a LexicalEntry instance.

This annex permits the description of specific syntactic properties of words and does not express the general grammar of a language.

E.2 Class diagram

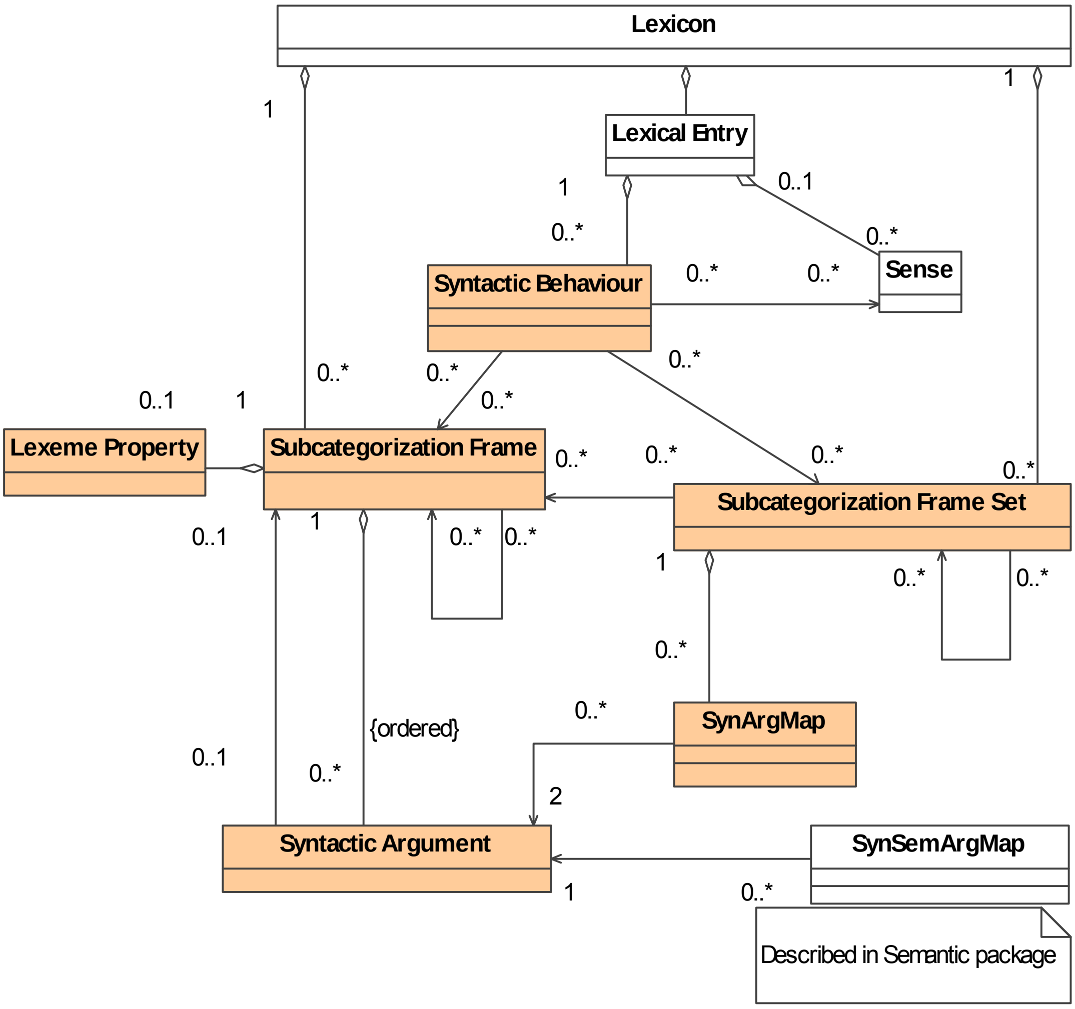


Figure E.1: Syntactic model

E.3 Description of the syntactic model

E.3.1 Syntactic Behaviour class

Syntactic Behaviour is a class representing one of the possible behaviours of a word. The *Syntactic Behaviour* instance is attached to the *Lexical Entry* instance and optionally to the *Sense* instance. The presence in a given lexicon of one *Syntactic Behaviour* instance for a lexical entry means that this word can have this behaviour in the language of the lexicon.

Syntactic description is optional, so it is possible to describe morphology and semantics without any syntactic description. *Lexical Entry*, *Syntactic Behaviour* and *Sense* instances form a triangle representing Morphology, Syntax and Semantics.

Detailed description of the syntactic behaviour of a lexical entry is defined by the *Subcategorization Frame* instance.

E.3.2 Subcategorization Frame class

Subcategorization Frame is a class representing one syntactic construction. A *Subcategorization Frame* instance is shared by all *Lexical Entry* instances that have the same syntactic behaviour in the same language. A *Subcategorization Frame* can inherit relationships and attributes from another more generic *Subcategorization Frame* by means of a reflexive link. Therefore, it is possible to integrate a hierarchical structure of *Subcategorization Frame* instances.

Example: In a L*exical Entry* for the Italian verb *amare*, a *Syntactic Behaviour* instance may be created and associated with a *Subcategorization Frame* instance called *regularSVOAvere*. This latter instance describes the regular subject, verb, object structure with a verb using the auxiliary *avere*.

E.3.3 Lexeme Property class

Lexeme Property is a class representing the central node of the *Subcategorization Frame* and is the class that refers to the current *Lexical Entry* instance. A *Lexeme Property* instance connected to a *Subcategorization Frame* instance is shared by all the words that have the same syntactic behaviour.

Example: In the Italian example, the attribute auxiliary on the *Lexeme Property* instance may be set to *avere*.

E.3.4 Syntactic Argument class

Syntactic Argument is a class representing an argument of a given *Subcategorization Frame*. A S*yntactic Argument* can be linked recursively to a Subcategorization Frame instance in order to describe deeply complex arguments. *Syntactic Argument* allows the connection with a semantic argument by means of a *SynSemArgMap* instance.

E.3.5 Subcategorization Frame Set class

Subcategorization Frame Set is a class representing a set of syntactic constructions and possibly the relationship between these constructions. A *Subcategorization Frame Set* can inherit relationships and attributes from another more generic *Subcategorization Frame Set* by means of means of a reflexive link. Therefore, it is possible to integrate a hierarchical structure of *Subcategorization Frame Set* instances.

A *Subcategorization Frame Set* groups various syntactic constructions that appear frequently for certain sets of words. The objective is to factorize syntactic descriptions and to maintain a minimum of syntactic behaviour instances in the lexicon.

E.3.6 SynArgMap class

The *SynArgMap* is a class representing the relationship that maps various *Syntactic Argument* instances of the same *Subcategorization Frame Set* instance.

Annex F (informative) NLP syntax examples

F.1.1 Example of class adornment

Classes may be adorned with the following attributes:

Class name	Example of attributes	Comment
Syntactic Behaviour	id label	
Subcategorization Frame	id label comment	
Lexeme Property	partOfSpeech mood voice auxiliary position	The /position/ data category may specify the relative position of the word in the sentence with respect to the syntactic arguments.
Syntactic Argument	function syntacticConstituent introducer label restriction	The /function/ data category may have values like /subject/ or /object/. The /syntacticConstituent/ may have values like /NP/ or /PP/ for *Noun Phrase* and *Prepositional Phrase* respectively. The /introducer/ may specify the preposition that is required to introduce the /syntacticConstituent/.
Subcategorization Frame Set	id label example comment	
SynArgMap	comment	

F.1.2 Examples of word description

F.1.2.1 Example in Italian

The example shown in Figure F.1 is taken from the Parole/CLIPS lexicon [3]. In this example, only syntactic structures are used, and no semantic information is described. The syntactic construction being described is a rather simple construction in Italian, where both the subject and the direct object have the simple data category property /nounPhrase/. The *Lexeme Property* instance describes a verb that takes the auxiliary *avere*. A typical example of such a construction is *Gianni ama Maria*.

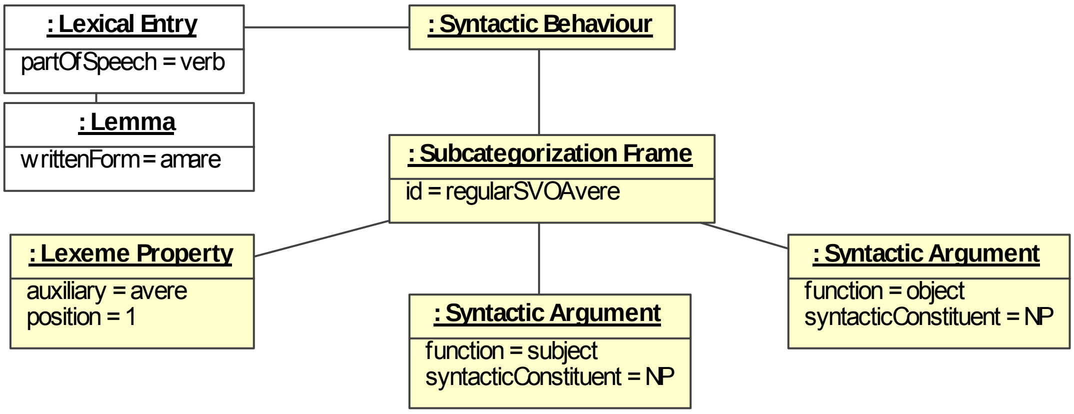


Figure F.1: Instance diagram in Italian

The same data can be expressed by the following XML fragment:

```
<LexicalEntry>
   <DC att="partOfSpeech" val="verb"/>
   <Lemma>
      <DC att="writtenForm" val="amare"/>
   </Lemma>
   <SyntacticBehaviour subcategorizationFrames="regularSVOAvere"/>
</LexicalEntry>
<SubcategorizationFrame id="regularSVOAvere">
   <LexemeProperty>
      <DC att="auxiliary" val="avere"/>
      <DC att="position" val="1"/>
   </LexemeProperty>
   <SyntacticArgument>
      <DC att="function" val="subject"/>
      <DC att="syntacticConstituent" val="NP"/>
   </SyntacticArgument>
   <SyntacticArgument>
      <DC att="function" val="object"/>
      <DC att="syntacticConstituent" val="NP"/>
   </SyntacticArgument>
</SubcategorizationFrame>
```

F.1.2.2 Example in English

In English, it is possible to use just one *Subcategorization Frame Set* for certain ergative verbs. For example, *boil* in *he boils a kettle of water* and *the kettle boils*, thus this verb may be described by means of only one syntactic behaviour, instead of two. So, only one *Subcategorization Frame Set* instance is required as shown in the following figure.

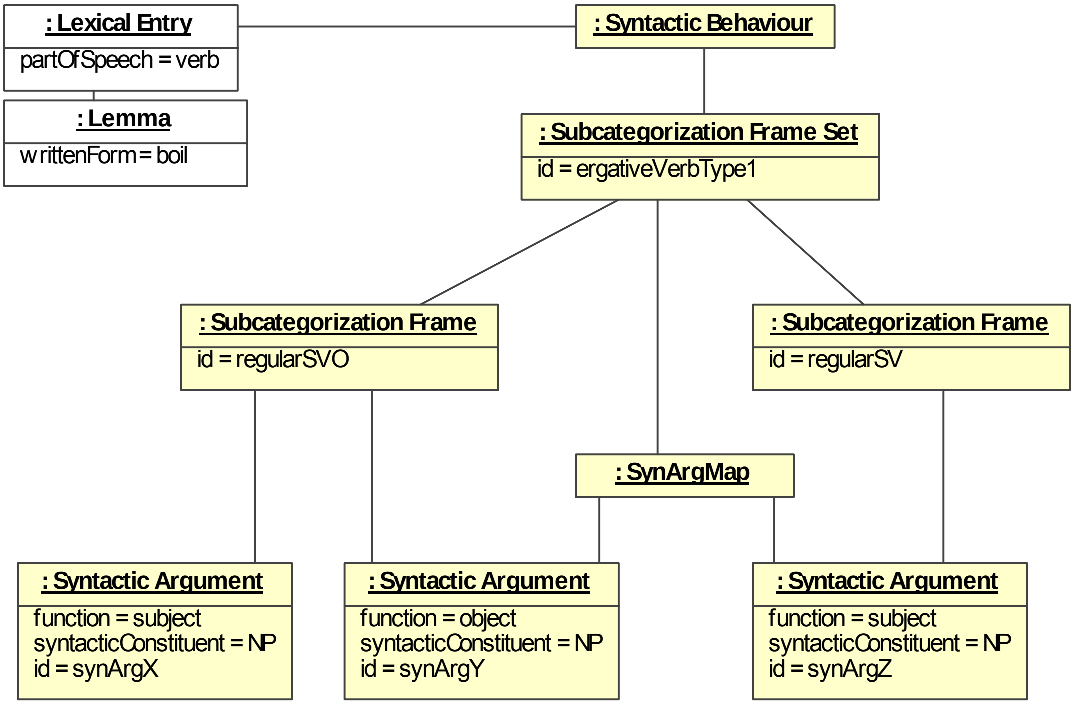


Figure F.2: Instance diagram in English

The same data can be expressed by the following XML fragment:

```
<LexicalEntry>
   <DC att="partOfSpeech" val="verb"/>
   <Lemma>
      <DC att="writtenForm" val="boil"/>
   </Lemma>
   <SyntacticBehaviour subcategorizationFrameSets="ergativeVerbType1"/>
</LexicalEntry>
<SubcategorizationFrameSet  id= "ergativeVerbType1"
   subcategorizationFrames= "regularSVO regularSV"/>
   <SynArgMap arg1="synArgY" arg2="synArgZ"/>
</SubcategorizationFrameSet>
<SubcategorizationFrame id="synArgX">
   <DC att="function" val="subject"/>
   <DC att="syntacticConstituent" val="NP"/>
</SubcategorizationFrame>
<SubcategorizationFrame id="synArgY">
   <DC att="function" val="object"/>
   <DC att="syntacticConstituent" val="NP"/>
</SubcategorizationFrame>
<SubcategorizationFrame id="synArgZ">
   <DC att="function" val="subject"/>
   <DC att="syntacticConstituent" val="NP"/>
</SubcategorizationFrame>
```

Annex G (normative) NLP semantics extension

G.1 Objectives

The purpose of this section is to describe one sense and its relationship with other senses belonging to the same language. Due to the intricate interactions between syntax and semantics in most languages, this section also provides the connection to syntax. The linkage of senses belonging to different languages will be described using the multilingual notations annex.

