INTERNATIONAL STANDARD

ISO 24613

First edition
2008-11-15

Language resource management — Lexical markup framework (LMF)

Gestion de ressources langagières — Cadre de balisage lexical (LMF)

Reference number
ISO 24613:2008(E)

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Contents

Page

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 2.

The main task of technical committees is to prepare International Standards. 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.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights.

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

ISO 24613 is designed to coordinate closely with ISO 12620, *Terminology and other content and language resources — Data categories — Specification of data categories and management of a Data Category Registry for language resources*[1)], and ISO 16642, *Computer applications in terminology — Terminological markup framework*.

1) To be published. (Revision of ISO 12620:1999)

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 of a lexical entry, including information on the form. 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 analysing and designing such linked systems.

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

a) machine readable dictionaries;

b) natural language processing 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 analysing 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 analysing 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.

Language resource management — Lexical markup framework (LMF)

1 Scope

This International Standard describes the Lexical Markup Framework (LMF), a metamodel 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[2]. These mechanisms will present existing lexicons as far as possible. If this is impossible, problematic information will be identified and isolated.

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO 639 (all parts), *Codes for the representation of names of languages*

ISO 1087-1, *Terminology work — Vocabulary — Part 1: Theory and application*

ISO 1087-2, *Terminology work — Vocabulary — Part 2: Computer applications*

ISO 12620, *Terminology and other content and language resources — Data categories — Specification of data categories and management of a Data Category Registry for language resources* [3]

ISO 15924, *Information and documentation — Code for the representation of names of scripts*

3 Terms and definitions

For the purposes of this International Standard, the terms and definitions given in ISO 1087-1, ISO 1087-2 and the following apply [4].

3.1
abbreviated form
form (3.14) resulting from the omission of any part of the **full form** (3.16) of the same **lexeme** (3.25)

2) LMF 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].

3) To be published. (Revision of ISO 12620:1999)

4) It is worth noting that we have purposely avoided defining and using highly controversial terms such as “word”, “morpheme”, “base”, “fusion”, “ergative”, “paradigm”, and “collocation”.

3.2
adjunct
non-essential element associated with a verb as opposed to **syntactic arguments** (3.43)

EXAMPLE Alfred (syntactic argument) reads a book (syntactic argument) today (adjunct).

NOTE Adverbs are possible adjuncts for a sentence.

3.3
affix
bound morph (3.8) that may contribute to a **form** (3.14) and participates in the process of **inflection** (3.20), **agglutination** (3.5), **derivation** (3.12) or **composition** (3.9)

NOTE Affixes function as prefixes (pre-positioned), suffixes (post-positioned), infixes (inserted) and circumfixes (combination of prefix and suffix).

3.4
affixation
process in which an **affix** (3.3) is added to a **lemma** (3.24) or a **stem** (3.40)

3.5
agglutination
process in which an **agglutinated form** (3.6) is made up

3.6
agglutinated form
word form (3.47) that a **lexeme** (3.25) can take when used in a sentence or a phrase within an **agglutinating language** (3.7)

3.7
agglutinating language
language where the different **word forms** (3.47) of the same **lexeme** (3.25) exhibit a variation and that may consist of more than one **morph** (3.31) but the boundaries between morphs are always clear-cut

EXAMPLE Korean, Japanese, Hungarian and Turkish are agglutinating languages [16].

3.8
bound morph
morph that appears only together with one or several other **morphs** (3.31)

3.9
composition
compounding
lexeme (3.25) formation in which a new lexeme [associated with its **part of speech** (3.37) information] is formed by adjoining at least two lexemes, in their original **forms** (3.14) or with slight transformations

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

3.10
compound
lexeme (3.25) associated with **part of speech** (3.37) information that is built from two or more lexemes

3.11
compound form
form (3.14) resulting from a **composition** (3.9)

3.12
derivation
change in the **forms** (3.14) of a **lexeme** (3.25) to create a new lexeme, usually by modifying the **stem** (3.40) or by **affixation** (3.4)

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*.

3.13
derived form
form (3.14) resulting from a **derivation** (3.12)

3.14
form
sequence of **morphs** (3.31)

3.15
free morph
morph (3.31) that may stand by itself

EXAMPLE The English noun *boy*.

3.16
full form
complete representation of a **lexeme** (3.25) for which there is an **abbreviated form** (3.6)

3.17
grammatical feature
property associated to the **inflected** (3.19), **agglutinated** (3.6), **compound** (3.11) or **derived form** (3.13) that describes the grammatical attribute of the form

NOTE An example of a grammatical feature is: /grammatical gender/. (Following the convention adopted in the revision of ISO 12620, the slashes are used in order to delimit data category values.)

3.18
graph
minimal unit in a written language including letters, pictograms, ideograms, numerals and punctuations

3.19
inflected form
word form (3.47) that a **lexeme** (3.25) can take when used in a sentence or a phrase within an **inflectional language** (3.21)

3.20
inflection
process in which an **inflected form** (3.19) is made up

3.21
inflectional language
inflecting language
language where the different **word forms** (3.47) of the same **lexeme** (3.25) exhibit a variation and where there is no clear-cut boundary between **morphs** (3.31) in that morphs are generally fused together to yield a single, non-segmentable **form** (3.14)

EXAMPLE Spanish, Italian, French and English are inflectional languages [16].

3.22
interlingua
abstract intermediary language used in the machine translation of human languages

3.23
isolating language
language where the vast majority of **morphs** (3.31) are **free morphs** (3.15)

EXAMPLE Chinese is an isolating language.

3.24
lemma
lemmatized form
canonical form
conventional **form** (3.14) chosen to represent a **lexeme** (3.25)

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.

3.25
lexeme
abstract unit generally associated with a set of **forms** (3.14) sharing a common meaning

3.26
lexical entry
container for managing one or several **forms** (3.14) and possibly one or several meanings in order to describe a **lexeme** (3.25)

3.27
lexical resource
lexical database
database consisting of one or several **lexicons** (3.28)

3.28
lexicon
resource comprising **lexical entries** (3.26) 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.

3.29
machine readable dictionary
MRD
electronic **lexical resource** (3.27) 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* now.

3.30
machine translation lexicon
electronic **lexical resource** (3.27) in which the individual **lexical entries** (3.26) contain equivalents in two or more languages together with morphological, syntactic and/or semantic information to facilitate automatic or semi-automatic processing of **lexemes** (3.25) during machine translation

3.31
morph
sequence of **graphs** (3.18) or sequence of **phones** (3.38)

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

3.32
morphological pattern
set of associations and/or operations that build the various forms of a **lexeme** (3.25), possibly by **inflection** (3.20), **agglutination** (3.5), **composition** (3.9) or **derivation** (3.12), depending on the language

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

3.33
morphology
description of the structure and formation of **forms** (3.14)

3.34
multiword expression
MWE
lexeme (3.25) made up of a sequence of two or more 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.

3.35
natural language processing
NLP
field covering knowledge and techniques involved in the processing of linguistic data by a computer

3.36
orthography
way of spelling or writing **lexemes** (3.25) 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.

3.37
part of speech
lexical category
word class
category assigned to a **lexeme** (3.25) based on its grammatical properties

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

3.38
phone
minimal unit in the sound system of a language

3.39
script
set of graphic characters used for the written **form** (3.14) of one or more languages

[ISO/IEC 10646:2003, definition 4.37]

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.

EXAMPLE Hiragana, Katakana, Latin and Cyrillic.

3.40
stem
sequence of **morphs** (3.31) that is smaller than or equal to the **form** (3.14) of a single **lexeme** (3.25) and that may be affected by an **inflectional** (3.20), **agglutinative** (3.5), **compositional** (3.9) or **derivation** (3.12) process

3.41
subcategorization frame
valence
valency
set of restrictions on a **lexeme** (3.25) indicating the properties of the **syntactic arguments** (3.43) that can or must occur with this given lexeme

3.42
support verb
verb that makes a generic semantic contribution to the context and that combines with a noun to form a **lexeme** (3.25)

EXAMPLE *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.

3.43
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) reads a book (syntactic argument) today (adjunct).

3.44
transcription
form (3.14) 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

3.45
transliteration
form (3.14) resulting from the conversion of one writing system into another, usually through a one to one correspondence between characters

3.46
variant
one of the alternative **forms** (3.14) of a **lexeme** (3.25)

3.47
word form
form (3.14) that a **lexeme** (3.25) takes when used in a sentence or a phrase

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 Language coding

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 ISO 639 have been addressed in external standards supported by the

technology community. It is recommended that users consult the current edition of IETF Best Common Practices (BCP) 47, *Tags for the Identification of Languages* in order to resolve issues involving choosing and matching identifiers for use in electronic environments [1].

4.3 Script Coding

When the script code is not part of the language identifier, script identifiers shall conform to criteria specified in ISO 15924.

4.4 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) located at www.isocat.org.

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** (5.2). Additionally, developers can optionally use classes that are defined in the **LMF extensions** (see relevant annexes). Developers shall define a data category selection (DCS) as specified for **LMF data category selection procedures** (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, see Figure 1.

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.

The *Global Information* instance must contain at least the following attribute:

— /language coding/ This attribute specifies which standard is used in order to code the language names within the whole *Lexical Resource* instance.

The *Global Information* instance may contain the following attributes:

— /script coding/ This attribute specifies which standard is used in order to code the script names within the whole *Lexical Resource* instance;

— /character coding/ This attribute specifies which Unicode version is used within the whole *Lexical Resource* instance.

NOTE Other standard related precisions may be specified on the *Global Information* instance.

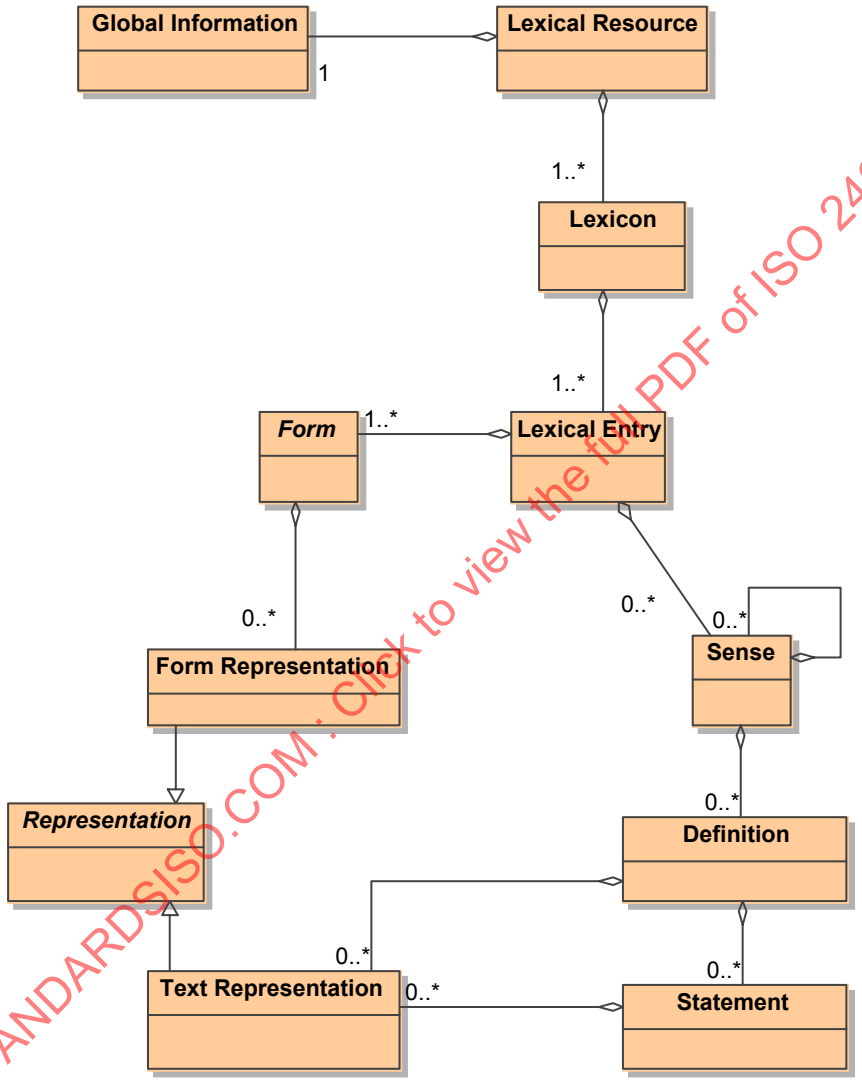

Figure 1 — LMF core package

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 an abstract class representing a lexeme, a morphological variant of a lexeme or a morph. The *Form* class manages one or more orthographical variants of the abstract *Form* as well as data categories that describe the attributes of the word form (e.g. lemma, pronunciation, syllabification). The *Form* class allows subclasses.