G.2 Class diagram

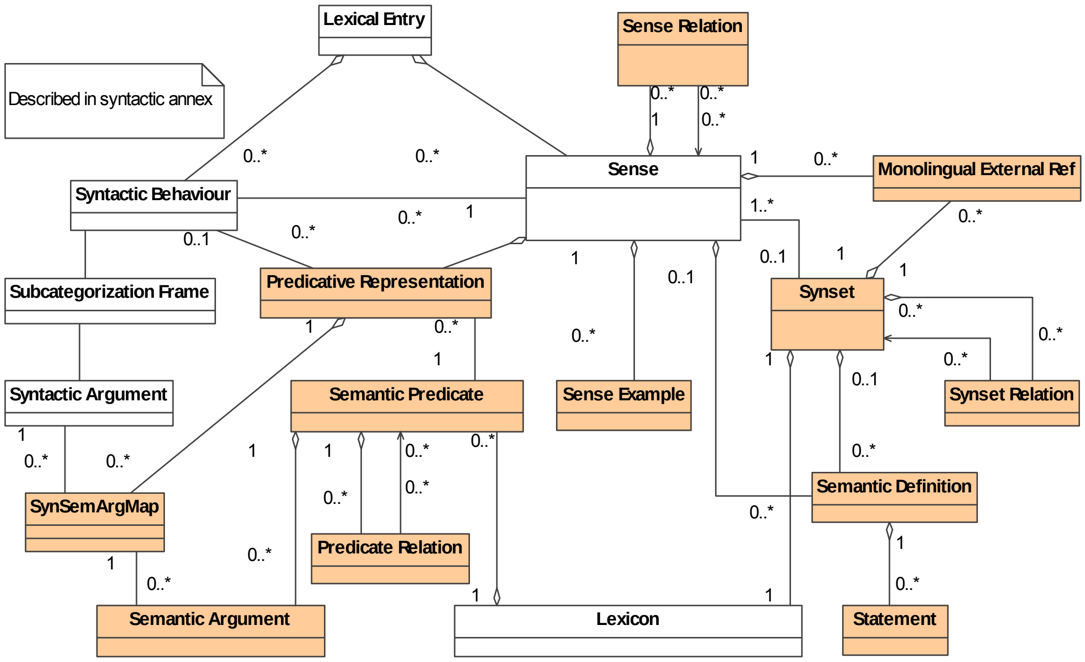


Figure G.1: Semantic model

G.3 Connection with the core package

The *Sense* class is specified in the core package. The *Sense* class is aggregated in the L*exical Entry* class. Therefore, a *Sense* instance is not shared among two different Lexical Entry instances.

G.4 Description of the semantic model

G.4.1 Synset class

Synset is a class representing a common and shared meaning within the same language. *Synset* links synonyms forming a synonym set [8]. A *Synset* instance can link senses of different *Lexical Entry* instances with the same part of speech.

Example: In WordNet 2.1 [7], the synset "12100067" groups together the meanings of *oak* and *oak tree* that are considered as synonymous.

G.4.2 Synset Relation class

Synset Relation is a class representing the relationship between *Synset* instances.

G.4.3 Sense Relation class

Sense Relation is a class representing the relationship between *Senses* instances.

G.4.4 Sense Example class

Sense Example is a class used to illustrate the particular meaning of a Sense instance. A *Sense* can have zero to many examples.

Example: In a L*exical Entry* for the MWE *non-governmental organization (NGO)*, a S*ense Example* might be *Amnesty International*.

G.4.5 Semantic Definition class

Semantic Definition is a class representing a narrative description of a *Sense* or a *Synset*. It is displayed for human users to facilitate their understanding of the meaning of a L*exical Entry* and is not meant to be processable by computer programs. A Sense or a Synset can have zero to many definitions and can be expressed in a different language than the language of the L*exical Entry*.

Example: In a L*exical Entry* for *abbess* the narrative description may be *woman who is in charge of a convent*.

G.4.6 Statement class

Statement is a class representing a narrative description and refines or complements *Semantic Definition*. A definition can have zero to many *Statement* instances.

Example: In WordNet 2.1 [7], each synset contains a narrative description made of several parts. The first part is always a definition and the other parts are statements, often of heterogeneous content. For instance, the synset "12100067" for *oak* has the definition "a deciduous tree of the genus Quercus". The second part is a narrative describing the properties of the oak: "has acorns and lobed leaves". The third part is a proverb "great oaks grow from little acorns". Within the current International Standard, the first part may give a *Semantic Definition* instance and the two last parts may give two *Statement* instances.

G.4.7 Semantic Predicate class

Semantic Predicate is a class representing an abstract meaning together with its association with the *Semantic Argument* class. A S*emantic Predicate* instance may be used to represent the common meaning between different senses that are not necessarily fully synonymous.

These senses may be linked to *Lexical Entry* instances whose parts of speech are different. A *Semantic Predicate* instance pertains to a given *Lexicon* instance.

Example: In a *Lexical Entry* instance for *to buy* in the sense of "to get something by paying money for it", a *Semantic Predicate* instance might be defined with two semantic arguments: one for the person who buys and one for what is bought. Another *Lexical Entry* instance could be recorded for *buyer* and linked to the same predicate.

G.4.8 Predicative Representation class

Predicative Representation class is a class representing the link between the *Sense* and the S*emantic Predicate* classes.

Example: In the example given in the *Semantic Predicate* class section, the link between the sense of the verb (i.e. to *buy*) and the predicate might be marked as *master*. The link between the sense of the noun (i.e. *buyer*) and the predicate might be marked for instance as *agentiveNominalization*.

G.4.9 Semantic Argument class

Semantic Argument is a class representing an argument of a given Semantic Predicate.

Example: In the example given in the *Semantic Predicate* class section, the predicate might have two *Semantic Argument* instances: one for the person who buys and one for what is bought.

G.4.10 SynSemArgMap class

SynSemArgMap is a class representing the links between a semantic argument and a syntactic argument.

G.4.11 Predicate Relation class

Predicate Relation is a class representing the relationship between instances of *Semantic Predicate*.

G.4.12 Monolingual External Ref class

Monolingual External Ref is a class representing the relationship between a *Sense* or a *Synset* instance and an external system.

Annex H (informative) NLP semantic examples

H.1.1 Example of class adornment

Classes may be adorned with the following attributes:

Class name	Example of attributes	Comment
Sense	dating style frequency geography animacy	
Sense Relation	label	Sense Relation class is a multipurpose class that can be used to represent antonymy, generic/specific or part of relationship.
Sense Example	text source language	For instance a lexicon in the Bambara language (Bamanankan, bam), can contain examples expressed with standard orthography and examples with tones added in order to permit beginners to understand and pronounce the example.
Semantic Definition	text source language view	
Statement	label type text	
Semantic Predicate	label definition	
Predicative Representation	type comment	For instance, a semantic derivation between a sense of a noun and a sense of a verb can be linked to a shared predicate. In such a situation, the P*redicative Representation* of the sense of the noun can be typed as /*verbNominalization*/.
Semantic Argument	semanticRole restriction	
SynSemArgMap		
Predicate Relation	label type	

Class name	Example of attributes	Comment
Synset	label source	
Synset Relation	label type	
Monolingual External Ref	externalSystem externalReference	It is not the purpose of the semantic extension to provide a complex knowledge organization system, which ideally should be structured as one or several external systems designed specifically for that purpose. However, */externalSystem/* and */externalReference/* are provided to refer respectively to the name(s) of the external system and to the specific relevant node in this given external system.

H.1.2 Examples of word description

H.1.2.1 Simple example

The following English example presents two adjectives: *visible* and *invisible* that are considered to be monosemous lexical entries for the purpose of the explanation. These two words are linked at semantic level by means of a *Sense Relation* instance in order to represent that *visible* is the contrary of *invisible*.

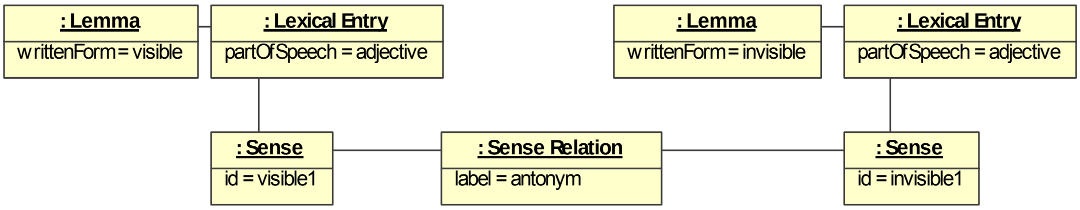


Figure H.1: instance diagram for a simple example

The same data can be expressed by the following XML fragment:

```
<LexicalEntry>
   <DC att="partOfSpeech" val="adjective"/>
   <Lemma>
      <DC att="writtenForm" val="visible"/>
   </Lemma>
   <Sense id="visible1">
      <SenseRelation targets="invisible1">
         <DC att="label" val="antonym"/>
      </SenseRelation>
   </Sense>
</LexicalEntry>
<LexicalEntry>
   <DC att="partOfSpeech" val="adjective"/>
   <Lemma>
```

```
        <DC att="writtenForm" val="invisible"/>
    </Lemma>
    <Sense id="invisible1"/>
</LexicalEntry>
```

H.1.2.2 Example from Princeton WordNet 2.1

The following English example focuses on *Synset* instances. This example is taken from WordNet version 2.1 [7] and presents two *Synset* instances: one for *oak* as a tree and one for *oak* as the wood of the tree. Each WordNet's lex_id is used to identify a *Sense* instance. Each gloss is split into a *Semantic Definition* instance and possibly several *Statement* instances. The two *Synset* instances are linked by a *Synset Relation* instance that is marked as *substanceHolonym*.

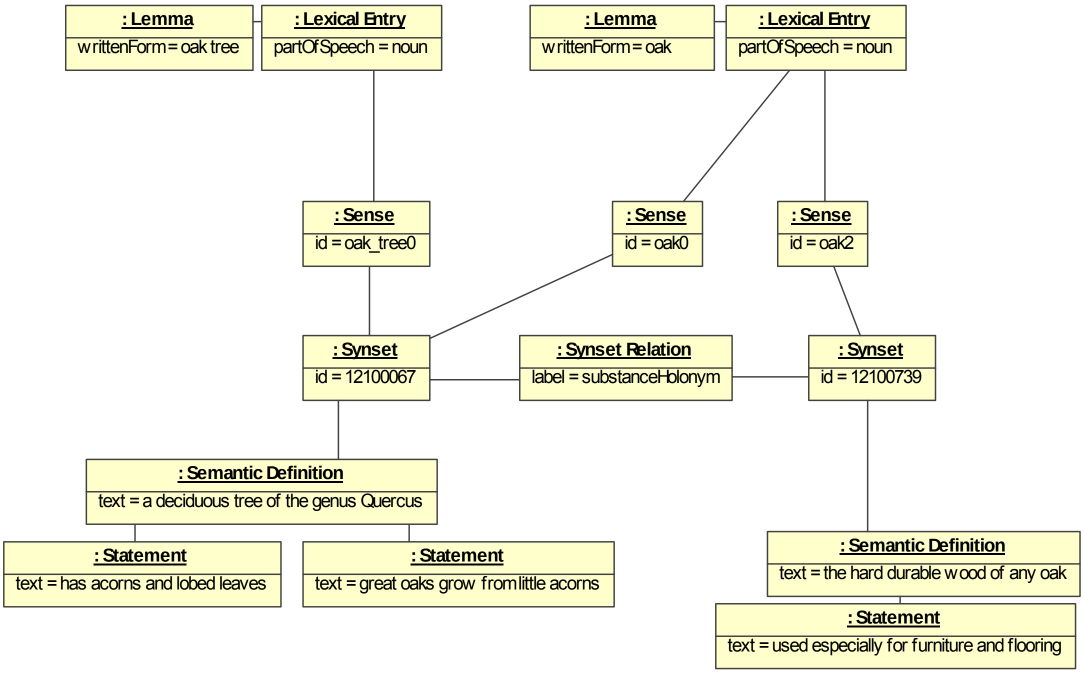


Figure H.2: Instance diagram for the example taken from WordNet

The same data can be expressed by the following XML fragment:

```
<LexicalEntry>
    <DC att="partOfSpeech" val="noun"/>
    <Lemma>
        <DC att="writtenForm" val="oak tree"/>
    </Lemma>
    <Sense id="oak_tree0" synset="12100067"/>
</LexicalEntry>
<LexicalEntry>
    <DC att="partOfSpeech" val="noun"/>
    <Lemma>
        <DC att=writtenForm" val="oak"/>
    </Lemma>
    <Sense id="oak0" synset="12100067"/>
    <Sense id="oak2" synset="12100739"/>
</LexicalEntry>
<Synset id="12100067">
```

```
    <SemanticDefinition>
        <DC att="text" val="a deciduous tree of the genus Quercus"/>
        <Statement>
            <DC att="text" val="has acorns and lobed leaves"/>
        </Statement>
        <Statement>
            <DC att="text" val="great oaks grow from little acorns"/>
        </Statement>
    </SemanticDefinition>
    <SynsetRelation targets="12100739"
        <DC att="label" val="substanceHolonym"/>
    </SynsetRelation>
</Synset>
<Synset id="12100739">
    <SemanticDefinition>
        <DC att="text" val="the hard durable wood of any oak"/>
        <Statement>
            <DC att="text" val="used especially for furniture and flooring"/>
        </Statement>
    </SemanticDefinition>
</Synset>
```

H.1.2.3 Example from *Dictionnaire Explicatif et Combinatoire*

The following French example focuses on *Semantic Predicate* instances and connection between syntactic and semantic representations. This example presents the syntax of the sense *Aider1* taken from *Dictionnaire Explicatif et Combinatoire* [4] [21]. “Aider1” is linked to the semantic argument: “X aide Y à Z-er par W” as in “il vous aidera par son intervention à surmonter cette épreuve”. (Literally: X helps Y to Z in order to W", "he will help you with his intervention in order to overcome this ordeal", i.e., he will intervene to help you overcome this ordeal.") This *Lexical Entry* instance yields eight different *Subcategorization Frame instances*, but figure H.2 supplies the representation for only the first two examples: "La Grande-Bretagne aide ses voisins" ("Great Britain helps its neighbors) and "La Grande-Bretagne a aidé à créer l'ONU" ("Great Britain helped create the UN"), with a special focus on links between syntactic and semantic representations. The two *Subcategorization Frame* instances are related to a common semantic predicate, which has its semantic arguments (X, Y, Z and W). They are shown to be related to particular S*yntactic Argument* instances in the different constructions of the verb. That is, the *Subcategorizations Frame* instances are not linked directly to the predicate, but a particular S*yntactic Argument* in each *Subcategorization Frame* instance is linked to a particular *Semantic Argument* instance.