5.2.6 Form Representation class

Form Representation is a class representing one variant orthography of a *Form*. When there is more than one variant orthography, the *Form Representation* class contains a Unicode string representing the *Form* as well as, if needed, the unique attribute-value pairs that describe the specific language, script, and orthography.

5.2.7 Representation class

Representation is an abstract class representing a Unicode string as well as, if needed, the unique attribute-value pairs that describe the specific language, script, and orthography. The *Representation* class allows subclasses.

5.2.8 Sense class

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

5.2.9 Definition class

Definition is a class representing a narrative description of a sense. It is displayed for human users to facilitate their understanding of the meaning of a *Lexical Entry* and is not meant to be processable by computer programs. A *Sense* instance can have zero to many definitions. Each *Definition* instance may be associated with zero to many *Text Representation* instances in order to manage the text definition in more than one language or script. The narrative description can be expressed in a different language and/or script than the one for the *Lexical Entry* instance.

EXAMPLE In a *Lexical Entry* for *abbess*, the narrative description may be *woman who is in charge of a convent*.

5.2.10 Statement class

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

NOTE A full example is given in WordNet context in Annex H.

5.2.11 Text Representation class

Text Representation is a class representing one textual content of *Definition* or *Statement*. When there is more than one variant orthography, the *Text Representation* class contains a Unicode string representing the textual content as well as the unique attribute-value pairs that describe the specific language, script, and orthography.

EXAMPLE In a Bambara lexicon, a given lexical entry may be associated with one definition that is expressed in Bambara for native speakers and in French for French speakers that are learning Bambara. The *Definition* instance will thus have two *Text Representation* instances, each with a specific narrative content and an attribute-value pair for the language information.

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 Figure 2.

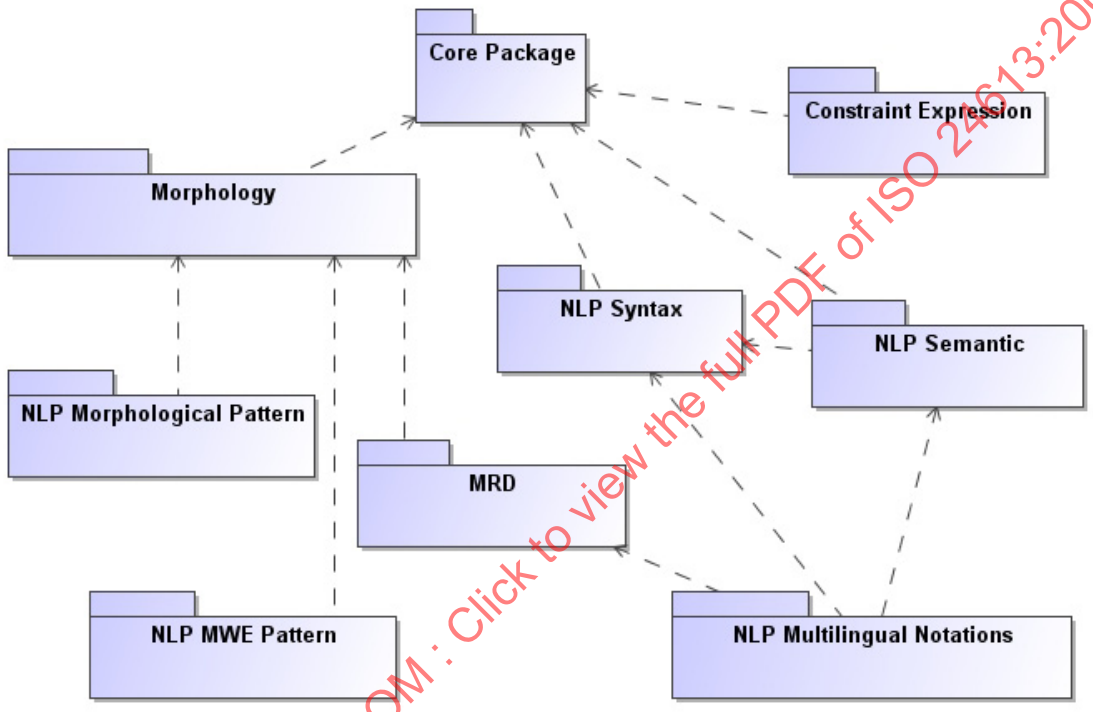

Figure 2 — Dependencies between the LMF core and extension packages

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 Annexes A, B, C, D, E, F, G, H, I, J, K, L, M, N, O and P of this International Standard. Annexes A, C, E, G, I, K, M and O form an integral part 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. See References [18], [19] and [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, see Figure 3.

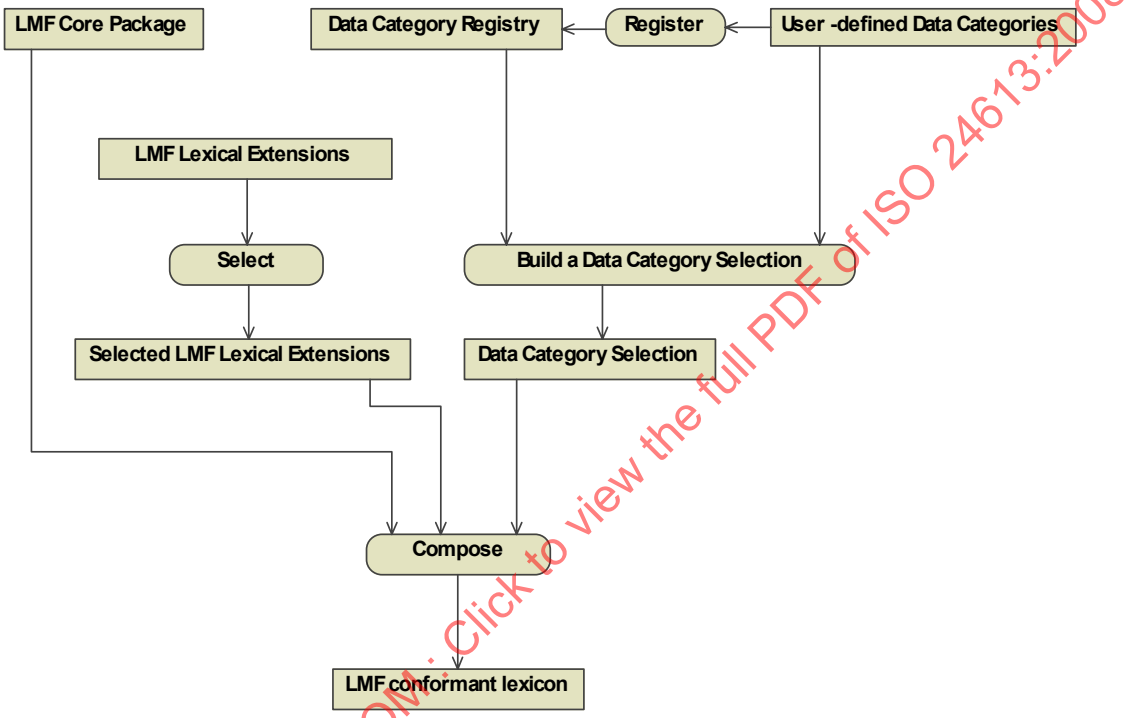

Figure 3 — LMF process

Annex A
(normative)

Morphology extension

A.1 Objectives

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

EXAMPLE When applied to an inflectional language, "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 Annex K (on morphological patterns).

A.2 Class diagram

The morphology extension is organized as described in Figure A.1.

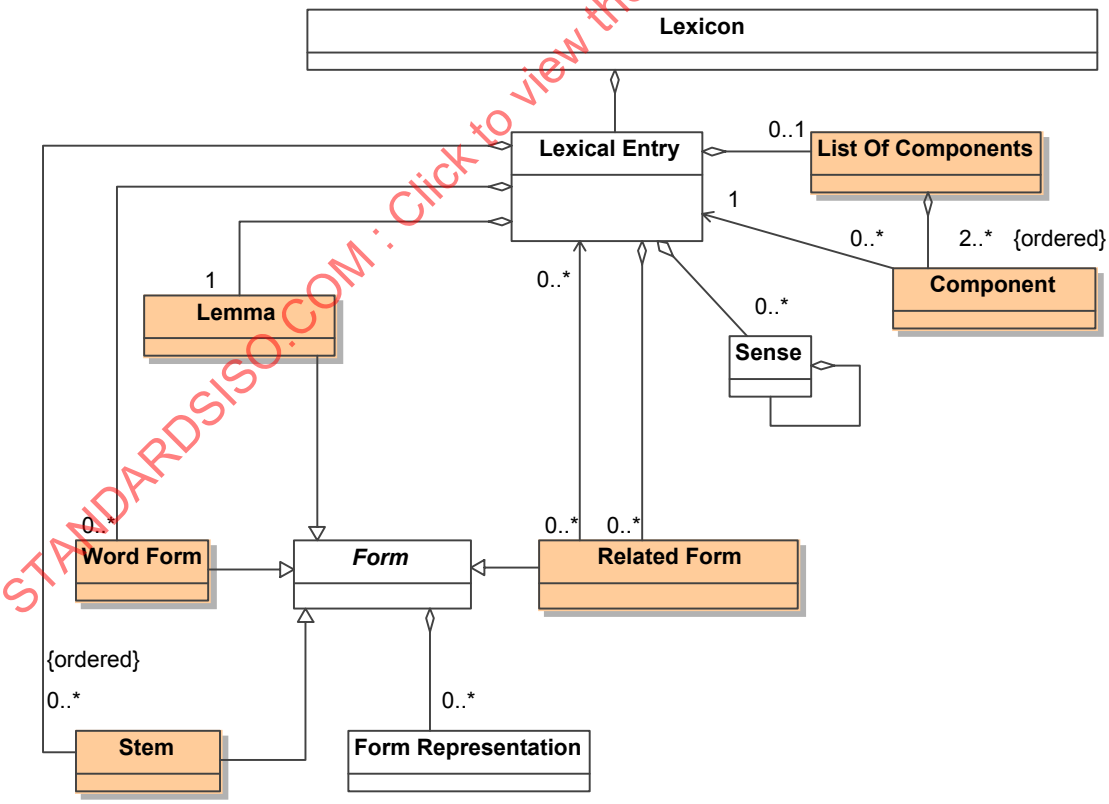

Figure A.1 — Morphology class model

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 that can be related to a form in another *Lexical Entry* instance. The former classes include the *Lemma*, *Word Form*, and *Stem*. The latter classes include the *Related Form*. 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 form that a lexeme can take when used in a sentence or a phrase. So, *Word Form* class can manage simple lexemes, compounds and multi-word expressions.