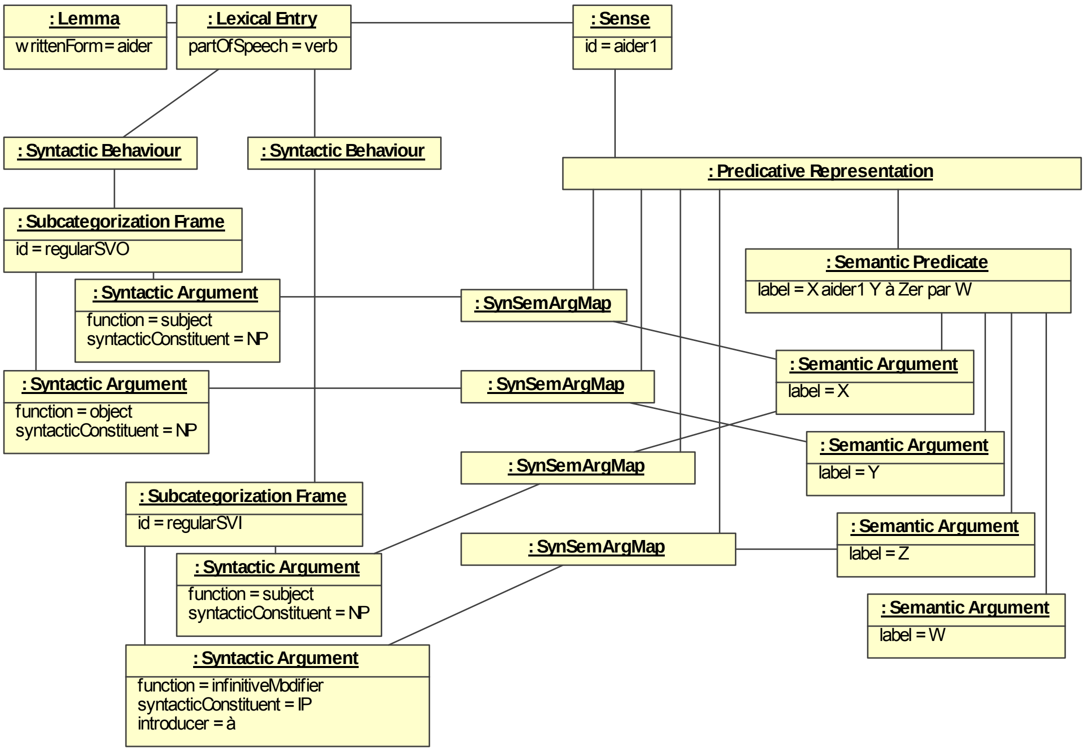


Figure H.3: Instance diagram for the example taken from the DEC

The same data can be expressed by the following XML fragment:

```
<LexicalEntry>
   <DC att="partOfSpeech" val="noun"/>
   <Lemma>
      <DC att="writtenForm" val="aider"/>
   </Lemma>
   <Sense id="aider1">
      <PredicativeRepresentation predicate="P1">
         <SynSemArgMap synArg="SA1" semArg="ARX"/>
         <SynSemArgMap synArg="SA2" semArg="ARY"/>
         <SynSemArgMap synArg="SA3" semArg="ARX"/>
         <SynSemArgMap synArg="SA4" semArg="ARZ"/>
      </PredicativeRepresentation>
   </Sense>
   <SyntacticBehaviour subcategorizationFrames="regularSVO"/>
   <SyntacticBehaviour subcategorizationFrames="regularSVI"/>
</LexicalEntry>
<SubcategorizationFrame id="regularSVO">
   <SyntacticArgument id="SA1">
      <DC att="function" val="subject"/>
      <DC att="syntacticConstituent" val="NP"/>
   </SyntacticArgument>
   < SyntacticArgument id="SA2">
      <DC att="function" val="object"/>
      <DC att="syntacticConstituent" val="NP"/>
   </SyntacticArgument>
</SubcategorizationFrame>
<SubcategorizationFrame id="regularSVI">
   <SyntacticArgument id="SA3">
      <DC att="function" val="subject"/>
```

```
        <DC att="syntacticConstituent" val="NP"/>
    </SyntacticArgument>
    </SyntacticArgument id="SA4">
        <DC att="function" val="infinitiveModifier"/>
        <DC att="syntacticConstituent" val="IP"/>
        <DC att="introducer" val="à"/>
    </SyntacticArgument>
</SubcategorizationFrame>
<SemanticPredicate id="P1">
    <DC att="label" val="X aider1 Y à Z par W"/>
    <SemanticArgument id="ARX">
        <DC att="label" val="X"/>
    </SemanticArgument>
    <SemanticArgument id="ARY">
        <DC att="label" val="Y"/>
    </SemanticArgument>
    <SemanticArgument id="ARZ">
        <DC att="label" val="Z"/>
    </SemanticArgument>
    <SemanticArgument id="ARW">
        <DC att="label" val="W"/>
    </SemanticArgument>
</SemanticPredicate>
```

Annex I (normative) NLP multilingual notations extension

I.1 Objectives

The purpose of this section is to describe the representation of equivalents for Sense or SyntacticBehaviour instances between or among two or more languages.

I.2 Absence versus presence of multilingual notations in a lexicon

The multilingual model can be used for lexical databases describing two or more languages (i.e., bilingual vs. multilingual resources). There is no need to use the multilingual notations in a monolingual database.

I.3 Class diagram

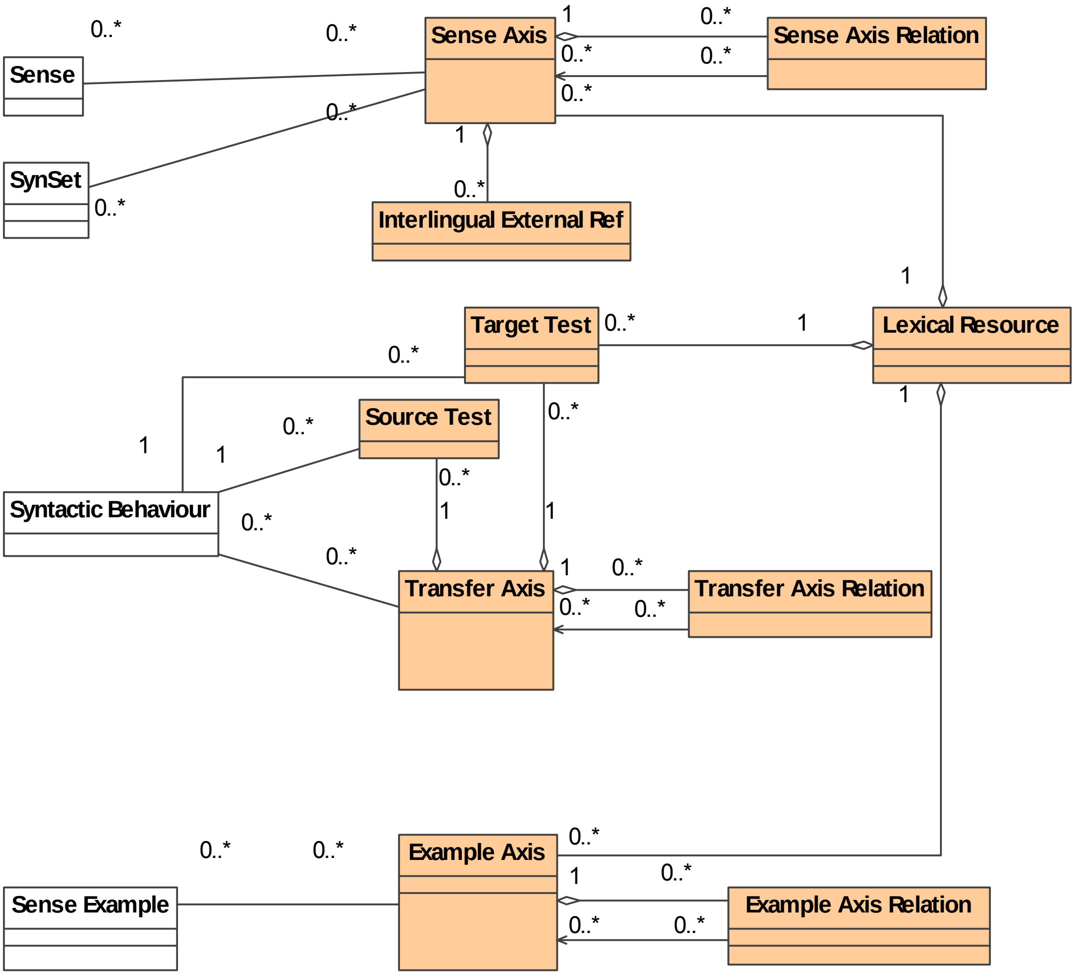


Figure I.1: Multilingual notations model

I.4 Options

The simplest configuration is the bilingual lexicon where a single link is used to represent the equivalent of a given sense from one language into another, but actual practice reveals at least five more complex configurations:

I.4.1 diversification and neutralization

In certain circumstances, simple one-to-one mapping between apparent equivalents in two or more languages does not work very well because the conceptual scope represented by words and expressions in the different languages is frequently not the same.

Example: German *Lack* covers a wide range of concepts expressed in English with very specific words: *paint, lacquer, varnish, shellac*, etc. In this case the German to English direction involves diversification and the English to German direction involves neutralization.

I.4.2 number of links

Although the strategy of one-to-one equivalence is viable for two languages, it becomes untenable for a more extensive number of languages because the number of links explodes to unmanageable proportions. Furthermore, it cannot necessarily be assumed that if under certain conditions, sense L_1-A equals L_2-A, and L_1-A = L_3-A, that L_2-A = L_3-A, despite the fact that common logic would imply that this is the case. The larger the number languages and the number of links, the greater the chance that lateral links between the various languages can prove faulty.

I.4.3 transfer or interlingual pivot

NLP-oriented translation is based on two approaches: the use of an *interlingual pivot*, which operates on the basis of semantic analysis and *transfer*, which is based on machine parsing of source text syntax. The pivot approach is implemented via the *SenseAxis* class, and the transfer approach via the *TransferAxis* class.

I.4.4 representation of similar languages

Very closely related languages that share significant patterns can be efficiently represented using shared *Sense Axis* instances (resp. *Transfer Axis* instances*)*, together with a limited number of specific *Sense Axis* instances (resp. *Transfer Axis* instances) for representing variations between the languages.

I.4.5 direction and tests

Some multilingual lexicons are very declarative in that every translation is represented by an interlingual object. Others are very procedural in that they restrict the translation by logical tests applied at the source or target language levels.

I.5 Description of multilingual notations model

The model is based on the notion of Axes that link *Sense, Syntactic Behaviour* and *Sense Example* instances pertaining to different languages. Lexicon designers can freely structure the various axes directly or indirectly between and among different languages. A direct link is implemented by a single axis. An indirect link is implemented using several axes and one or more relations.

The model is based on three main classes:

- Sense Axis
- Transfer Axis
- Example Axis

I.5.1 Sense Axis class

S*ense Axis* is a class representing the relationship between different closely related senses in different languages and implements approach based on the interlingual pivot. The purpose is to describe the translation of words from one language to another. Optionally, a *Sense Axis* may refer to an external knowledge representation system where the appropriate equivalent can be found.

I.5.2 Sense Axis Relation class

Sense Axis Relation is a class representing the relationship between two different *Sense Axis* instances.

I.5.3 Interlingual External Ref class

Interlingual External Ref is a class representing the relationship between a *Sense Axis* instance and an external system.

I.5.4 Transfer Axis class

Transfer Axis is a class representing multilingual transfer. A *Transfer Axis* instance links several *Syntactic Behaviour* instances pertaining to different languages.

I.5.5 Transfer Axis Relation class

Transfer Axis Relation is a class representing the relationship between two *Transfer Axis* instances.

I.5.6 Source Test class

Source Test is a class representing a condition that affects the translation with respect to the usage on the source language side.

I.5.7 Target Test class

Target Test is a class representing a condition that affects the translation with respect to the usage on the target language side.

I.5.8 Example Axis class

Example Axis is a class representing previously translated translation examples that meet matching or fuzzy matching criteria for a given text chunk.

I.5.9 Example Axis Relation class

Example Axis Relation is a class representing the relationship between two *Example Axis* instances.

Annex J (informative) NLP multilingual notations examples

J.1 Example of class adornment

Classes may be adorned with the following attributes:

Class name	Example of attributes	Comment
Sense Axis	label id	A single word in one language can have as its equivalent a compound in another language.
Sense Axis Relation	label view	The label enables the coding of simple interlingual relations like the specialization of *fleuve* compared to *rivière* and *river*. It is not, however, the goal of this strategy to code a complex knowledge organization system.
Interlingual External Ref	externalSystem externalReference	It is not the purpose of the multilingual extension to provide a complex knowledge organization system, which ideally should be structured as one or several external systems designed specifically for that purpose. However, */externalSystem/* and */externalReference/* are provided to refer respectively to the name(s) of the external system and to the specific relevant node in this given external system.
Transfer Axis	label id	This approach enables the translation of syntactic arguments involving inversion, such as: fra:“elle me manque” => eng:“I miss her”. Due to the fact that a L*exicalEntry* can contain as its form a support verb, it is possible to represent equivalents between a simple verb in one language and a more complex verb structure in another language, involving, e.g. a support verb or other supplemental elements, such as in the equivalence relation between French: “Marie rêve” and Japanese: "Marie wa yume wo miru".
Transfer Axis Relation	label variation	The *Transfer Axis Relation* class may be used to represent slight variations between closely related languages. For instance, in order to represent slight variations between European and Brazilian Portuguese, different intermediate *Transfer Axis* instances can be created. The *Transfer Axis Relation* class

Class name	Example of attributes	Comment
		holds a label to distinguish which of the *Transfer Axis* instances to use depending on the object language.
Source Test	text comment	
Target Test	text comment	
Example Axis	comment source id	The purpose of this class is not to record large scale multilingual corpora; the goal is to link a *Lexical Entry* instance with a typical sample translation.

J.2 Examples of word description

J.2.1 Example of equivalence at sense level

The example shown in figure J.1 illustrates how to use two intermediate *Sense Axis* instances in order represent a near match between *fleuve* in French and *river* in English. This phenomenon is usually called diversification and neutralization. The *Sense Axis* instance on the top is not linked directly to any English sense because this notion does not exist in English.

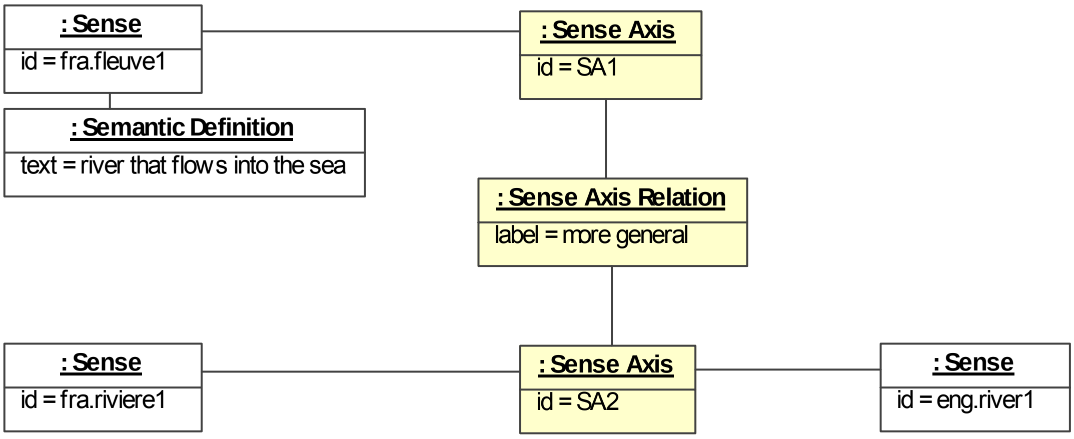


Figure J.1: Instance diagram for "river"

The instances modeled in the multilingual notation annex can be expressed by the following XML fragment, with the assumption that the *Sense* and *Semantic Definition* instances are defined elsewhere:

```
<SenseAxis id="SA1" senses="fra.fleuve1">
   <SenseAxisRelation targets="SA2">
      <DC att="label" val="more general"/>
   </SenseAxisRelation>
</SenseAxis>
<SenseAxis id="SA2" senses="fra.riviere1 eng.river1"/>
```

J.2.2 Example of equivalence at transfer level

This example shows how to use the *Transfer Axis Relation* instance to relatate different information in a multilingual transfer lexicon. It represents the translation of the English *develop* into Italian and Spanish. Recall that the more general sense links *develop* in English and *desarrollar* in Spanish. Both Spanish and Italian have restrictions that should be tested in the source language: if the second argument of the *Subcategorization Frame* instance refers to certain elements like building, it should be translated into specific verbs.

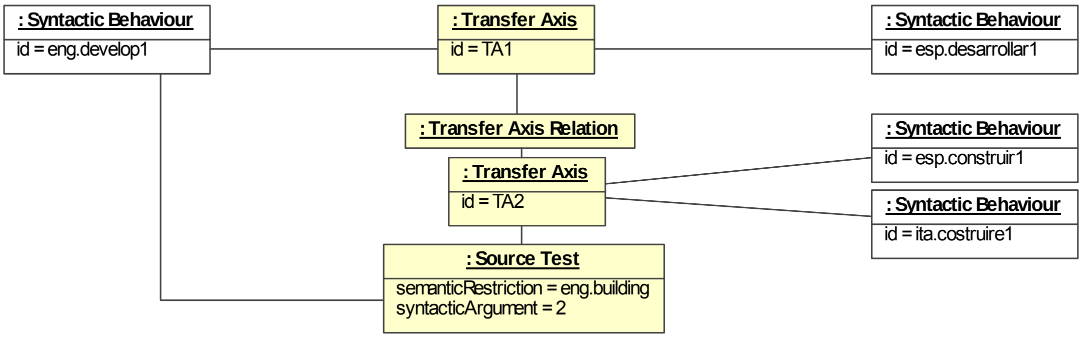


Figure J.2: Instance diagram for "develop"

The instances modeled in the multilingual notation annex can be expressed by the following XML fragment, with the assumption that the *Syntactic Behaviour* instances are defined elsewhere:

```
<TransferAxis id="TA1" synBehaviours="eng.develop1 esp.desarrollar1">
   <TransferAxisRelation targets="TA2"/>
</TransferAxis>
<TransferAxis id="TA2" synBehaviours="esp.construir1 ita.costruir1">
   <SourceTest>
      <DC att="semanticRestriction" att="eng.building"/>
      <DC att="syntacticArgument" att="2"/>
   </SourceTest>
</TransferAxis>
```

Annex K (normative) NLP paradigm class extension

K.1 Objectives

The objective of this extension is to provide the description in intension of the morphology of a given language. The aim is to support the organization and storage of lexical information needed for the analysis and generation of inflected, agglutinated, derived, or compound word forms. These forms are not explicitly listed but the *Lexical Entry* instance is associated with a shared *Paradigm Class* instance. The forms documented in the lexical entry may include the root, stem, or stem allomorphs. And these forms are unique to a specific lexical entry.

The lexical information documented in the paradigm structure may include shared forms (e.g. affixes) and associated rules intended to support the design of morphological lexicons that are process independent. That is, algorithms used to analyze and generate the forms. This extension is not intended to meet all the needs for morphological lexicons; however, LMF core package and this extension provide the basis for developing additional morphology extensions.

K.2 Class diagram

The following diagram specifies the classes of the Paradigm Class model:

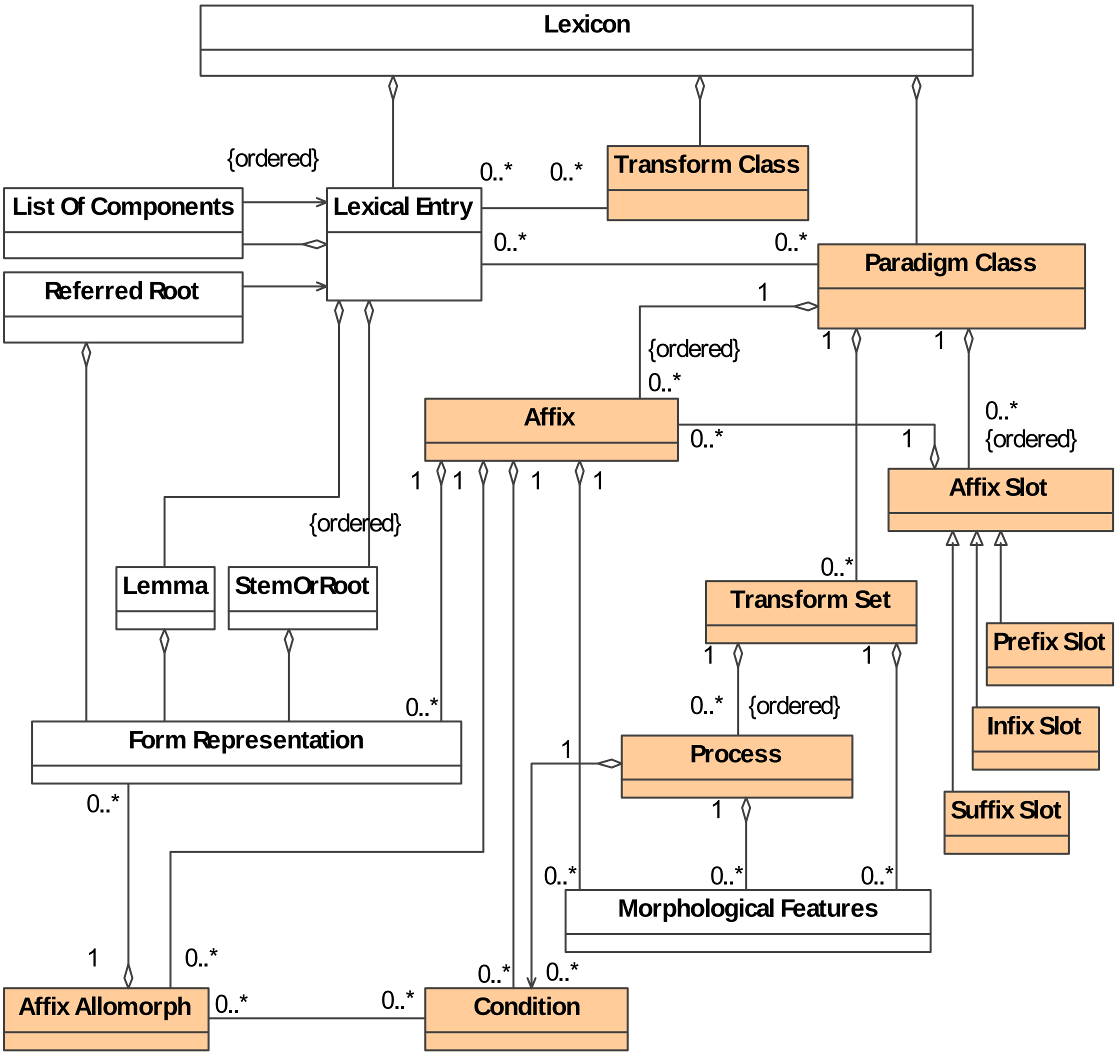


Figure K.1: Paradigm class model

K.3 Description of the paradigm class model

K.3.1 Introduction

Lexical Entry and *Paradigm Class* are in aggregate association with the *Lexicon* class. The *Lexical Entry* class manages the word forms and morphemes that are unique to a specific lexical entry. In contrast, the *Paradigm Class* manages the classes that constitute a paradigm schema shared by several lexical entries.

K.3.2 Paradigm Class class

Paradigm Class is a class representing a paradigm pattern for the morphology of a given language. The *Paradigm Class* is constrained on the part of speech. The Paradigm class may be subtyped, e.g. paradigmType='inflectional'. A *Paradigm Class* can manage an *Affix* class indirectly through the *Affix Slot* class, or can manage the *Affix* class directly, but not both. The

Paradigm Class also manages a *Transform Set* class that supports the modelling of linguistic rules and processes. The *Transform Set* class association and *Affix Slot/Affix* association are not mutually exclusive.

K.3.3 Transform Set class

Transform Set is a class representing the association between *Process* class and *Morphological Features* class that further define the scope or range of the managed paradigm. *Transform Set* is in a zero to many aggregation with the *Paradigm Class*.

K.3.4 Process class

Process is a class representing the rules or linguistic operations applied to one word form or combination of word forms. A *Process* instance can be subtyped, e.g. processType='phonologicalOperation' and is in ordered aggregation with the *Transform Set* Class.

K.3.5 Condition class

Condition is a class representing the conditions determining the usage of a *Process* or an *Affix Allomorph* instance.

K.3.6 Affix class

Affix is a class representing an affix, that is a word form or morpheme that is required for analyzing of generating word forms. An *Affix* instance may be refined by several *Form Representation* instances in situations where multiple orthographies are required.

K.3.7 Affix Slot class and Affix Slot subclasses

Affix Slot is a class representing the *Affix* position in the word. There is a zero to many aggregation between *Affix* class and *Affix Slot* class. The *Affix Slot* class is in an ordered aggregation with the *Paradigm Class* and has the following subclasses: *Prefix Slot*, *Infix Slot*, and *Suffix Slot*.

Note: The direction of the ordered constraint is dependent on the subclass and language. Right-to-left and left-to-right languages would have different orders; also, suffix slots and prefix slots would have different orders.

K.3.8 Affix Allomorph class

Affix Allomorph is a class representing allomorphs of the canonical affix form in all scripts and representations. An *Affix Allomorph* may be refined by several *Form Representation* instances in situations where multiple orthographies are required. An *Affix Allomorph* may be associated to *Condition* instances in order to constraint the application of such or such *Affix Allomorph* instance.

K.3.9 Transform Class class

Transform Class is a class representing an additional mark to the *Lexical Entry* instance, in order to be combined with a *Paradigm Class* instance.

Annex L (informative) NLP paradigm class examples

L.1.1 Absence versus presence of inflectional paradigm classes in a lexicon

Not all lexicons utilize the paradigm approach. Theoretically, it would be possible to list all forms in a lexicon, but the use of paradigm classes has the following important advantages:

- Description of languages with complex morphology is possible without resorting to voluminous, unmanageable documentation.

- The linguistic knowledge describing how to associate a lemma with an inflected, agglutinated, compound, derived form is focused on a specific exemplary element instead of being spread throughout a great number of elements.

L.1.2 Example of class adornment

Classes may be adorned with the following attributes:

Class name	Example of attributes	Comment
Paradigm Class	id comment example partOfSpeech paradigmType	A *Paradigm Class* instance is designed to be shared and referred, thus it holds an identifier. A *Paradigm Class* instance cannot be used for two different parts of speech (section K.3.2), so it's important to record the part of speech mark.
Transform Set	comment	This class is designed to link instances, thus, aside from a comment, the class is not intended to be marked with any linguistic information.
Process	operator affixRank componentRank stemRank rule stringValue	The values for /operator/ may be for instance, /addLemma/, /addAffix/, or /addComponentStem/. The values for rules are string values that can represent a wide range of linguistic rules, for instance, a pattern such as /CVx/ or a formalism such as /[X]n -> [1 *ut*]v/.
Condition	id localization agreement affix transformType	
Affix	writtenForm type	The type may be specified for instance with values like /prefix/ or /suffix/.
Affix Slot and subclasses	label position	The /position/ specifies where the affix is to be set in the word form.

Class name	Example of attributes	Comment
Affix Allomorph	writtenForm	
Transform Class	id comment	A *Transform Class* instance is designed to be shared and referred, thus it holds an identifier.

L.1.3 Examples of word description

L.1.3.1 Introduction

The model allows the development of implementation to support different modeling goals, e.g. Item-and-Arrangement model, Item-and-Process model, lemma based approach [11], [12], [13], [14], [15], [17].

The examples are presented in the following order:

- examples of inflection, beginning with simple phenomena and ending with more complex ones
- examples of agglutination
- example of word creation
- example of derivation
- examples of composition

StemOrRoot, *Affix*, *Affix Slot*, *Lexical Entry* (in *List Of Components* association) and *Process* instances are ordered. In the following diagrams, the order is not mentioned. The assumption is made that the instances are to be read from left to right and top to bottom.

L.1.3.2 Inflection using an underspecified paradigm class instance

In this example, the lemma *clergyman* is declared as conforming to the *Paradigm Class* “asMan”. This *Paradigm Class* instance has a name but is not analytically described within the lexicon. For instance, the *Paradigm Class* may be implemented by an external opaque algorithm.

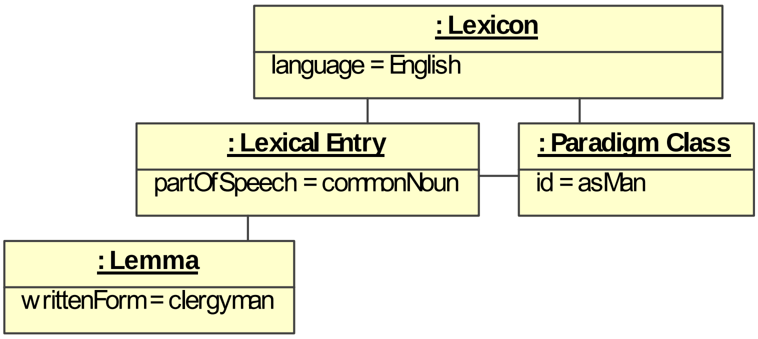


Figure L.1 inflection using an underspecified paradigm class

L.1.3.3 Inflection using the Transform Set Class

This example implements a traditional lemma based inflection using the English "table". This word is considered as inflected according to the paradigm class "regularNoun". The strategy used in this example is based on the lemma. The singular form is set as the lemma, that is "table". The plural form is set as the lemma plus the affix "s". In the diagram, there is only one affix, but more complex situations may contain more than one affix, thus, in order to adopt a generic strategy, these affixes are numbered. In the example, the affix number is one.