A.3.1.3 Stem class

Stem is a *Form* subclass representing a morph. The aggregation association between a *Lexical Entry* and a *Stem* is ordered. So, *Stem* class manages the sub-lexeme parts.

A.3.1.4 Related Form class

Related Form is a *Form* subclass representing a word form or a morph 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. There is no assumption that the *Related Form* is associated with the *Sense* class in the *Lexical Entry*.

A.3.2 List Of Components class

List Of Components 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 Components* 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 Morphological Pattern and MWE Pattern packages.

A.3.3 Component class

Component is a class representing a reference to a lexical entry for each lexical component aggregated in a *List Of Components* class.

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.
Related Form	writtenForm phoneticForm type	
Component Form		
List Of Components		

B.3 Example of lexeme 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*. The language coding is set for the whole lexical resource using ISO 639-3 as described in Figure B.1[5)].

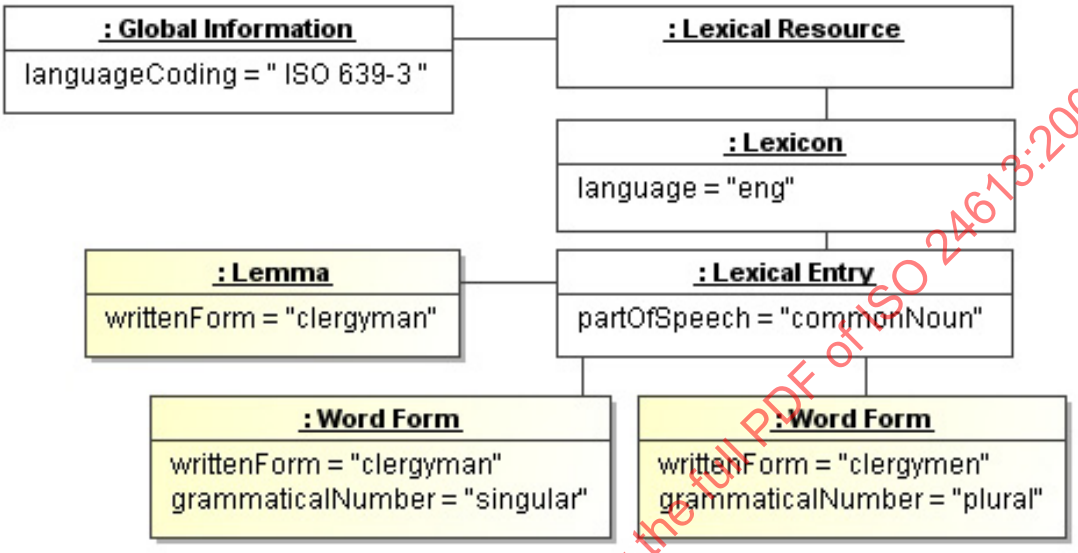

Figure B.1 — Instance diagram for a simple example

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

```
<LexicalResource dtdVersion="16">
   <GlobalInformation>
      <feat att="languageCoding" val="ISO 639-3"/>
   </GlobalInformation>
   <Lexicon>
      <feat att="language" val="eng"/>
      <LexicalEntry>
         <feat att="partOfSpeech" val="commonNoun"/>
         <Lemma>
            <feat att="writtenForm" val="clergyman"/>
         </Lemma>
         <WordForm>
            <feat att="writtenForm" val="clergyman"/>
            <feat att="grammaticalNumber" val="singular"/>
         </WordForm>
         <WordForm>
            <feat att="writtenForm" val="clergymen"/>
            <feat att="grammaticalNumber" val="plural"/>
         </WordForm>
      </LexicalEntry>
   </Lexicon>
</LexicalResource>
```

5) In order to make this figure easier to read, shaded box outlines are used for the instances of the classes defined in the current package. The box outlines of the instances of the classes defined in another package are not shaded.

It is also possible to specify the type of *Word Form* by adding a specific attribute *lexicalType* as in the following instance diagram, see Figure B.2.

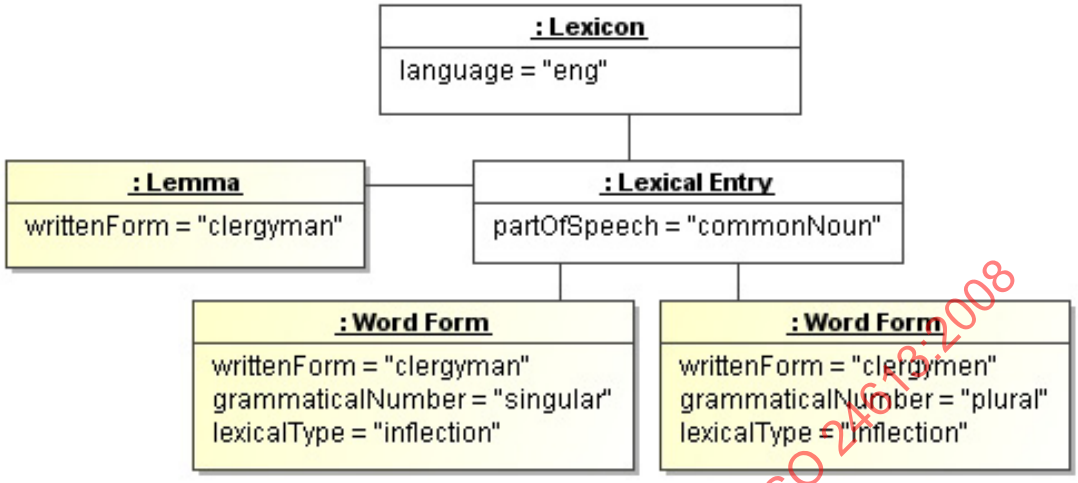

Figure B.2 — Highly specified Word Form example

B.3.2 Example of regional variants

Regional variants can be modeled in English using the *Form Representation* class, with a shared *phonetic form* attribute, as shown in Figure B.3.

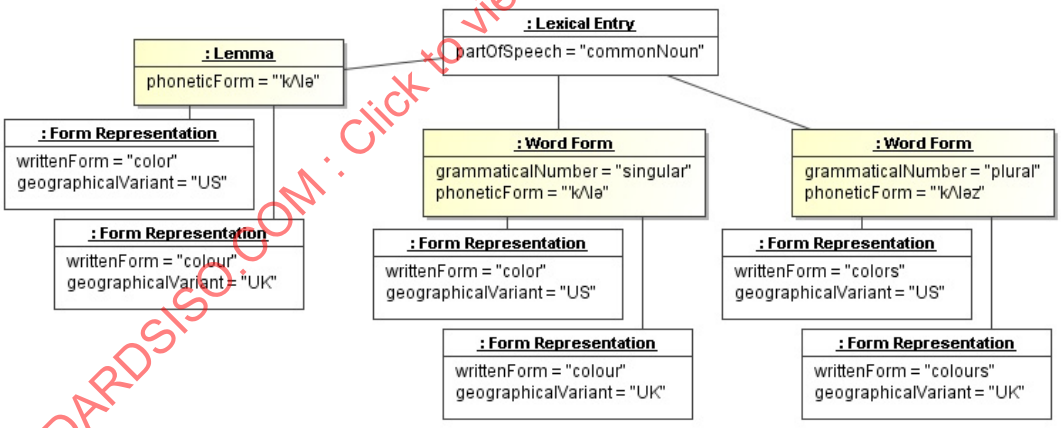

Figure B.3 — Example of regional variants using Form Representation

B.3.3 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 associated with an inflected form that also has three different ways to express the word form, as follows in Figure B.4.

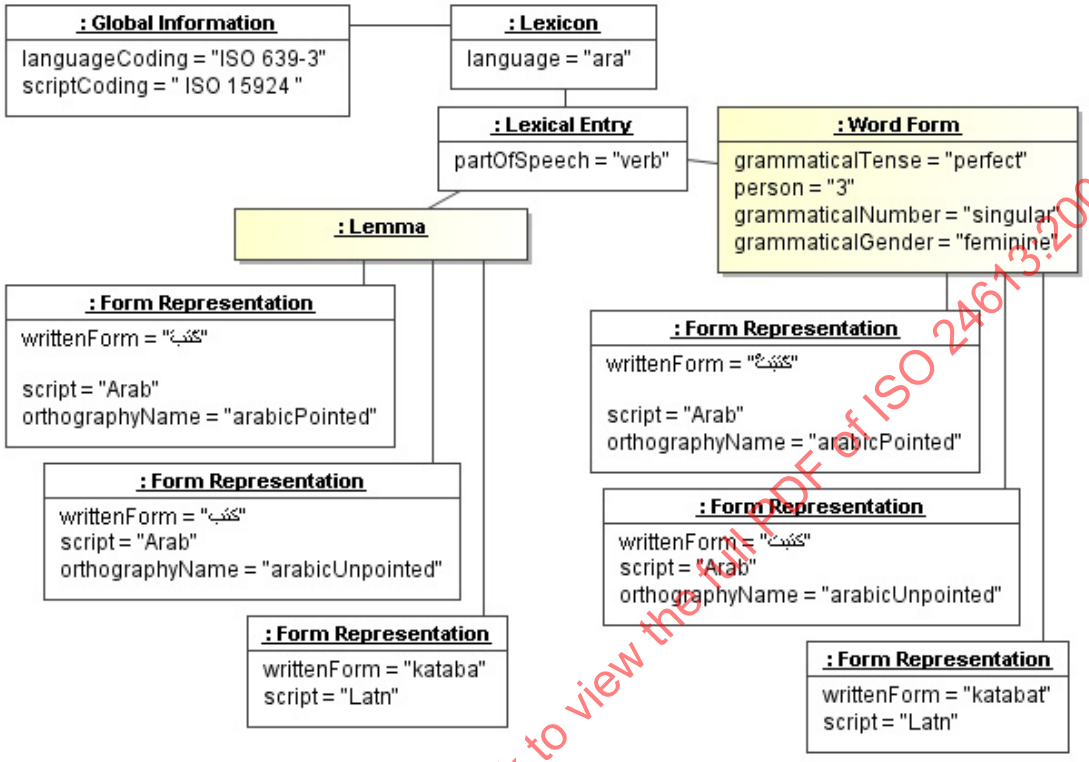

Figure B.4 — Example of multiple scripts and orthographies

It is 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.3.4 Example of multiple scripts, orthographies and variants

The number of *Form Representation* instances may be more important in Japanese where four kinds of writing systems co-exist and combine: hiragana, katakana, kanji and their romanization. A set of variants with the same script name may be combined as in the following example representing *curly hair*, see Figure B.5.