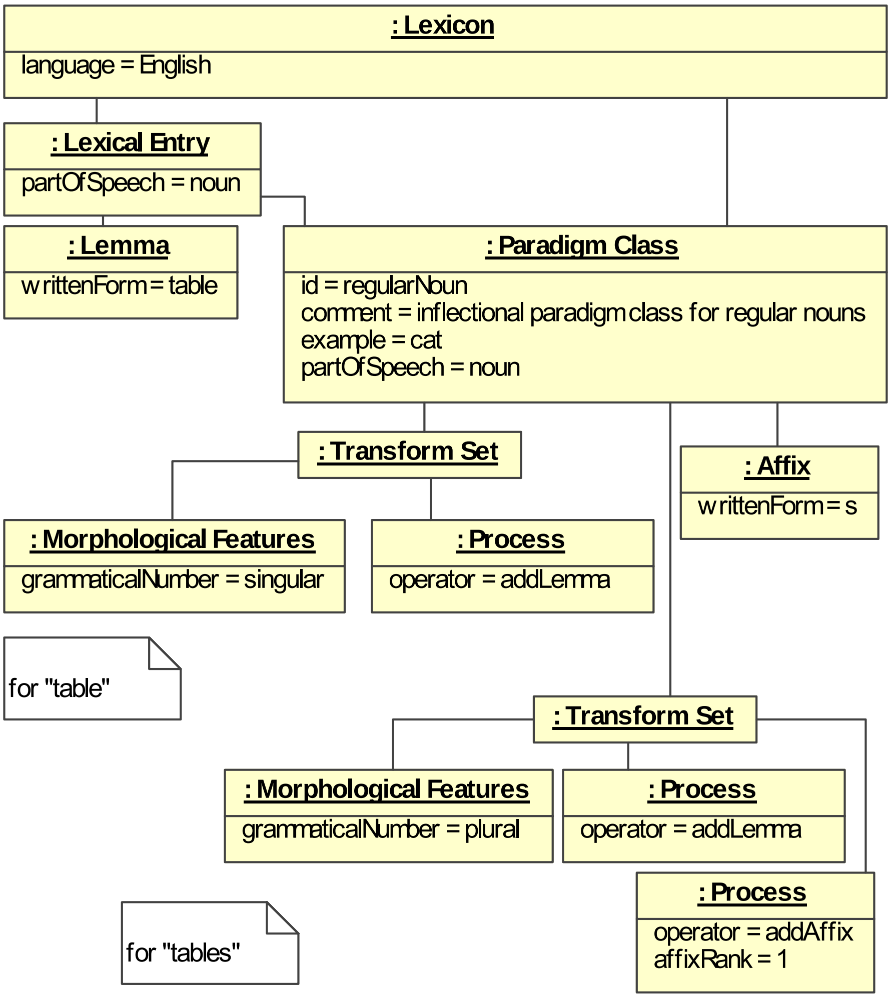


Figure L.2: Inflection using Transform Set class

L.1.3.4 Inflection based on an item and arrangement approach

This example implements an item and arrangement approach. On the lexical entry side, stems are represented and associated with a *Morphological Features* instance. On the Paradigm Class side, affixes are associated with *Morphological Features* instances. Stem and affix combination are correlated through the morphological features. Thus, the first and third person will be associated to "pesqu" + "e" to give "pesque". And the second person will be associated to "pesqu" + "es" to give "pesques".

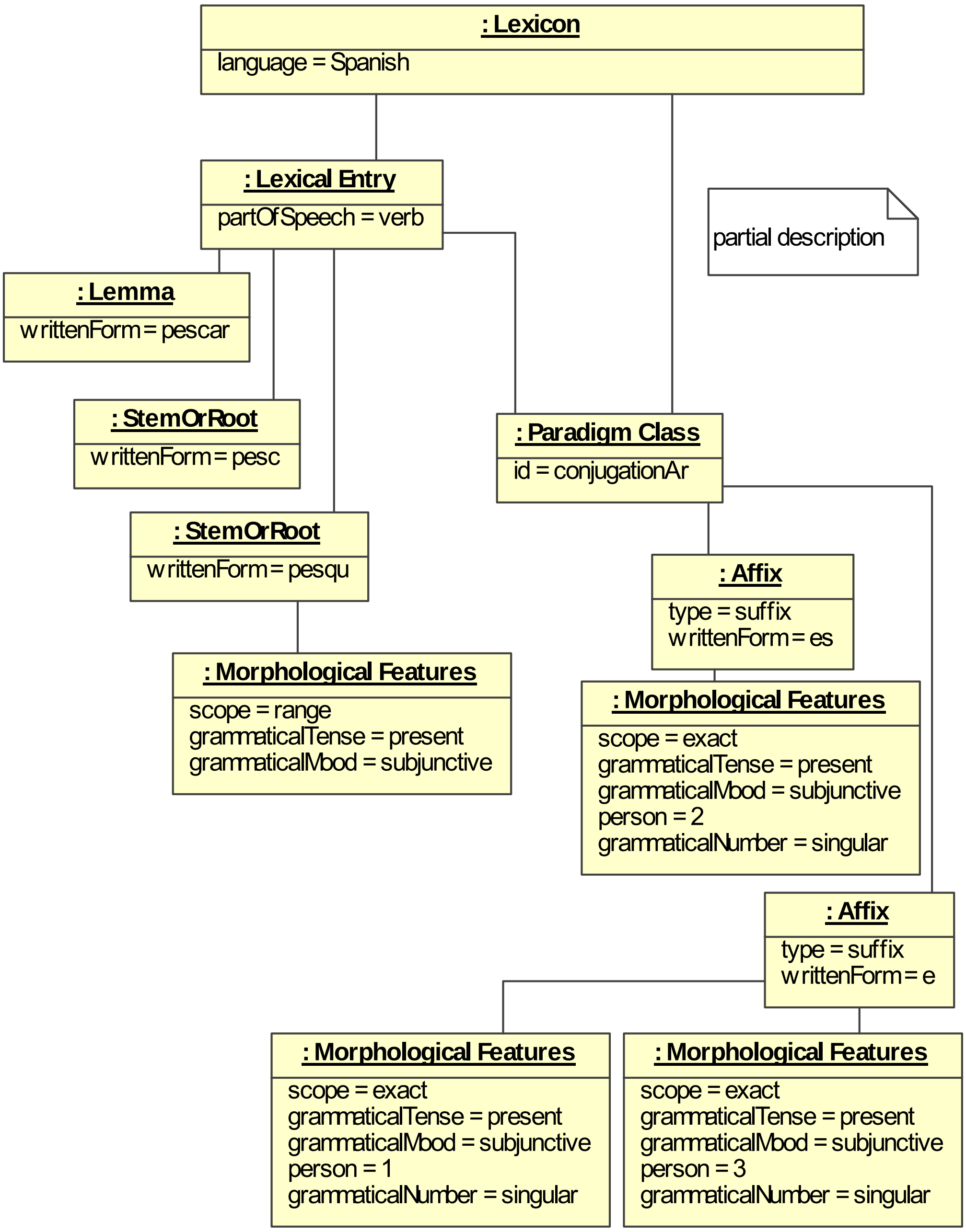


Figure L.3: Inflection using item and arrangement approach

L.1.3.5 Complex inflection: verbal forms in Tagalog

Verbs in Tagalog are difficult to describe by the means of a pure item and arrangement strategy. This particular example shows how to form the future tense by taking the first consonent, adding the first vowel and adding these letters to the left side of the lemma.

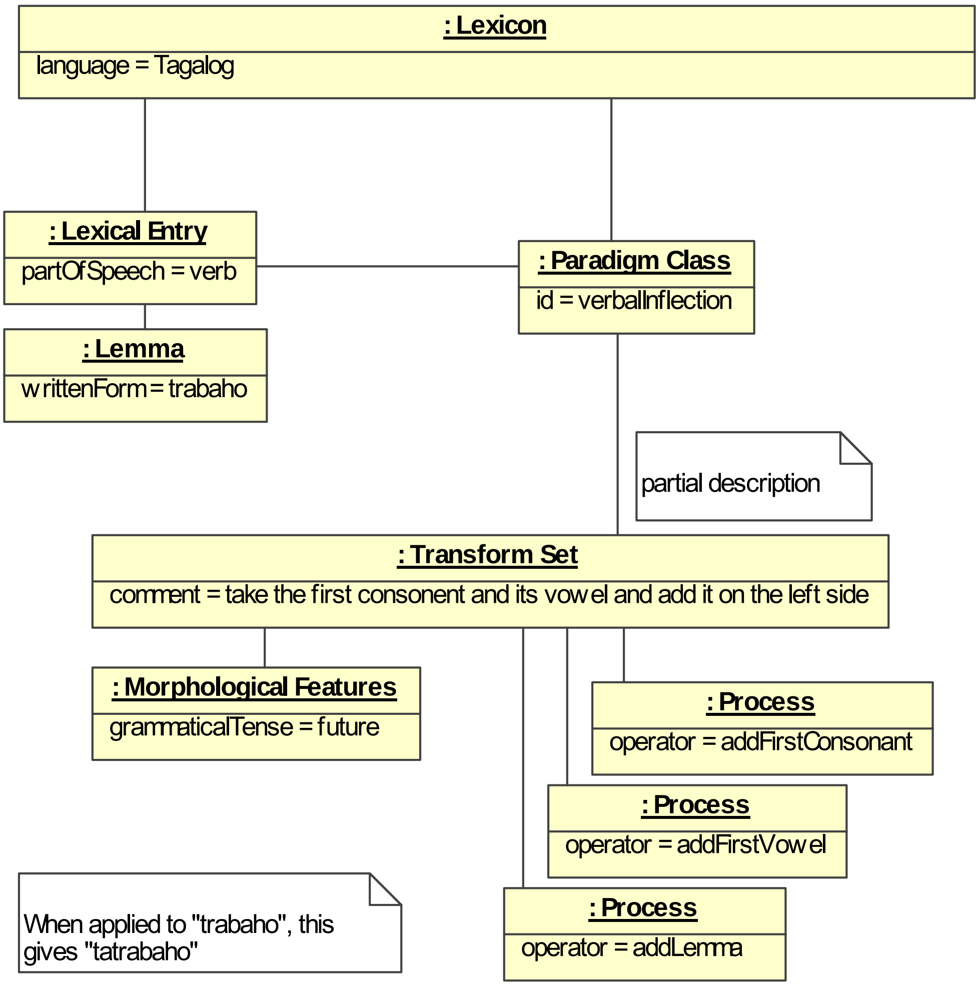


Figure L.4: verbal forms in Tagalog

L.1.3.6 Lemma based agglutination

This example implements a lemma based strategy with a multiple underlying form (MUF) model for the allomorphs. The examples shown in the previous sections deal with inflectional languages. On the contrary, the following example is about Hungarian that is an agglutinative language. In Hungarian, agglutination is ruled by two interrelated mechanisms that are repeated many times: a suffixation mechanism, where a stem is associated with a suffix, and a vowel harmony mechanism that selects an affix depending on vowel agreement.

For instance: "ház" (house) gives "ház+ak" (houses) because of "á", but "szék" (chair) gives "szék+ek" (chairs) because of "é". The system is rather general but cannot be associated with the whole lexicon because there are exceptions. And these exceptions must be recorded in *Paradigm Class* instances. For example, imported words that have variants like "hotelban" vs "hotelben" (at the hotel) do not respect the general rule.

Vowel harmony is represented by a *Condition* instance associated with an *Affix Allomorph* instance.

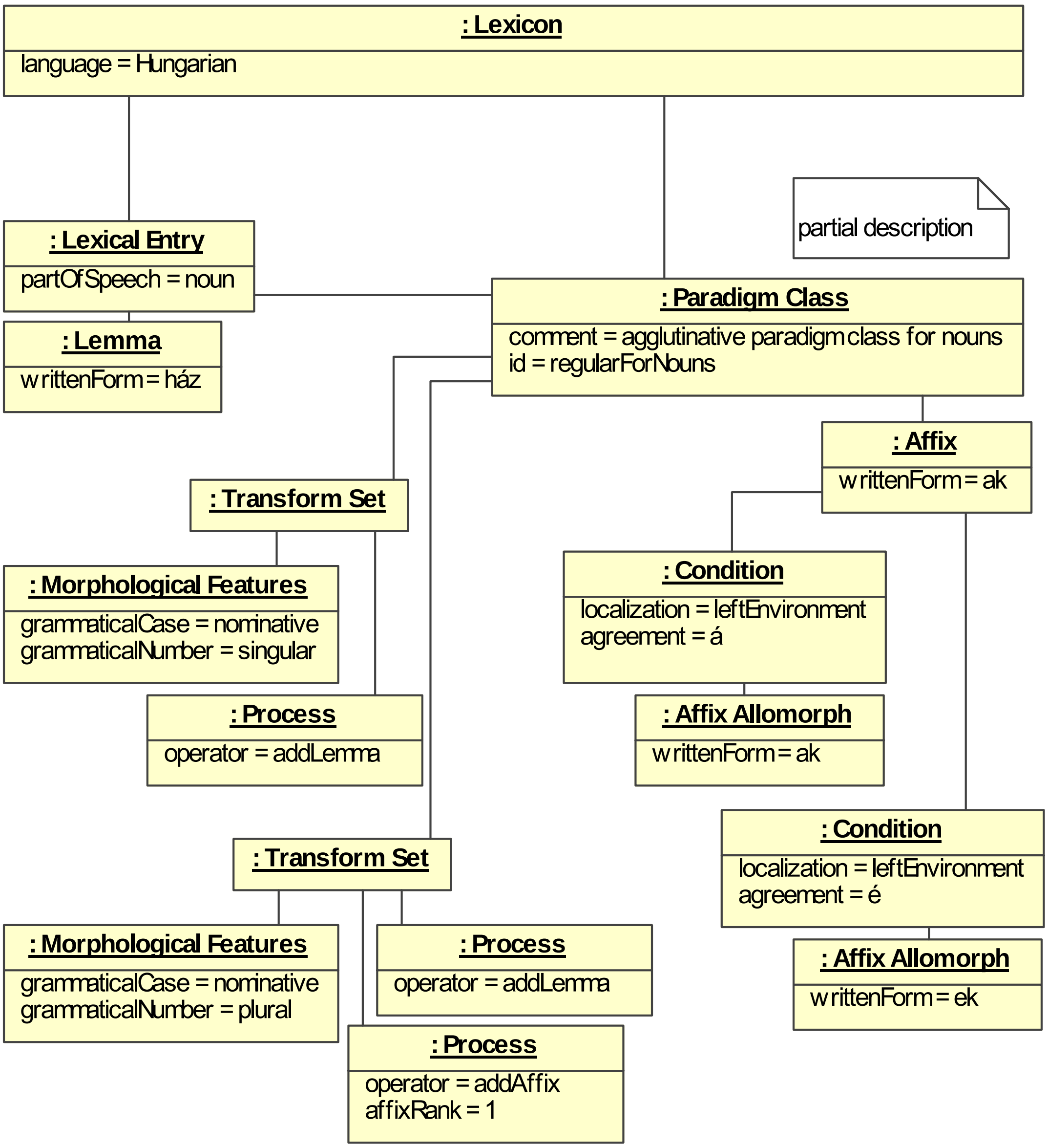


Figure L.5: Lemma based agglutination

Let's note that due to the fact that in Hungarian the number of forms for a noun, for instance, is more than two hundred, the strategy of listing all the agglutinated forms explicity in the lexicon (like in Annex A) produces unmanageable documentation. Usually, a strategy based on paradigm classes is preferred.

L.1.3.7 Agglutination using the Suffix Slot class

This example implements an item and arrangement approach. The following diagram shows a simplified Turkish verb conjugation. To support the agglutinative process, each *Suffix Slot* instance represents a different verbal aspect. This example shows the suffix for the past tense, using the Affix Slot to indicate a tense affix and a *Morphological Features* instance to indicate past tense. This model differs significantly from the model for Spanish conjugation (as presented previously) where tense and person are both managed through a *Morphological Feature* instance.

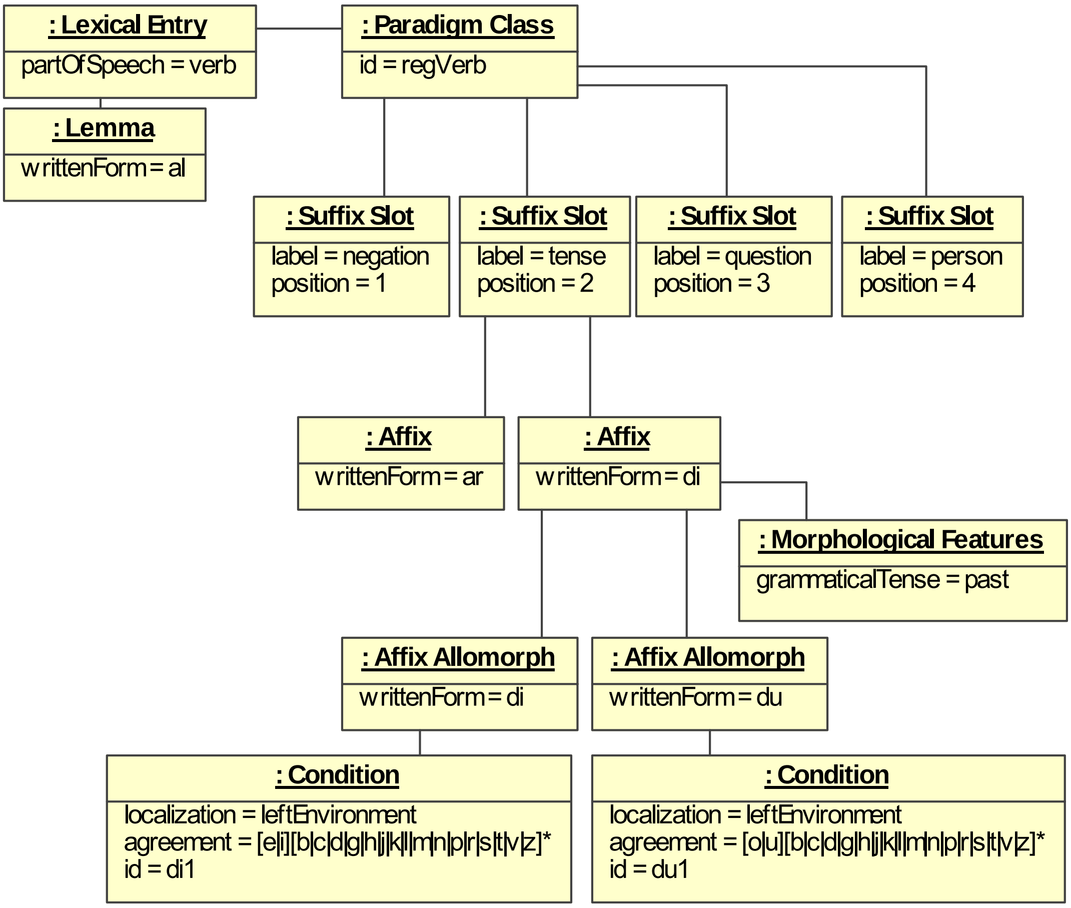


Figure L.6: Agglutination using the Affix slots class

L.1.3.8 Word creation: an Arabic dialect scheme

This example implements an Item and Process approach applied to an Arabic scheme and paradigm class description. In the following diagram, 'C' stands for consonant and 'V' stands for vowel. The *Process* class is used to implement rules that may have conditions. The first rule states that "a" is added to the stem for third person singular perfect. The second rule covers phonological changes under certain conditions: if the transform type is C2=C3 and the affix consists of a consonant-vowel-anything combination, then the stem is used. Otherwise, the format is consonant-vowel-consonant-consonant. This approach represents stems, rules and conditions within the lexicon but an external parser is needed to fully interpret the rules.

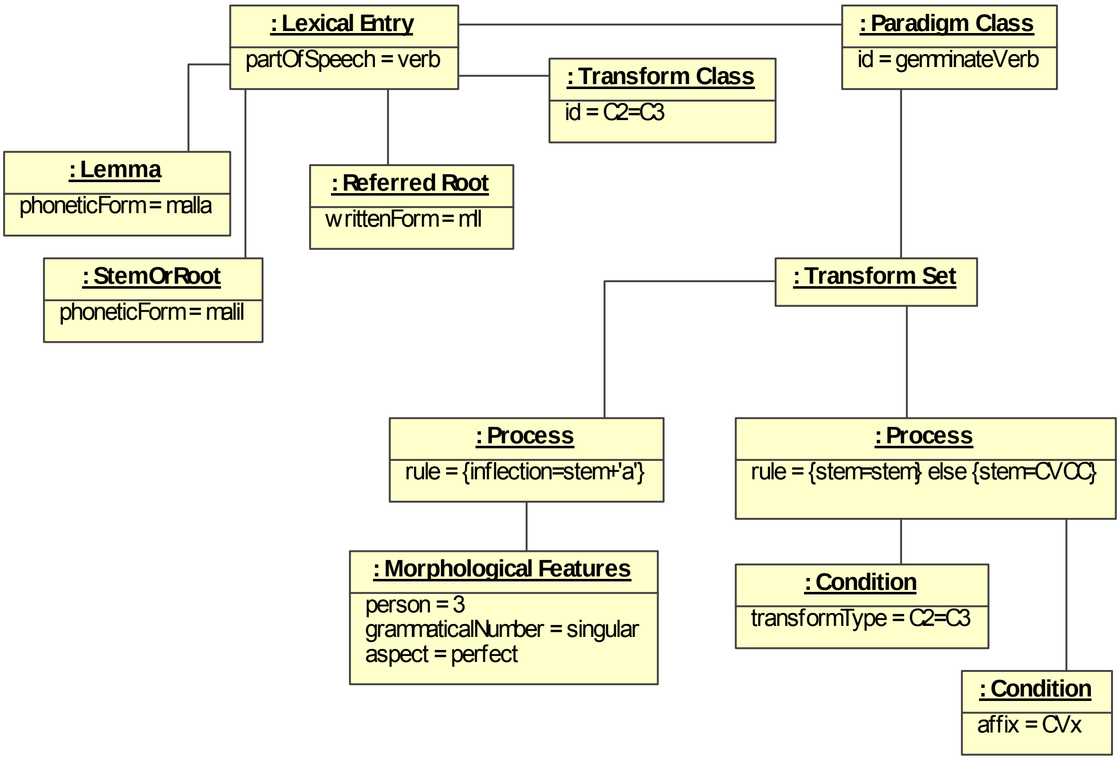


Figure L.7 Arabic dialect scheme

L.1.3.9 Derivation using duplication

In Thai, the derivation is a frequent mechanism, thus it is time consuming to record a separate entry for the derived form. On the contrary, a derivation may be associated with one entry. In the following Thai example, the derivation for intensification is built by duplication. The adjective "dam" (black) is the source of a derivation that produces "damdam" (blackish).

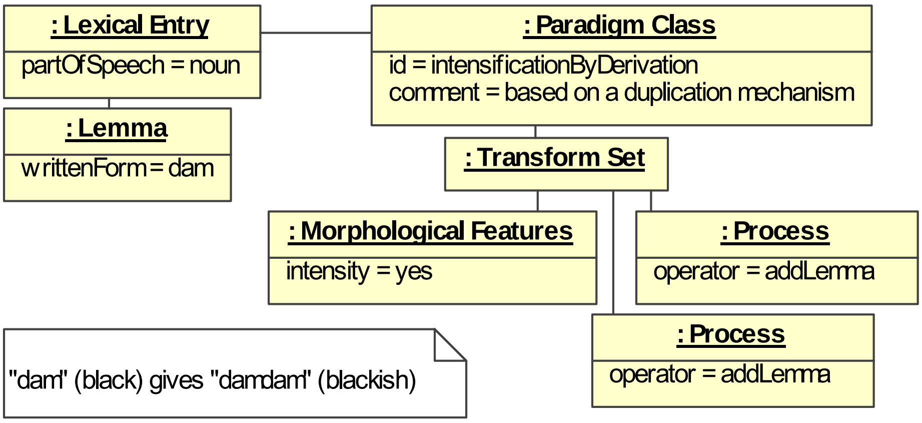


Figure L.8 Derivation using duplication

It's worth noting that duplication is not specific to derivation but appears also in languages like Indonesian in order to express plural forms [23].

L.1.3.10 Lemma based composition: Anwendungsprogramm

Composition is a bit different from inflection and agglutination. In these latest examples, each form is defined from the lemma (or one stem) of the entry with various operations like adding affixes. In the composition process, each form is defined from the *List of Component* instance and thus relies on the morphological behaviour of each component.

The following diagram illustrates a German example of an inflectional paradigm class applied to a compound involving the use of an orthographic separator. The compound forms are deduced from the two components presented in the *List Of Components* instance. The lemma is associated with a paradigm class that describes how to build the compound. The first component is selected, then an "s" is added and the second component is added with the initial letter transformed into a lowercase letter.

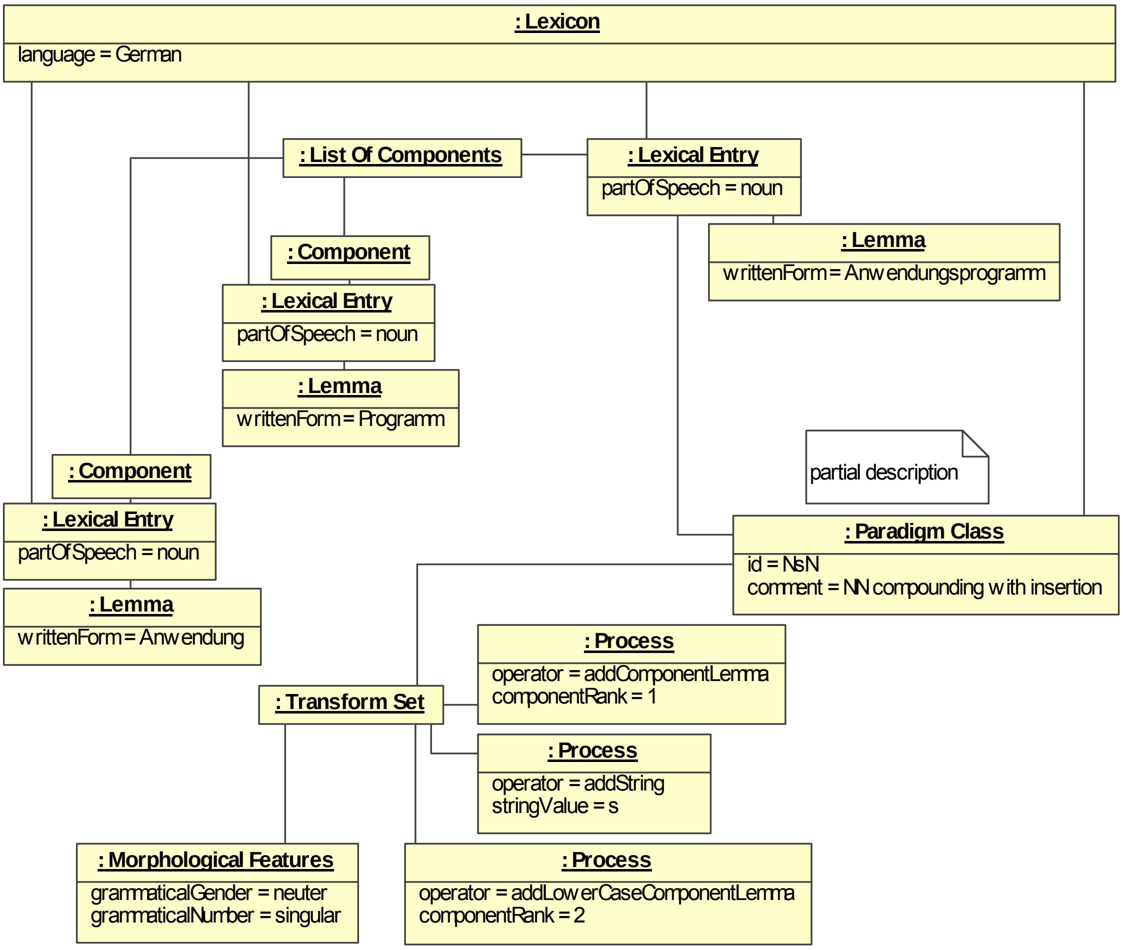


Figure L.9: example of a lemma based composition

L.1.3.11 Stem based composition: Schulkind

The following figure illustrates a second German example of a paradigm class applied to noun-noun compounds involving the use of a stem. In cases where the initial word ends in "e" or "en", there is a truncation of letters from the end of a word during compounding. In this example, "Shule+Kind" produces "Shulkind". Instead of using a "delete" operation on "Schule", this example illustrates an approach which treats "schul" as the stem of the noun "Schule", with the assumption that internal additions transform the initial letter into lowercase.