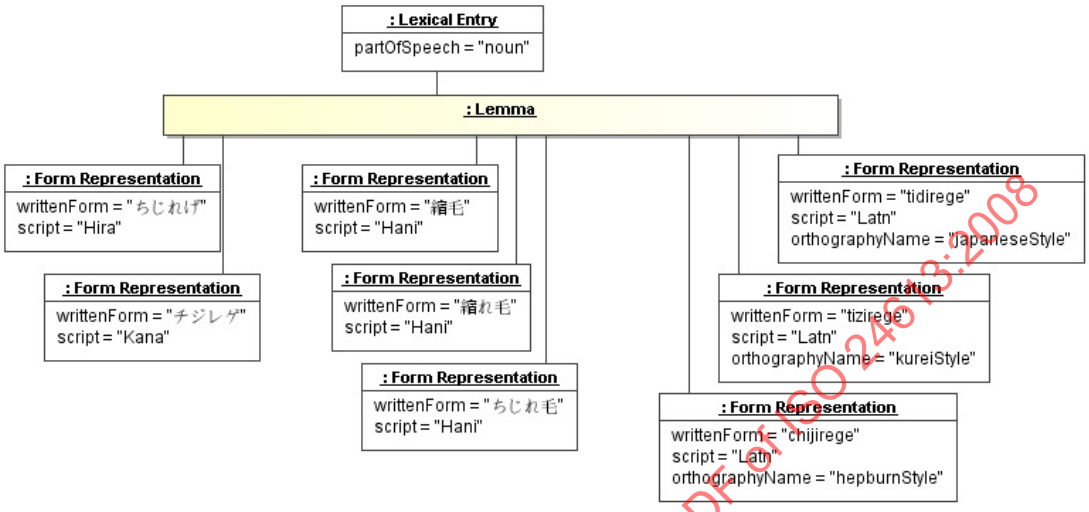

Figure B.5 — Example of multiple scripts orthographies and variants

B.3.5 Example in Chinese

Chinese is an isolating language. Over the years, the Chinese writing system has changed. The movement to simplify the Chinese writing system originated in the 1890s and was extended in the 1950s. The strategy of simplification involves a reduction in the number of strokes of commonly used characters. And at the moment, the two variants are in use. According to ISO 15924, the script code is *Hans* for the simplified variant and *Hant* for the traditional variant.

The following example shows a situation where two different traditional forms are equivalent to one unique simplified form. If the user wants to restrict the description to simplified forms, a single lemma is sufficient, for *behavior on the stage* and *typhoon*, as in the following diagram. The language and script information are global to all lexical entries, thus these attributes are located on the *Lexicon* instance, see Figure B.6.

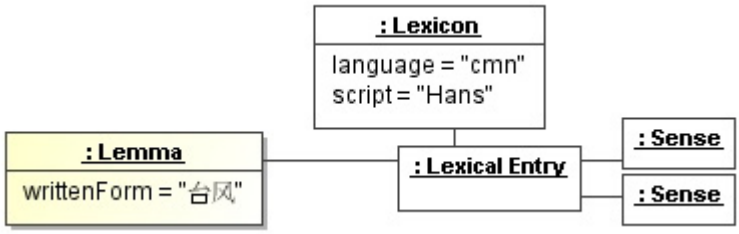

Figure B.6 — Example in simplified Chinese writing system

But if the user wants to describe traditional forms, two *Lexical Entry* instances are required because there are two distinct traditional forms and the meaning of each of these lexemes is different, see Figure B.7.

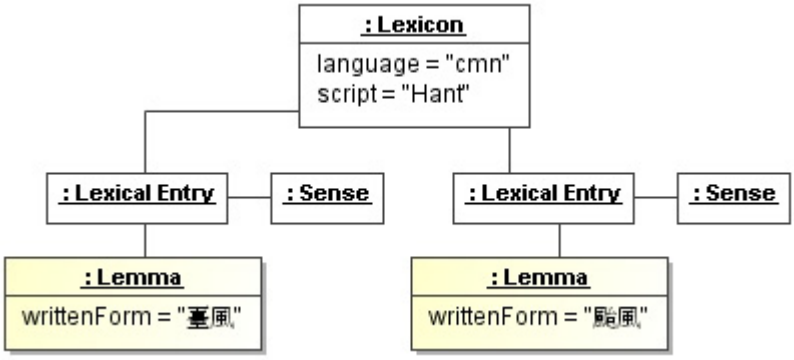

Figure B.7 — Example in traditional Chinese writing system

It is worth noting that if the user wants to mix simplified and traditional forms in the same lexicon, the script attribute cannot be set to the *Lexicon* instance but must be set to each *Form Representation* instance, as in the previous Japanese example.

B.3.6 Example of Arabic root management

In this example, Arabic root is represented by a shared specific *LexicalEntry* instance. The verb *kataba* and the noun *maktabatun* are both associated with the *LexicalEntry* instance *ktb*, see Figure B.8.

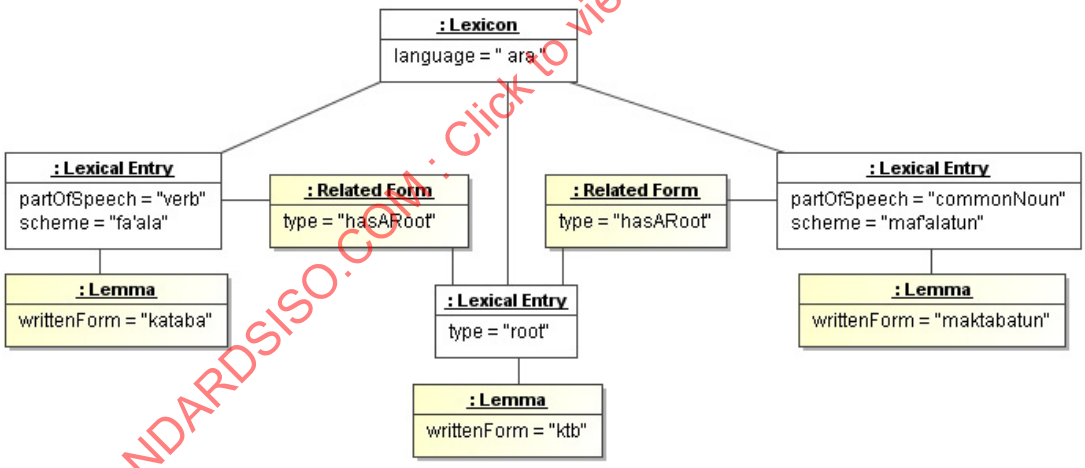

Figure B.8 — Example of Arabic root management

Annex C
(normative)

Machine readable dictionary extension

C.1 General objectives

The machine readable dictionary (MRD) extension provides a metamodel package 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 and the morphological extension, it is designed to interchange data with other LMF extensions where applicable.

C.2 MRD package

C.2.1 Objectives

The objectives of the 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

The MRD extension is organized as presented in Figure C.1.

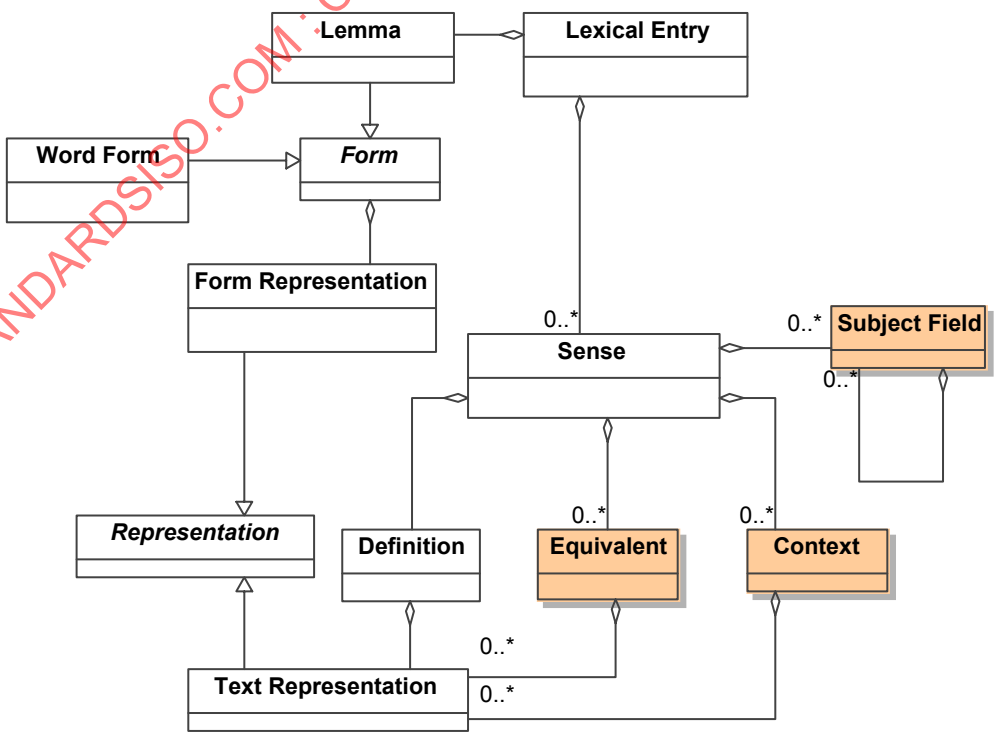

Figure C.1 — MRD class model

C.2.3 Description of the MRD metamodel

The MRD metamodel is based on the NLP morphological extension with the following modifications.

— The MRD package relaxes all constraints on the *Related Form* class. (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 provides additional classes for the *Sense* class.

Certain classes of MRD, such as bilingual dictionaries, generally require a *Sense* class instantiation. Other classes of MRD, such as orthographic dictionaries, may not require a *Sense* class instantiation.

C.2.4 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 lexicon developer to omit the *Equivalent* class from a monolingual dictionary.

C.2.5 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.2.6 Subject Field class

Subject Field is a class representing a text string that provides domain or status information. The *Subject Field* class is in a zero to many aggregate association with the *Sense* class. The *Subject Field* class allows for hierarchical senses in that a *Subject Field* instance may be more specific than another *Subject Field* instance of the same lexical entry.

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 *Text Representation* instances can represent different languages and scripts within the scope of the aggregating class.

Annex D
(informative)

Machine readable dictionary examples

D.1 MRD example

D.1.1 Example of a bilingual MRD with multiple representations

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, see Figure D.1[6)].

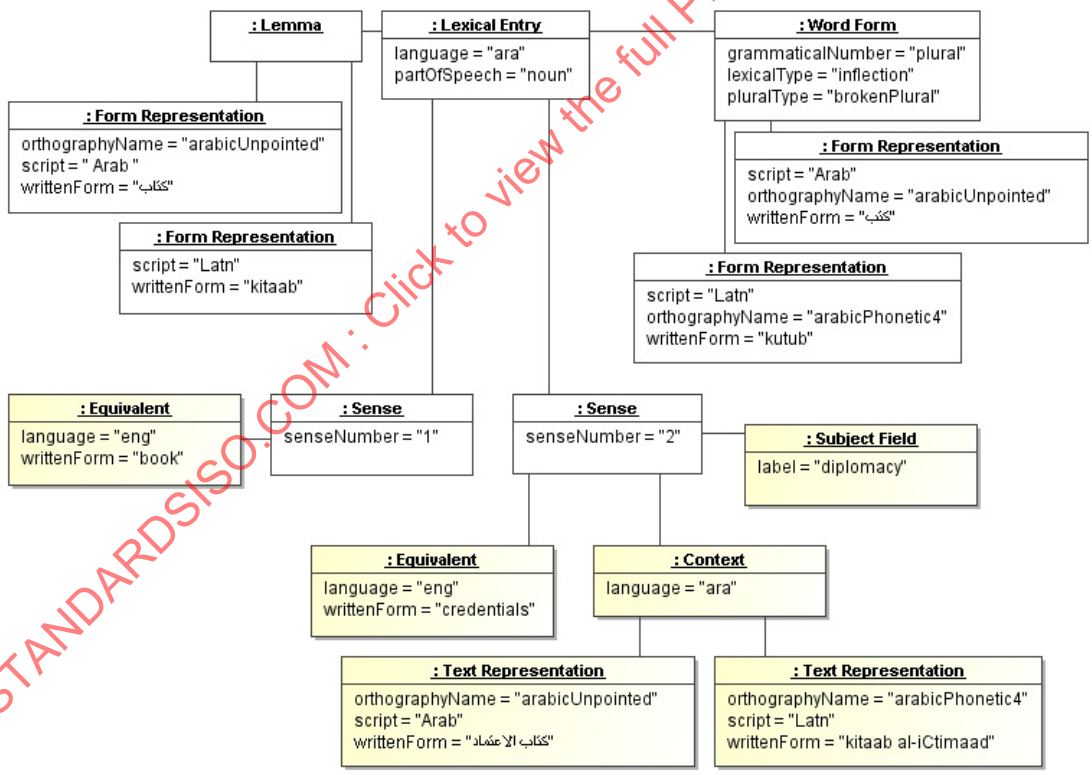

Figure D.1 — Instantiation example for a bilingual MRD

6) In order to make this figure easier to read, shaded box outlines are used for the instances of the classes defined in the current package. The box outlines of the instances of the classes defined in another package are not shaded.

Annex E
(normative)

NLP syntax extension

E.1 Objectives

The purpose of this annex is to describe the properties of a lexeme when combined with other lexemes in a sentence. When recorded in a lexicon, the syntactic properties make up the syntactic description of a Lexical Entry instance.

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

E.2 Class diagram

The NLP syntax extension is organized as described in Figure E.1.

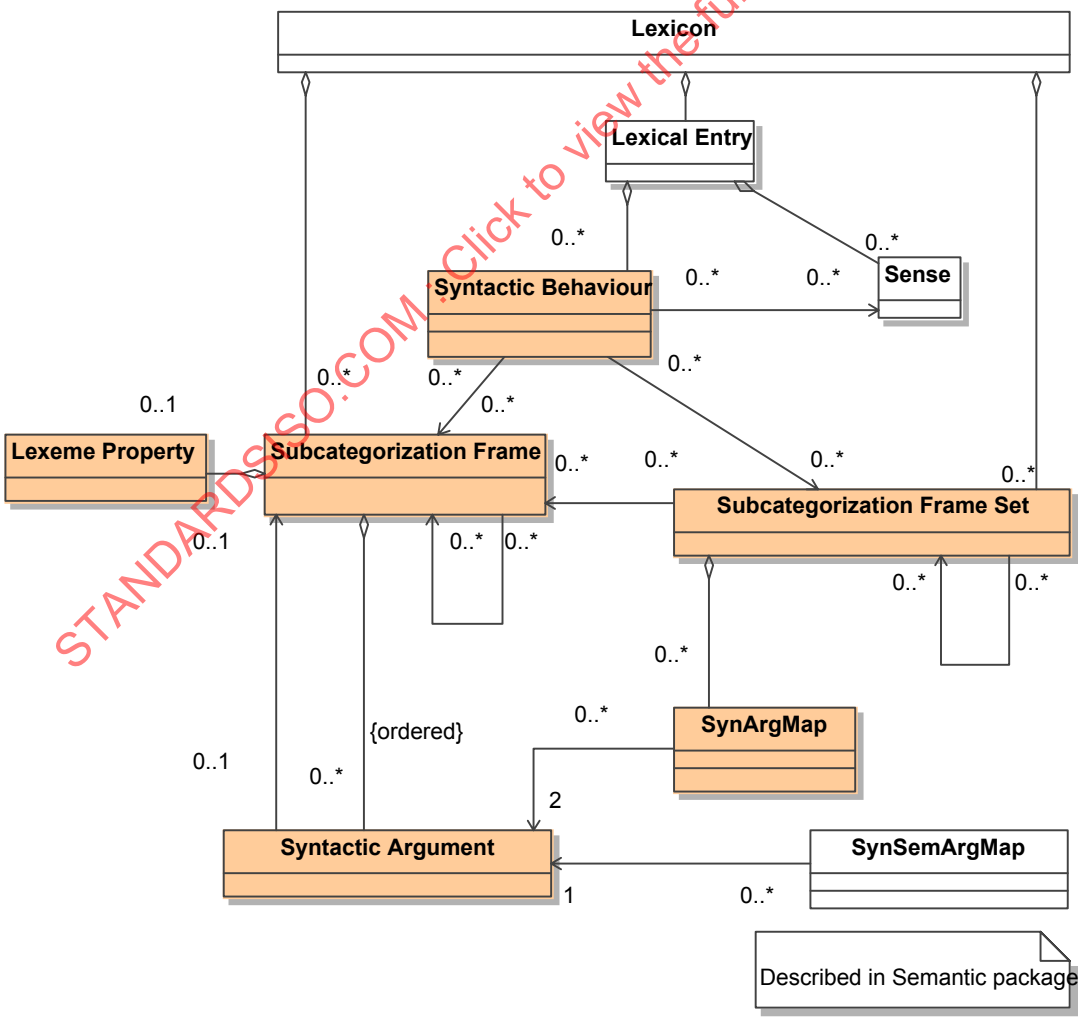

Figure E.1 — Syntactic model

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 lexeme. 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 lexeme 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 *Lexical 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 and 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 lexemes 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 *Syntactic 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 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 lexemes. 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 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 lexeme in the sentence with respect to the syntactic arguments.
Syntactic Argument	syntacticFunction syntacticConstituent introducer label restriction example	The /syntacticFunction/ 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 comment	
SynArgMap	comment	

F.2 Examples of lexeme description

F.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*, see Figure F.1 [7)].

7) In order to make this figure easier to read, the instances of the classes defined in the current package have shaded outlines. The outlines of the instances of the classes defined in another package are not shaded.

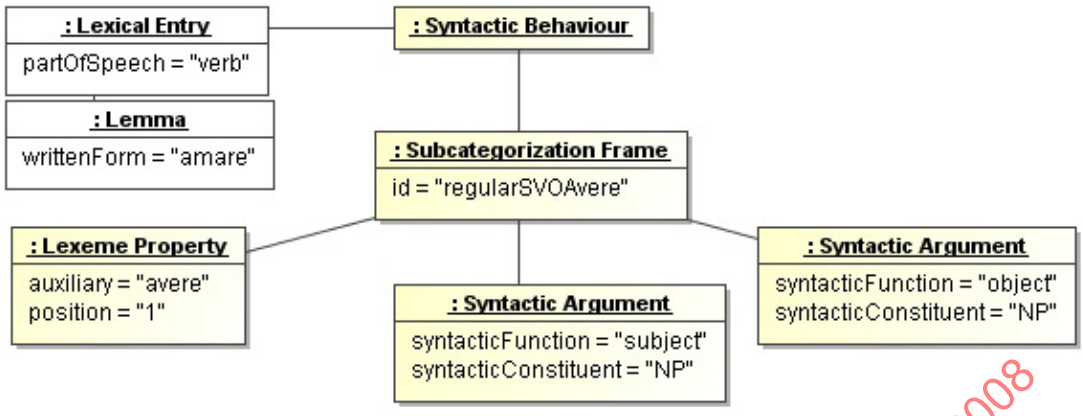

Figure F.1 — Instance diagram in Italian

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

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

F.2.2 Example in English

In English, it is possible to use just one *Subcategorization Frame Set* for certain anticausative 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 Figure F.2.

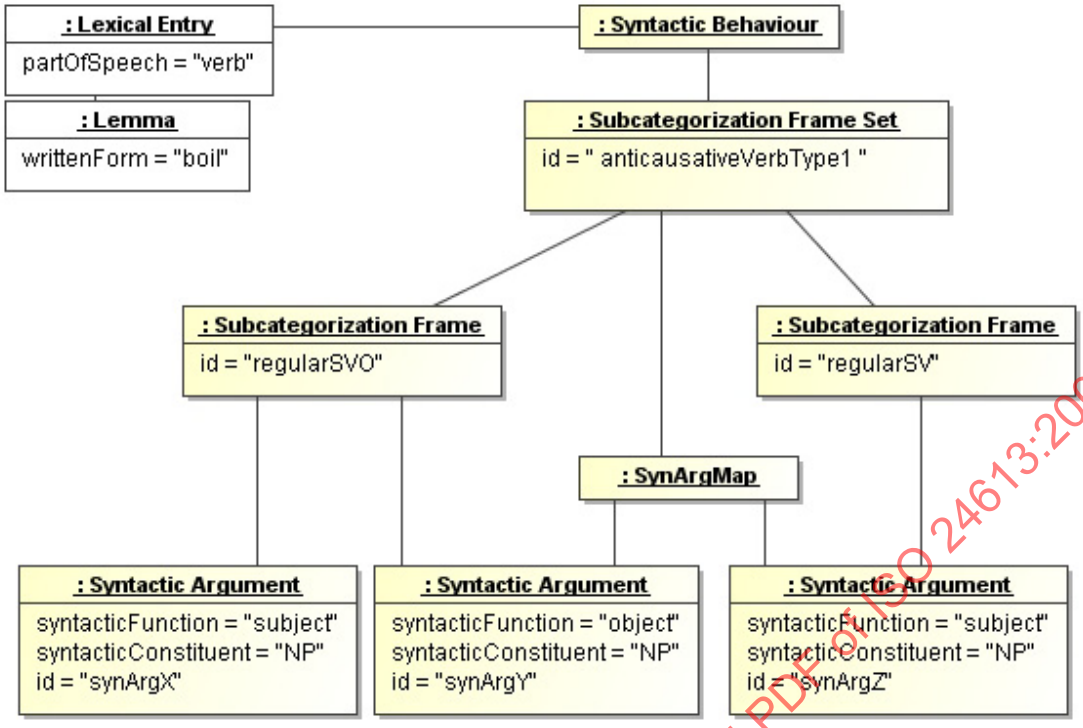

Figure F.2 — Instance diagram in English

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

```
<LexicalEntry>
   <feat att="partOfSpeech" val="verb"/>
   <Lemma>
      <feat att="writtenForm" val="boil"/>
   </Lemma>
   <SyntacticBehaviour subcategorizationFrameSets="anticausativeVerbType1"/>
</LexicalEntry>
<SubcategorizationFrameSet  id= "anticausativeVerbType1"
                            subcategorizationFrames= "regularSVO regularSV">
   <SynArgMap arg1="synArgY" arg2="synArgZ"/>
</SubcategorizationFrameSet>
<SubcategorizationFrame id="regularSVO">
   <SyntacticArgument id="synArgX">
      <feat att="syntacticFunction" val="subject"/>
      <feat att="syntacticConstituent" val="NP"/>
   </SyntacticArgument>
   <SyntacticArgument id="synArgY">
      <feat att="syntacticFunction" val="object"/>
      <feat att="syntacticConstituent" val="NP"/>
   </SyntacticArgument>
</SubcategorizationFrame>
<SubcategorizationFrame id="regularSV">
   <SyntacticArgument id="synArgZ">
      <feat att="syntacticFunction" val="subject"/>
      <feat att="syntacticConstituent" val="NP"/>
   </SyntacticArgument>
</SubcategorizationFrame>
```

Annex G
(normative)

NLP semantics extension

G.1 Objectives

The purpose of this annex 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 annex 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

The NLP semantics extension is organized as described in Figure G.1.