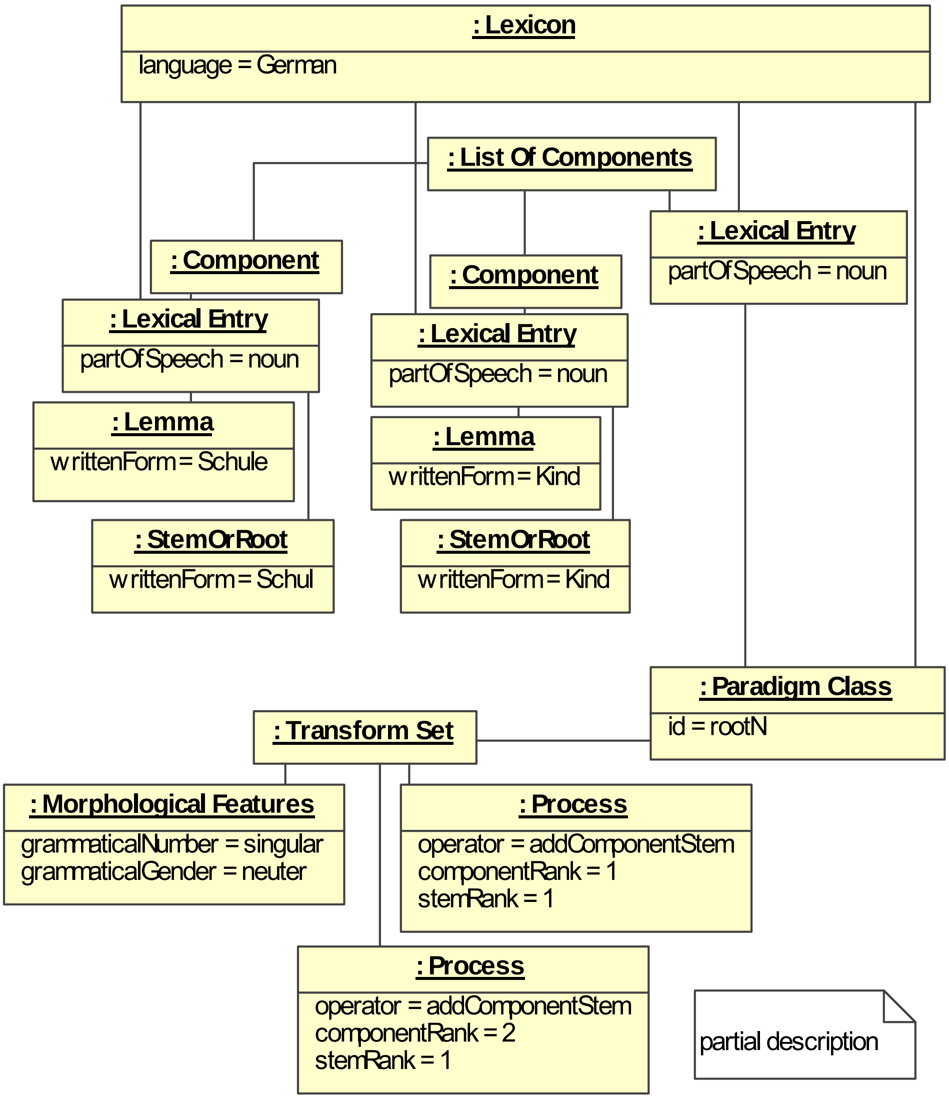


Figure L.10: Example of a stem based composition

Annex M (normative) NLP multiword expression patterns extension

M.1 Objectives

The purpose of this section is to allow a representation of the internal (semi-fixed or flexible) structure of MWEs in a given language.

In all languages, MWEs comprise a wide-range of distinct but related phenomena such as collocations, phrasal verbs, noun-noun compounds and many others. Some systems or linguistic traditions also treat shorter idioms as MWEs. Even though some MWEs are fixed, and do not present internal variation such as *ad hoc,* others are much more flexible and allow different degrees of internal variation and modification.

M.2 Absence versus presence of MWE patterns

It is also possible to describe some MWEs using the Paradigm Class extension, but such cases are limited to simple MWEs without any variation. In contrast, this annex allows for the analysis of the entire MWE based on the grammar of the language.

M.3 Class diagram

Essentially, the *MWE Pattern* class is a phrase structure grammar.

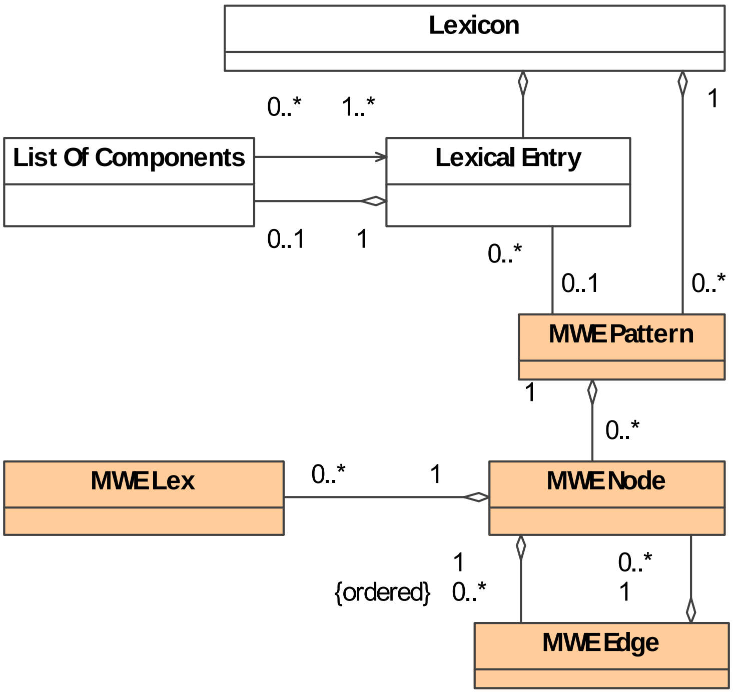


Figure M.1: MWE pattern class model

M.4 Description of MWE pattern model

M.4.1 MWE Pattern class

MWE Pattern is a class representing a certain type of lexical combination phenomena. A pattern always refers to the *List Of Components* instances of the *Lexical Entry* instance*. MWE Pattern* shall not to be used for a *Lexical Entry* instance that is not associated with a *List Of Component*s instance. An *MWE Pattern* instance is described using *MWE Node* instances.

M.4.2 MWE Node class

MWE Node is a class representing the details about the structure of the MWE. A *Combiner* instance can be linked with zero to many *MWE Edge* instances.

M.4.3 MWE Edge class

MWE Edge is a class representing a smaller element of information as the *MWE Node* class. A *MWE Edge* instance may itself be associated recursively to a *MWE Node* instance.

M.4.4 MWE Lex class

MWE Lex is a class representing a reference to a lexical component. The objective of the whole package being to provide a generic representation of MWE combinations within a give language, the components are not referenced directly but on the contrary, they are referenced by their respective ordering as specified in the *List Of Component* instance.

Annex N (informative) NLP multiword expression patterns examples

N.1.1 Example of class adornment

Classes may be adorned with the following attributes:

Table N.1: Class adornment for NLP Multiword expression patterns

Class name	Example of attributes	Comment
MWE Pattern	id comment	The purpose of an *MWE Pattern* instance is to be shared by all the forms that have this structure. *MWE Pattern* instances are shared resources and therefore are associated with an *id* so that they can be targeted by cross-reference links.
MWE Node	syntacticConstituent semanticRestriction grammaticalNumber	
MWE Edge	function	
MWE Lex	structureHead rank graphicalSeparator	

N.1.2 Example of word description

The example is "to throw somebody to the lions".

The structure contains three sub-structures:

- A fully specified verb: *throw*, referenced by rank one within the *List Of Components* instance;
- A noun phrase: *somebody*, that is not fully specified in the sense that the only restriction that is expressed is that the nucleus of the phrase must be of type /human/.
- A fully specified second noun phrase *to the lions* referenced by ranks one, two and three within the *List Of Components* instances. This prepositional phrase is labelled as /plural/.

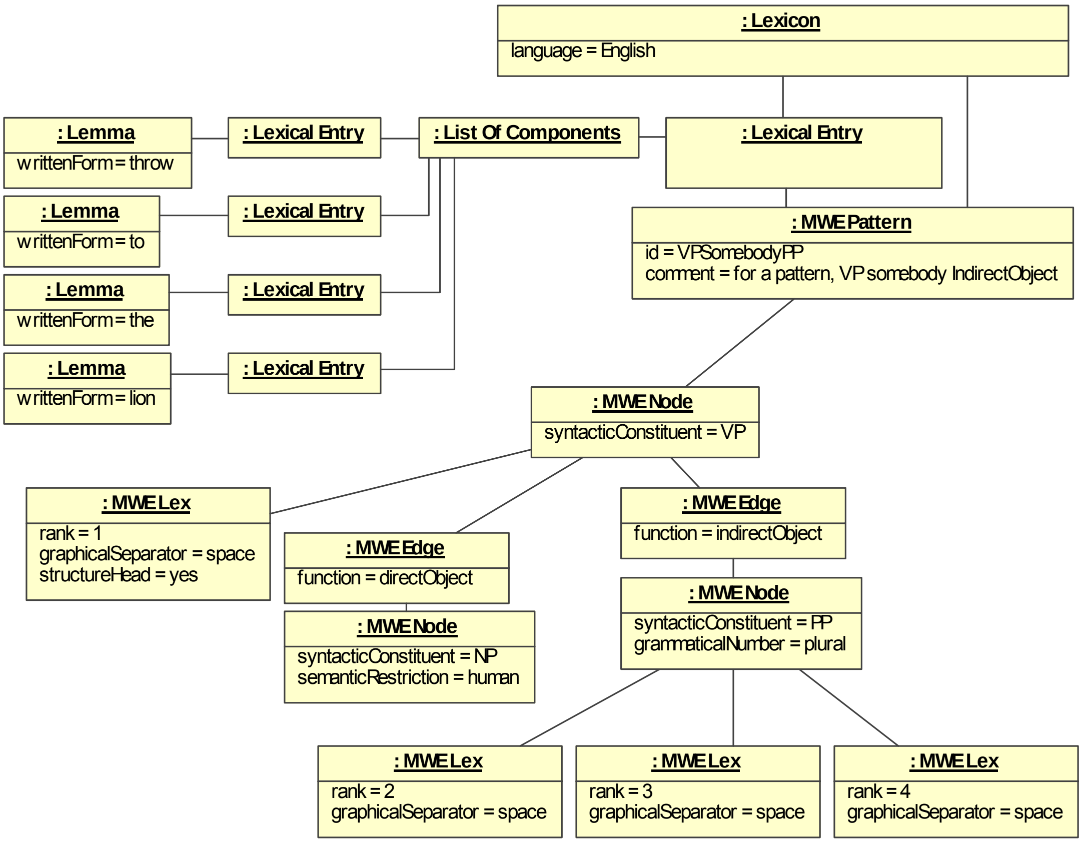


Figure N.1: MWE pattern example

Annex O (normative) Constraint expression extension

O.1 Objectives

The aim is to allow the description of constraints on pairs of attribute-values. The scope of the constraints is the *Lexicon* instance.

O.1.1 Class diagram

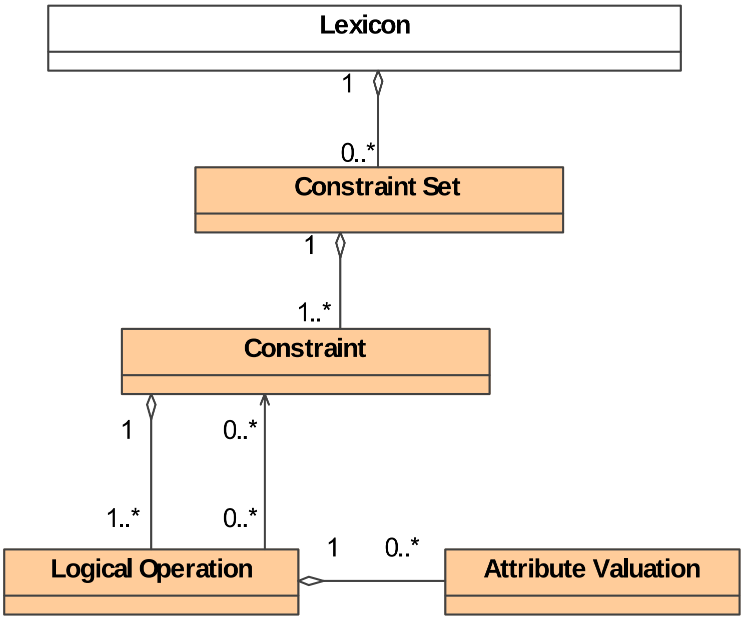


Figure O.1 Constraint Expression model

O.1.2 Description of the constraint expression model

Constraint Set

Constraint Set is a class representing a group of constraints. *Constraint Set* class is associated to *Lexicon* class with a zero to many cardinality.

Example: a constraint set defining how to combine every part of speech value of a given language with the morphological features of this language.

Example: it is possible to define a *Constraint Set* instance for the morphological representations of a *Lexicon* instance and a *Constraint Set* instance for the syntactic representations of this given lexicon.

Constraint

Constraint is a class representing one or several boolean expressions that must respected in a given *Lexicon* instance.

Logical Operation

Logical Operation is a class representing a boolean expression between *Attribute Valuation* instances and possibly *Constraint* instances.

Example: A *Logical Operation* instance may represent a conjunction (e.g. "and") or a disjunction (e.g. "or").

Note: it is possible to define recursively a *Constraint* instance as a combination of other *Constraint* instances.

Attribute Valuation

Attribute Valuation is a class representing a pair between an attribute name of an LMF class and a value of this particular attribute. A special value /any/ means that the attribute must be present but that any value may be set.

Example: An *attribute Valuation* instance may be the pair *partOfSpeech* and *adjective*.

Note: the class name of this attribute is not specified.

Annex P (informative) Constraint expression example

P.1 Example of class adornment

Table O.1: Class adornment for constraint expression

Class name	Example of attributes	Comment
Constraint Set	label comment	
Constraint	label	
Logical Operation	operator	The values for the operator attribute may be /and/, /or/, /not/.
Attribute Valuation	partOfSpeech grammaticalNumber grammaticalGender	The value /any/ means that the attribute must exist and that it could be set to any value.

P.2 Example of constraint expression

In this French example, the *Constraint Set* instance holds only one constraint that states that an adjective may vary according to grammatical number and grammatical gender. The *Attribute Valuation* instance for the part of speech is set with a particular value, that is *adjective*. On the contrary, the *Attribute Valuation* instance for the morphological feature is set with the specification that the number and gender may hold any value.

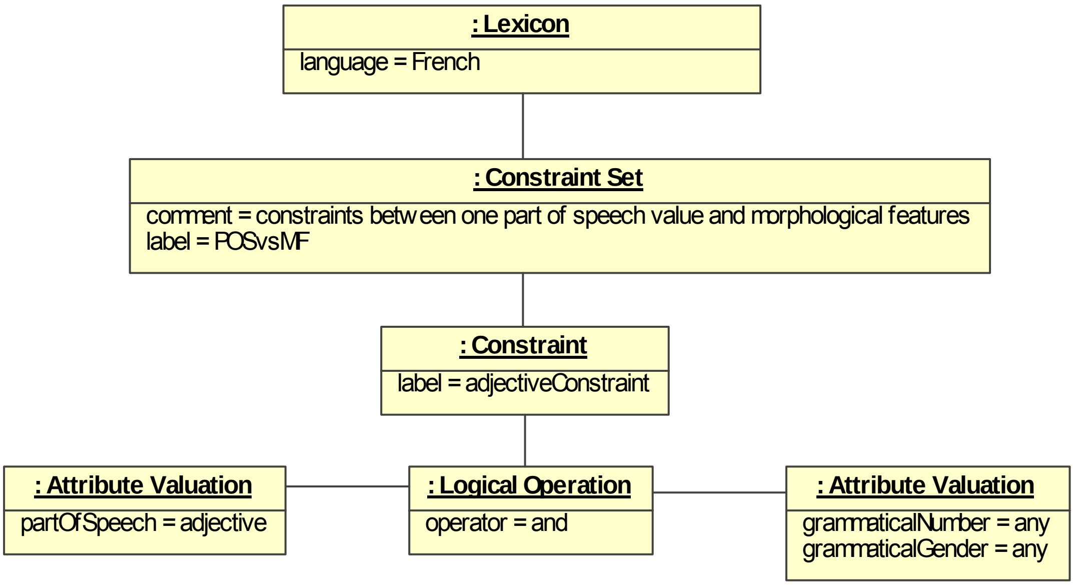


Figure P.1: example of constraint expression

Annex Q (informative) Connection with Terminological Markup Framework (TMF) and other concept based representation systems

Q.1 Introduction

The aim of this annex is to provide some guidelines on how to link an LMF compliant lexicon with external systems like Terminological Markup Framework (TMF) resources and other concept based representation systems.