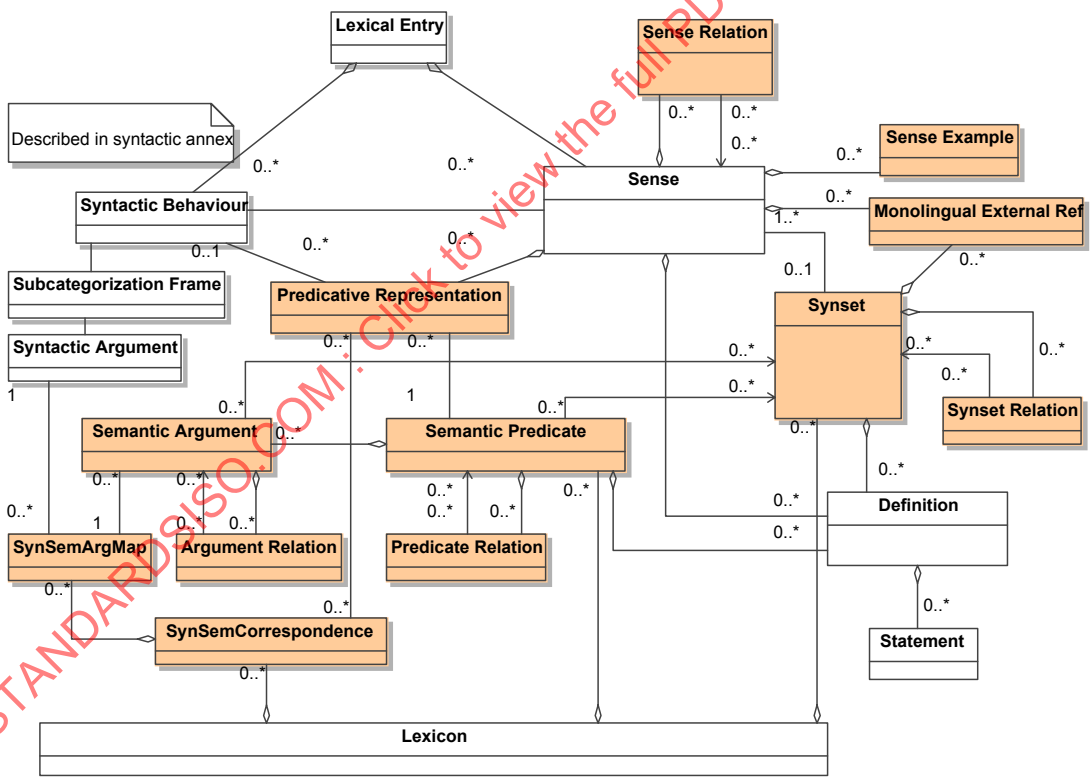

Figure G.1 — Semantic model

G.3 Connection with the core package

The *Sense* class is specified in the core package. The *Sense* class is aggregated in the *Lexical 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 the set of shared meanings 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 oriented relationship between *Synset* instances.

G.4.3 Sense Relation class

Sense Relation is a class representing the oriented 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 *Lexical Entry* for the MWE non-governmental organization (NGO), a *Sense Example* might be Amnesty International.

G.4.5 Semantic Predicate class

Semantic Predicate is a class representing an abstract meaning together with its association with the *Semantic Argument* class. A *Semantic 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 may be typed by means of a relation to one or many *Synset* instances. 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 [29],[30].

G.4.6 Predicative Representation class

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

EXAMPLE In the example given in the *Semantic Predicate* class subclause (G.4.5), 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.4.7 Semantic Argument class

Semantic Argument is a class representing an argument of a given *Semantic Predicate*. A *Semantic Argument* instance may be typed by means of a relation to one or many *Synset* instances.

EXAMPLE In the example given in the *Semantic Predicate* class subclause (G.4.5), the predicate might have two *Semantic Argument* instances: one for the person who buys and one for what is bought.

G.4.8 Argument Relation class

Argument Relation is a class representing an oriented relationship between *Semantic Argument* instances of the same *Predicate* instance.

G.4.9 SynSemArgMap class

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

G.4.10 SynSemCorrespondence class

SynSemCorrespondence is a class representing a set of *SynSemArgMap* instances for a given *Subcategorization Frame* instance.

G.4.11 Predicate Relation class

Predicate Relation is a class representing the oriented 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.

NOTE Guidelines for this class are given in Annex Q and in Reference [24].

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 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 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 *Predicative Representation* of the sense of the noun can be typed as */verbNominalization/*.
Semantic Argument	semanticRole restriction	
Argument Relation		
Semantic Type	label	
SynSemArgMap		
SynSemCorrespondence		
Predicate Relation	label type	
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 lexeme 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 lexemes are linked at semantic level by means of a *Sense Relation* instance in order to represent that *visible* is the contrary of *invisible*, see Figure H.1 [8)].

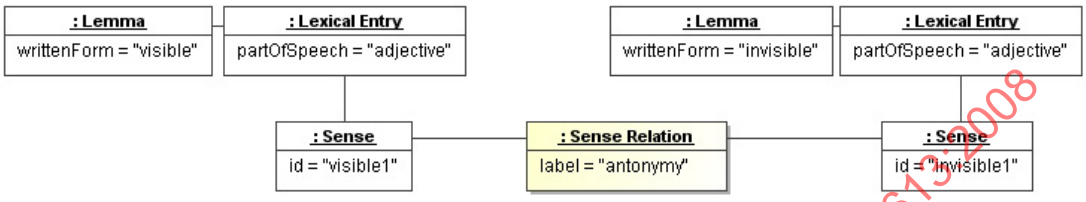

Figure H.1 — Instance diagram for a simple example

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

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

It is worth noting that there is no need for an XML tag in the reverse direction, that is from “invisible1” to “visible1” because this information is already specified from “visible1”.

8) In order to make this figure easier to read, shaded box outlines are used for the instances of the classes defined in the current package. The box outlines of the instances of the classes defined in another package are not shaded.

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 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 this International Standard, the first part may give a *Definition* instance and the two last parts may give two *Statement* instances. The two *Synset* instances are linked by a *Synset Relation* instance that is marked as *substanceHolonym*, see Figure H.2.

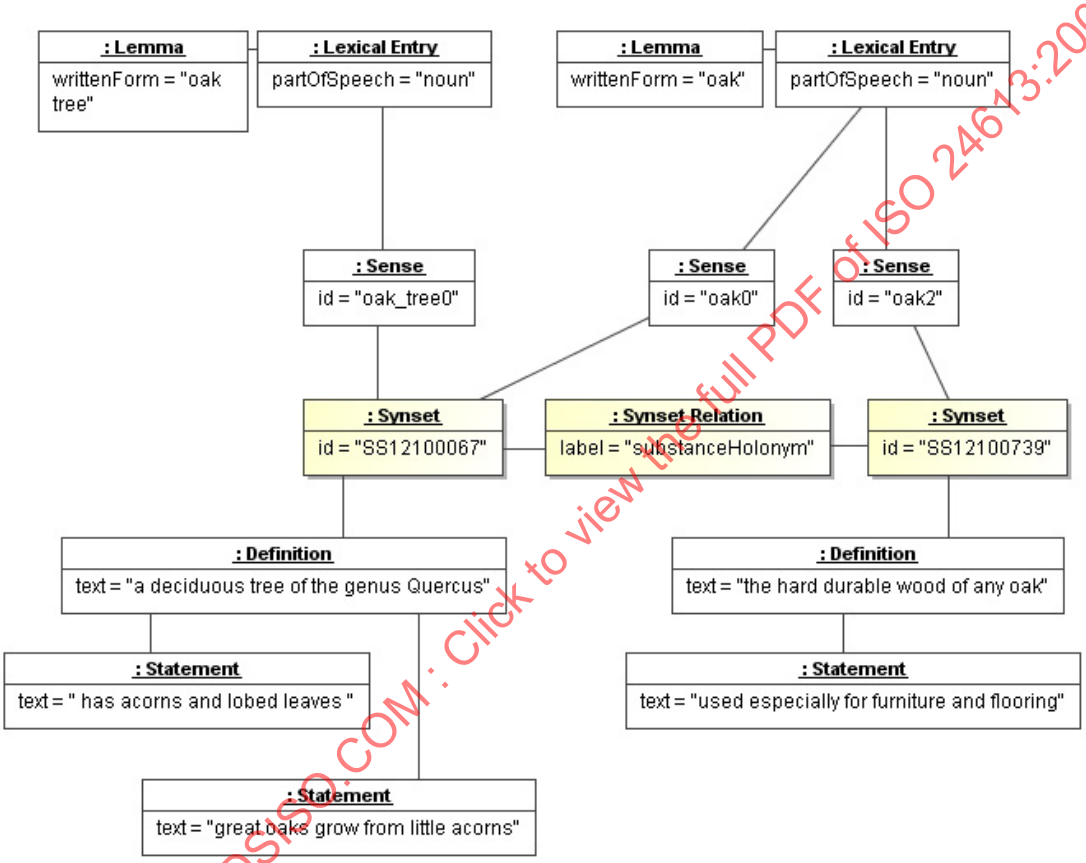

Figure H.2 — Instance diagram for the example taken from WordNet

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

```
<LexicalEntry>
   <feat att="partOfSpeech" val="noun"/>
   <Lemma>
      <feat att="writtenForm" val="oak tree"/>
   </Lemma>
   <Sense id="oak_tree0" synset="SS12100067"/>
</LexicalEntry>
<LexicalEntry>
   <feat att="partOfSpeech" val="noun"/>
   <Lemma>
      <feat att=writtenForm" val="oak"/>
   </Lemma>
   <Sense id="oak0" synset="SS12100067"/>
```

```
    <Sense id="oak2" synset="SS12100739"/>
</LexicalEntry>
<Synset id="12100067">
    <Definition>
        <feat att="text" val="a deciduous tree of the genus Quercus"/>
        <Statement>
            <feat att="text" val="has acorns and lobed leaves"/>
        </Statement>
        <Statement>
            <feat att="text" val="great oaks grow from little acorns"/>
        </Statement>
    </Definition>
    <SynsetRelation targets="SS12100739"
        <feat att="label" val="substanceHolonym"/>
    </SynsetRelation>
</Synset>
<Synset id="SS12100739">
    <Definition>
        <feat att="text" val="the hard durable wood of any oak"/>
        <Statement>
            <feat att="text" val="used especially for furniture and flooring"/>
        </Statement>
    </Definition>
</Synset>
```

H.1.2.3 Example from *Dictionnaire Explicatif et Combinatoire (DEC)*

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 by W”, i.e. “He will help you to overcome this ordeal by intervening.”). This *Lexical Entry* instance yields eight different *Subcategorization Frame instances*, but the following figure 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, see Figure H.3.

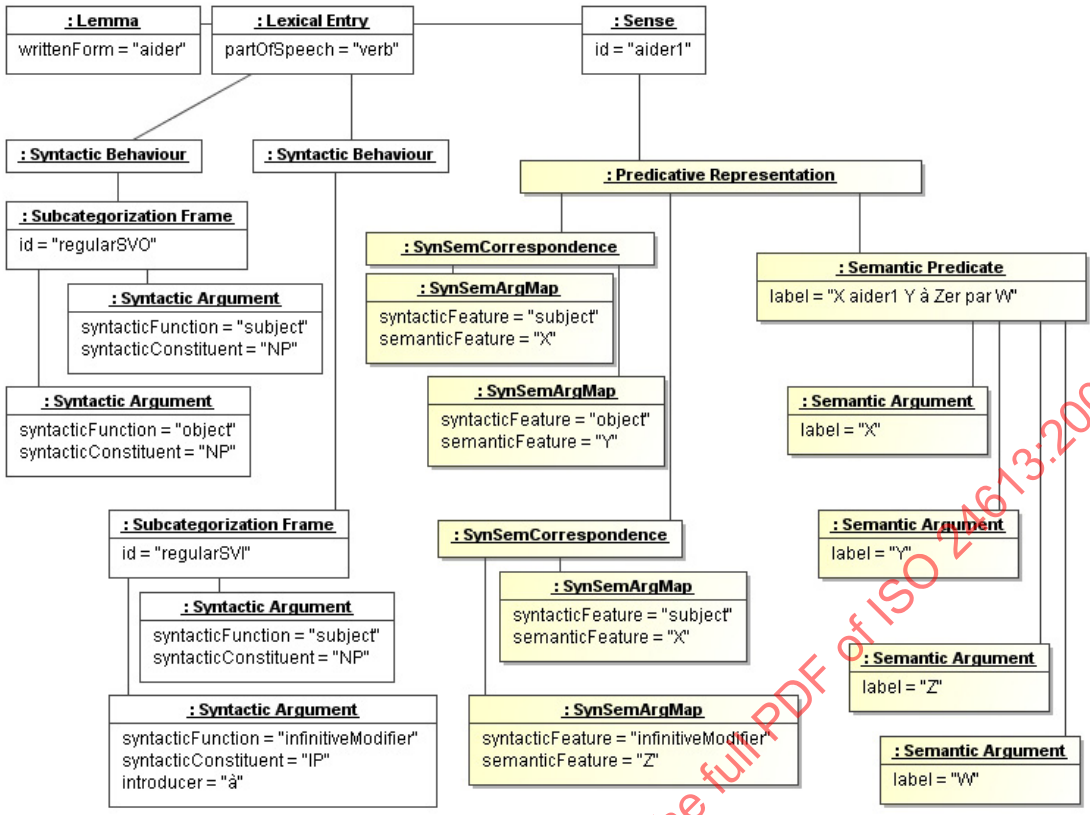

Figure H.3 — Instance diagram for the example taken from the DEC

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

```
<LexicalEntry>
   <feat att="partOfSpeech" val="verb"/>
   <Lemma>
      <feat att="writtenForm" val="aider"/>
   </Lemma>
   <Sense id="aider1">
      <PredicativeRepresentation predicate="P1"
          correspondences="SVO_XY SVI_XZ">
      </PredicativeRepresentation>
   </Sense>
   <SyntacticBehaviour subcategorizationFrames="regularSVO"/>
   <SyntacticBehaviour subcategorizationFrames="regularSVI"/>
</LexicalEntry>
<SynSemCorrespondence id="SVO_XY">
   <SynSemArgMap>
     <feat att="syntacticFeature val="subject"/>
     <feat att="semanticFeature val="X"/>
   </SynSemArgMap>
   <SynSemArgMap>
     <feat att="syntacticFeature val="object"/>
     <feat att="semanticFeature val="Y"/>
   </SynSemArgMap>
</SynSemCorrespondence>
<SynSemCorrespondence id="SVI_XZ">
   <SynSemArgMap>
     <feat att="syntacticFeature val="subject"/>
     <feat att="semanticFeature val="X"/>
```

```
    </SynSemArgMap>
    <SynSemArgMap>
      <feat att="syntacticFeature val=" infinitiveModifier "/>
      <feat att="semanticFeature val="Z"/>
    </SynSemArgMap>
</SynSemCorrespondence>
<SubcategorizationFrame id="regularSVO">
    <SyntacticArgument>
        <feat att="syntacticFunction" val="subject"/>
        <feat att="syntacticConstituent" val="NP"/>
    </SyntacticArgument>
    < SyntacticArgument>
        <feat att="syntacticFunction" val="object"/>
        <feat att="syntacticConstituent" val="NP"/>
    </SyntacticArgument>
</SubcategorizationFrame>
<SubcategorizationFrame id="regularSVI">
    <SyntacticArgument>
        <feat att="syntacticFunction" val="subject"/>
        <feat att="syntacticConstituent" val="NP"/>
    </SyntacticArgument>
    </SyntacticArgument>
        <feat att="syntacticFunction" val="infinitiveModifier"/>
        <feat att="syntacticConstituent" val="IP"/>
        <feat att="introducer" val="à"/>
    </SyntacticArgument>
</SubcategorizationFrame>
<SemanticPredicate id="P1">
    <feat att="label" val="X aider1 Y à Z par W"/>
    <SemanticArgument>
        <feat att="label" val="X"/>
    </SemanticArgument>
    <SemanticArgument>
        <feat att="label" val="Y"/>
    </SemanticArgument>
    <SemanticArgument>
        <feat att="label" val="Z"/>
    </SemanticArgument>
    <SemanticArgument>
        <feat att="label" val="W"/>
    </SemanticArgument>
</SemanticPredicate>
```

H.1.2.4 Example of homonymy

The following example presents two different lexical entries with the same written form.

Two *Lexical Entry* instances are created: one for the verb *to book* and one for the noun *book*. Each of them is associated with a distinct *Lemma* instance. Each *Lexical Entry* instance may be completed by descriptors related to the morphology of the lexemes, and these descriptors are completely different from each other. The two *Lexical Entry* instances are linked at semantic level by means of a *Sense Relation* instance that is labeled *homonym*, see Figure H.4.

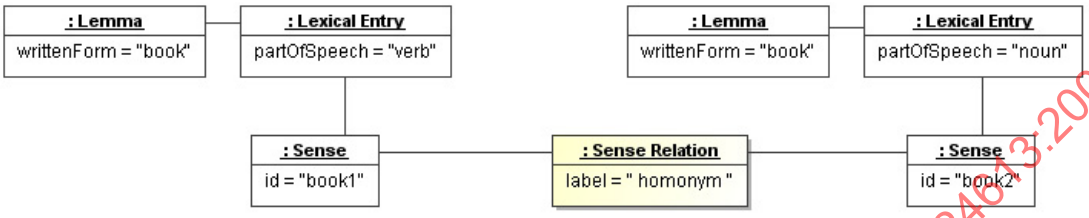

Figure H.4 — Example of homonymy

The notion of homonymy is represented as a relation, not as a structure. In other terms, the *Lemma* instance is not conceptually shared between lexical entries. This does not imply that a physical implementation will necessarily duplicate the character strings in the database but this mechanism is considered as a data compression mechanism, not as a conceptual consideration.

It is worth noting that the criterion that rules the decision about having one or two lexemes is not specified in this current standard: it is up to the lexicon manager to decide to split or to share the lexical information.

Annex I
(normative)

NLP multilingual notations extension

I.1 Objectives

The purpose of this annex 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

The multilingual notations extension is organized as described in Figure I.1.

I.4 Options

I.4.1 General

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 as given in I.4.2 through I.4.6.

I.4.2 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 lexemes 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 lexemes: *paint, lacquer, varnish, shellac*, etc. In this case, the German to English direction involves diversification and the English to German direction involves neutralization.

I.4.3 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 $L_1 - A = L_2 - A$, 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.4.4 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.4.5 Representation of similar languages

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

I.4.6 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.

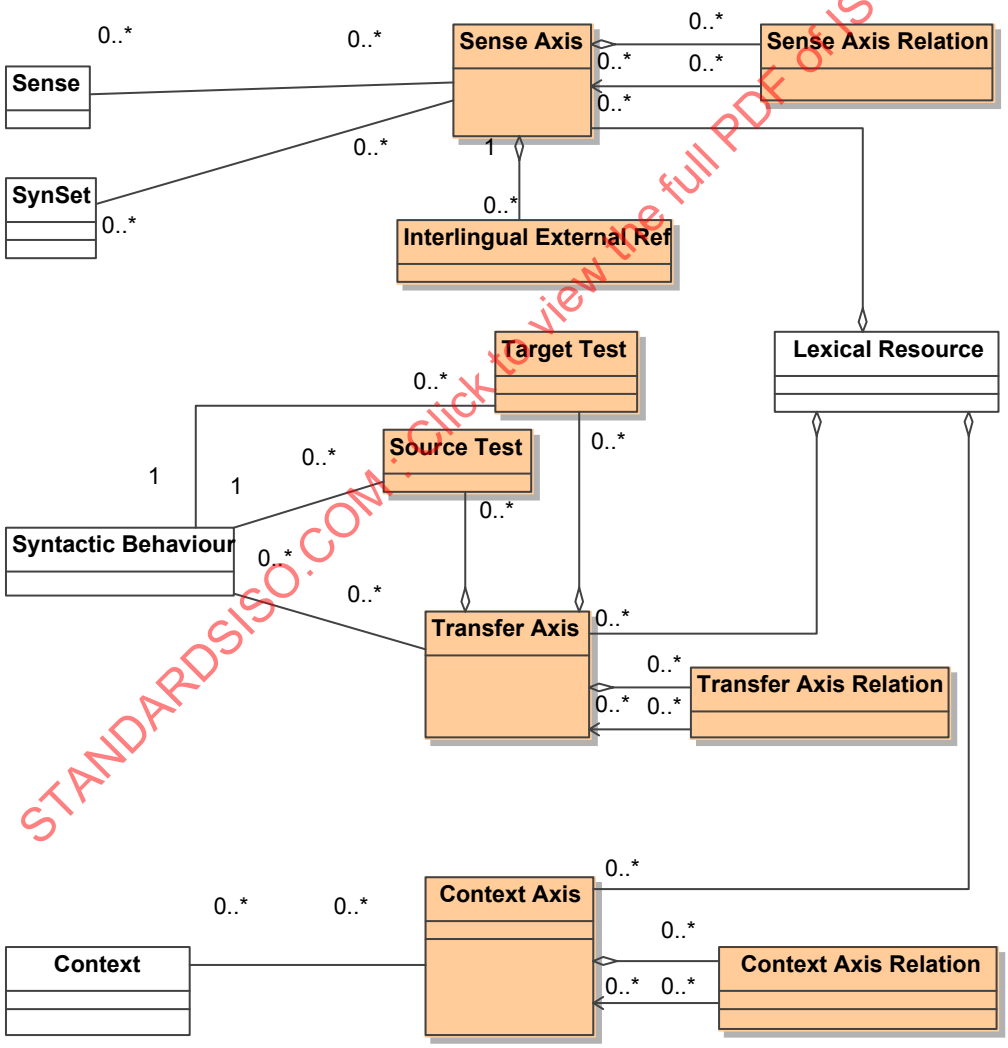

Figure I.1 — Multilingual notations model

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;

— Context Axis.

I.5.1 Sense Axis class

Sense Axis is a class representing the relationship between different closely related senses in different languages and implements an approach based on the interlingual pivot. The purpose is to describe the translation of lexemes 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.

NOTE Guidelines are given in Reference [24].