LMF is focused on linguistics properties of words. It is not the purpose of a lexicon to provide a complex knowledge organization system.

Q.2 Configurations

There are two main different types of lexical resources:

- Configuration 1: *Lexical Resource* instance is monolingual, thus, only one *Lexicon* instance is aggregated in the *Lexical Resource* instance. In this situation, the external linking is attached to a *Sense* instance.

- Configuration 2: *Lexical Resource* instance is multilingual, thus, more than one *Lexicon* instance is aggregated in the *Lexical Resource* instance and *Sense Axis* instances are recorded in order to link *Sense* instances belonging to different languages. In this situation, the external linking is attached to a *Sense Axis* instance.

As defined in Core Package, an attribute is unique, but a *Sense* instance (respectively *Sense Axis* instance) may need to be associated to more than one node of an external system. Thus, the linking is not represented by an attribute-value pair but by the means of a specific class.

In a monolingual configuration, the external linking is represented by a *Monolingual External Ref* instance, as specified in NLP semantic package. The attributes */externalSystem/* and */externalReference/* are provided to refer respectively to the name(s) of the external system and to the specific relevant node in this given external system.

In a multilingual configuration, the external linking is represented by a *Multilingual External Ref* instance, as specified in the NLP multilingual notation package. The attributes */externalSystem/* and */externalReference/* are provided to refer respectively to the name(s) of the external system and to the specific relevant node in this given external system.

Annex R (informative) DTD for NLP

R.1 Foreword

The following material is provided for information only and covers core package and NLP annexes. XML elements are transcoded from the LMF UML class diagrams [10] with the class adornment implemented as a set of DC elements (DC stands for *data category*).

A DTD for MRD or for a different combination of annexes is not difficult to define and left to the reader. A user can decide to define another DTD or schema to implement LMF. It is also possible for instance to use the XML structures that are defined in the Feature Structure Representation standard (i.e. ISO 24610).

R.2 DTD for core package and NLP annexes

```
<?xml version='1.0' encoding="UTF-8"?>
      <!-- DTD for LMFNLP packages-->
      <!--#################### Core package-->
<!ELEMENT LexicalResource (DC*, GlobalInformation, Lexicon+, SenseAxis*, TransferAxis*, ExampleAxis*)>
<!ATTLIST LexicalResource
  dtdVersion CDATA     #FIXED "1.2">
<!ELEMENT GlobalInformation (DC*)>
<!ELEMENT Lexicon (DC*, LexicalEntry+,  SubcategorizationFrame*, SubcategorizationFrameSet*, SemanticPredicate*, Synset*,
                  ParadigmClass*, TransformClass*, MWEPattern*, ConstraintSet*)>
<!ELEMENT LexicalEntry (DC*, Lemma, WordForm*, StemOrRoot*, DerivedForm*, ReferredRoot*, ListOfComponents?, Sense*,
                      SyntacticBehaviour*)>
<!ATTLIST LexicalEntry
  id      ID #IMPLIED
  paradigmClass IDREFS #IMPLIED
  mwePattern  IDREF #IMPLIED>
<!ELEMENT Sense (DC*, PredicativeRepresentation*, SenseExample*, SemanticDefinition*, SenseRelation*,
                 MonolingualExternalRef*)>
<!ATTLIST Sense
  id      ID #IMPLIED
  inherit  IDREFS #IMPLIED
  synset IDREF #IMPLIED>
      <!--#################### Package for Morphology -->
<!ELEMENT Lemma (DC*, FormRepresentation*)>
<!ELEMENT WordForm (DC*, FormRepresentation*)>
<!ELEMENT StemOrRoot (DC*, FormRepresentation*, MorphologicalFeatures*)>
<!ELEMENT FormRepresentation (DC*)>
<!ELEMENT DerivedForm (DC*, FormRepresentation*)>
<!ATTLIST DerivedForm
  targets IDREFS #IMPLIED>
<!ELEMENT ReferredRoot (DC*, FormRepresentation*)>
<!ATTLIST ReferredForm
```

```
  targets IDREFS #IMPLIED>
<!ELEMENT ListOfComponents (DC*, Component+)>
<!ELEMENT Component (DC*)>
<!ATTLIST Component
  entry IDREF #REQUIRED>
<!ELEMENT MorphologicalFeatures (DC*)>
      <!--#################### Package for Syntax -->
<!ELEMENT SyntacticBehaviour (DC*)>
<!ATTLIST SyntacticBehaviour
  id                         ID #IMPLIED
  senses                     IDREFS #IMPLIED
  subcategorizationFrames    IDREFS #IMPLIED
  subcategorizationFrameSets IDREFS #IMPLIED>
<!ELEMENT SubcategorizationFrame (DC*, LexemeProperty?, SyntacticArgument*)>
<!ATTLIST SubcategorizationFrame
  id         ID #IMPLIED
  inherit    IDREFS #IMPLIED>
<!ELEMENT LexemeProperty (DC*)>
<!ELEMENT SyntacticArgument (DC*)>
<!ATTLIST SyntacticArgument
  id         ID #IMPLIED
  target     IDREF #IMPLIED>
<!ELEMENT SubcategorizationFrameSet (DC*, SynArgMap*)>
<!ATTLIST SubcategorizationFrameSet
  id                         ID #IMPLIED
  subcategorizationFrames    IDREFS #IMPLIED
  inherit                    IDREFS #IMPLIED>
<!ELEMENT SynArgMap (DC*)>
<!ATTLIST SynArgMap
  arg1     IDREF #REQUIRED
  arg2     IDREF #REQUIRED>
      <!--#################### Package for Semantics -->
<!ELEMENT PredicativeRepresentation (DC*,SynSemArgMap*)>
<!ATTLIST PredicativeRepresentation
 predicate   ID #REQUIRED>
<!ELEMENT SemanticPredicate (DC*, SemanticArgument*, PredicateRelation*)>
<!ATTLIST SemanticPredicate
  id         ID #REQUIRED>
<!ELEMENT SemanticArgument (DC*)>
<!ATTLIST SemanticArgument
  id         ID #IMPLIED>
<!ELEMENT SynSemArgMap (DC*)>
<!ATTLIST SynSemArgMap
 synArg      IDREF #REQUIRED
 semArg      IDREF #REQUIRED>
<!ELEMENT PredicateRelation (DC*)>
<!ATTLIST PredicateRelation
```

```
   targets          IDREFS #IMPLIED>
<!ELEMENT SenseExample (DC*)>
<!ATTLIST SenseExample
   id              ID #IMPLIED>
<!ELEMENT SemanticDefinition (DC*, Statement*)>
<!ELEMENT Statement (DC*)>
<!ELEMENT Synset (DC*, SemanticDefinition*, SynsetRelation*, MonolingualExternalRef*)>
<!ATTLIST Synset
   id              ID #IMPLIED>
<!ELEMENT SynsetRelation (DC*)>
<!ATTLIST SynsetRelation
   targets          IDREFS #IMPLIED>
<!ELEMENT MonoLingualExternalRef (DC*)>
<!ELEMENT SenseRelation (DC*)>
<!ATTLIST SenseRelation
   targets  IDREFS #REQUIRED>
         <!--#################### Package for Multilingual notations -->
<!ELEMENT SenseAxis (DC*, SenseAxisRelation*, InterlingualExternalRef*)>
<!ATTLIST SenseAxis
   id             ID #IMPLIED
   senses     IDREFS #IMPLIED
   synsets    IDREFS #IMPLIED>
<!ELEMENT InterlingualExternalRef (DC*)>
<!ELEMENT SenseAxisRelation (DC*)>
<!ATTLIST SenseAxisRelation
   targets          IDREFS #REQUIRED>
<!ELEMENT TransferAxis (DC*, TransferAxisRelation*, SourceTest*, TargetTest*)>
<!ATTLIST TransferAxis
   id              ID #IMPLIED
   synBehaviours IDREFS #IMPLIED>
<!ELEMENT TransferAxisRelation (DC*)>
<!ATTLIST TransferAxisRelation
   targets          IDREFS #REQUIRED>
<!ELEMENT SourceTest (DC*)>
<!ATTLIST SourceTest
   synBehaviours IDREFS #REQUIRED>
<!ELEMENT TargetTest (DC*)>
<!ATTLIST TargetTest
   synBehaviours IDREFS #REQUIRED>
<!ELEMENT ExampleAxis (DC*, ExampleAxisRelation*)>
<!ATTLIST ExampleAxis
   id              ID #IMPLIED
   examples IDREFS #IMPLIED>
<!ELEMENT ExampleAxisRelation (DC*)>
<!ATTLIST ExampleAxisRelation
   targets          IDREFS #REQUIRED>
         <!--#################### Package for paradigm classes -->
```

```
<!ELEMENT ParadigmClass (DC*, TransformSet*, Affix*, PrefixSlot*, InfixSlot*, SuffixSlot*)>
<!ATTLIST ParadigmClass
   id       ID #REQUIRED>
<!ELEMENT TransformSet (DC*, Process*, MorphologicalFeatures*)>
<!ELEMENT Process (DC*, Condition*, MorphologicalFeatures*)>
<!ELEMENT Condition (DC*)>
<!ATTLIST Condition
   id      ID #IMPLIED>
<!ELEMENT Affix (DC*, FormRepresentation*, Condition*, AffixAllomorph*, MorphologicalFeatures*)>
<!ELEMENT AffixAllomorph (DC*, FormRepresentation*)>
<!ATTLIST AffixAllomorph
   conditions IDREFS #IMPLIED>
<!ELEMENT PrefixSlot (DC*, Affix*)>
<!ELEMENT InfixSlot (DC*, Affix*)>
<!ELEMENT SuffixSlot (DC*, Affix*)>
<!ELEMENT TransformClass (DC*)>
<!ATTLIST TransformClass
   id       ID #REQUIRED>
         <!--#################### Package for MWE patterns -->
<!ELEMENT MWEPattern (DC*, MWENode*)>
<!ELEMENT MWENode (DC*, MWEEdge*, MWELex)>
<!ELEMENT MWEEdge (DC*, MWENode*)>
<!ELEMENT MWELex (DC*)>
         <!--#################### Package for Constraint expression -->
<!ELEMENT ConstraintSet (DC*, Constraint*)>
<!ELEMENT Constraint (DC*, LogicalOperation*)>
<!ATTLIST Constraint
   id       ID #IMPLIED>
<!ELEMENT LogicalOperation (DC*, AttributeValuation*)>
<ATTLIST LogicalOperation
  constraints  IDREFS #IMPLIED>
<!ELEMENT AttributeValuation (DC*)>
         <!--#################### for datcat adornment. DC stands for data category -->
<!ELEMENT DC EMPTY>
         <!-- att=constant to be taken from the DCR -->
         <!-- val=free string or constant to be taken from the DCR-->
<!ATTLIST DC
   att      CDATA #REQUIRED
   val      CDATA #REQUIRED>
```

Bibliography

[1] IETF BCP 47, currently [June, 2006] represented by *RFC 3066 "Tags for the Identification of Languages"*, H. Alvestrand, January 2001, http://www.w3.org/TR/SVGMobile12/refs.html

[2] Rumbaugh J., Jacobson I., Booch G. 2004 The unified modeling language reference manual, second edition, Addison Wesley

[3] CLIPS. 2000, ff. Clips: Browsing Semantic and Syntactic Data. http://www.ilc.cnr.it/clips/SYN_SEM/browsing.htm

[4] Mel'cuk I., Clas A., Polguère A. 1995 Introduction à la lexicologie explicative et combinatoire. Duculot. Bruxelles. (*Dictionnaire Explicatif et Combinatoire,* http://www.olst.umontreal.ca/decfr.html)

[5] Calzolari N., Mc Naught J., Zampolli A. 1996 Eagles, editors introduction. http://www.ilc.cnr.it/EAGLES96/edintro/edintro.html

[6] Calzolari N., Bertagna F., Lenci A., Monachini M. editors, 2003. Standards and best Practice for Multilingual Computational Lexicons. MILE (The Multilingual ISLE Lexical Entry). ISLE CLWG Deliverable D2.2 & 3.2 Pisa.

[7] Wordnet 2.1 http://wordnet.princeton.edu

[8] Fellbaum C. 1998, A semantic network of English: the mother of all WordNets, in Vossen (ed), EuroWordNet: a multilingual database with lexical semantic networks. Kluwer academic publishers

[9] Antoni-Lay M-H., Francopoulo G., Zaysser L. 1994, A generic model for reusable lexicons: the GENELEX project, Literary and linguistic computing 9(1) 47-54.

[10] Carlson D. 2001, Modeling XML applications with UML, Addision Wesley

[11] Mel'cuck I. 1993-2000 Cours de morphologie générale, 5 volumes, Presses de l'Université de Montréal

[12] Fradin B. 2003 Nouvelles approches en morphologie, Presses universitaires de France

[13] Matthews P.H. 1991 Morphology, 2nd ed, Cambridge University press

[14] Matthews P.H. 1972 Inflectional morphology: a theoretical study based on aspects of latin verb conjugation, Cambridge University press

[15] Stump G. 2001 Inflectional morphology: a theory of paradigm structure, Cambridge University press

[16] Comrie B. 1989 Language universals and linguistic typology, 2nd ed, University of Chicago press

[17] Aronoff M. 1994 Morphology by itself: stems and inflectional classes, MIT press

[18] Wright S.E. 2004 A global data category registry for interoperable language resources LREC Lisbon

[19] Ide N., Romary L. 2004 A registry of standard data categories for linguistic annotation LREC Lisbon

[20] Francopoulo G., Declerck T., Monachini M., Romary L. 2006 The relevance of standards for research infrastructure LREC Genoa

[21] Mel'cuk I.1984-1999 Dictionnaire explicatif et combinatoire du français contemporain, 4 volumes. Presses de l'Université de Montréal

[22] Blachère R, Gaudefroy-Demombynes 2004, Grammaire de l'arabe classique. Maisonneuve & Larose

[23] Lombard D. 1991 Introduction à l'indonésien. Archipel, Paris.