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 Context Axis class

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

I.5.9 Context Axis Relation class

Context Axis Relation is a class representing the relationship between two *Context 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	It's worth noting that there is no constraint on the types of lexeme that are linked by a *Sense Axis* instance. For instance, a single lexeme 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 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	
Context 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 lexeme 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 bottom is not linked directly to any English sense because this notion does not exist in English, see Figure J.1 [9].

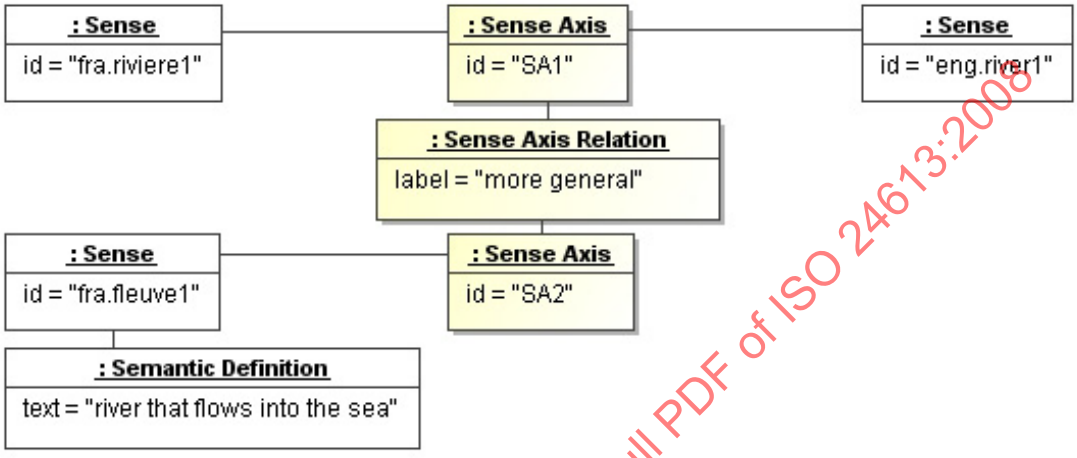

Figure J.1 — Instance diagram for "river"

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


9) In order to make this figure easier to read, shaded box outlines are used for the instances of the classes defined in the current package. The box outlines of the instances of the classes defined in another package are not shaded.

J.2.2 Example of equivalence at transfer level

This example shows how to use the *Transfer Axis Relation* instance to relate 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, see Figure J.2.

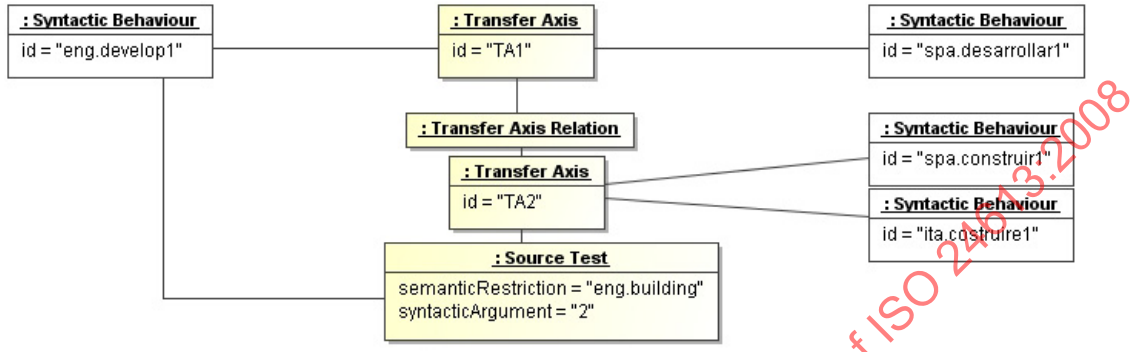

Figure J.2 — Instance diagram for "develop"

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

Annex K
(normative)

NLP morphological patterns 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 *Morphological Pattern* instance. The forms documented in the lexical entry may include the root, stem, or stem allomorphs. These forms are unique to a specific lexical entry.

The lexical information documented in the *Morphological Pattern* 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 analyse and generate the forms. This extension is not intended to meet all the needs for morphological lexicons; however, the LMF core package and this extension provide the basis for developing additional morphology extensions.

K.2 Class diagram

The Morphological Pattern extension is organized as in Figure K.1.

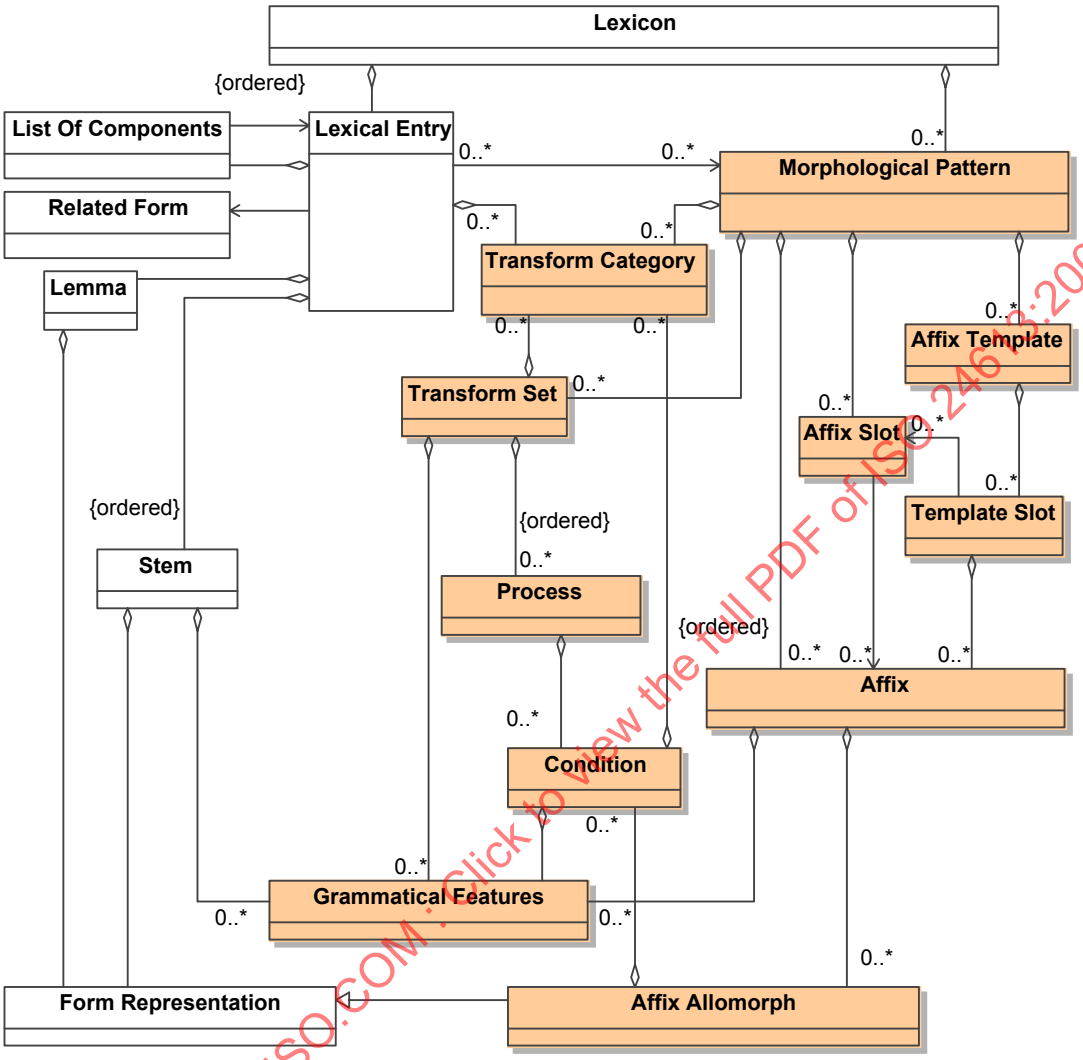

Figure K.1 — Morphological Pattern model

K.3 Description of the Morphological Pattern model

K.3.1 Introduction

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

K.3.2 Morphological Pattern class

Morphological Pattern is a class representing the structure of affixes and/or a set of processes that describe a pattern of morphological changes for a given language. *Morphological Pattern* supports the modelling of inflectional, derivational, agglutinative and compounding patterns and may be subtyped accordingly, e.g. patternType='inflectional'. Class constraints, associated types of morphological features, and specific pattern

choices enable the *Morphological Pattern* class to describe a variety of different methods used by diverse language groups for encoding morphological changes. For example, a *Morphological Pattern* may be constrained on a part of speech to manage inflectional patterns for Indo-European languages, or be used to manage a reduplication process that models verb intensification for Thai with no need to constrain on a part of speech. A *Morphological Paradigm* can manage an *Affix* class indirectly through the *Affix Slot* class, or can manage the *Affix* class directly, but not both in the same model. The *Morphological Paradigm* also manages a *Transform Set* class that supports the modelling of linguistic rules and processes, and an *Affix Template* class that supports the management of affix patterns (e.g. ordered sets of suffixes and/or prefixes). The *Transform Set* class associations, *Affix Slot/Affix* associations, and *Affix Template* are not mutually exclusive. However, the use of the *Affix Template* class may provide additional constraints on the use of the *Affix* and *Affix Slot* classes.

K.3.3 Transform Set class

Transform Set is a class representing the association between *Process* class and *Grammatical Features* class that further defines the scope or range of the managed pattern. *Transform Set* is in a zero to many aggregation with the *Morphological Pattern*.

K.3.4 Process class

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

K.3.5 Grammatical Features class

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

K.3.6 Condition class

Condition is a class representing the conditions that determine or constrain the usage of a *Process* or *Affix Allomorph* instance.

EXAMPLE A *Condition* instance may describe the phonetic environments that determine the choice of affix allomorphs.

K.3.7 Affix class

Affix is a class representing an affix, that is a word form or morpheme that is qualified by a set of grammatical features and is required for analysing or generating word forms. An *Affix* class manages one or more affix allomorphs through aggregate association with the *Affix Allomorph* class.

K.3.8 Affix template class

Affix Template is a class managing a pattern of ordered affixes for inflectional, derivational, or agglutinative morphology indirectly through a *Template Slot* class. *Affix Template* attributes may describe the directionality of the affixes, the number of affixes in an ordered set, and any special conditions applicable to the affix pattern (e.g. optionality for specific slots).

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.3.9 Template slot class

Template Slot is a class representing a set of affixes that can be attached to an ordered position in the *Affix Template* class. A *Template Slot* can manage an *Affix* class indirectly through the *Affix Slot* class, or can manage the *Affix* class directly, but not both in the same model. *Affix Slot* attributes may describe the type of

affix (e.g. suffix, circumfix), the rank of the affix in an ordered set, the number of affixes in an ordered set, and any special conditions applicable to the affix (e.g. the morphological functions shared by the affixes that occupy the slot).

K.3.10 Affix Slot class

Affix Slot is a class referencing a set of affixes that attach to the same position relative to a stem through the *Template Slot* class. The set of affixes represents a subset of the affixes managed directly or indirectly by a *Morphological Paradigm.* An affix may be referenced by one or more *Affix Slot* class objects.

NOTE Managing affixes through a directed association from the *Affix Slot* class reduces the need to instantiate redundant affix objects when implementing LMF in data models.

K.3.11 Affix Allomorph class

Affix Allomorph is a generalization of the *Form Representation* class representing allomorphs of the canonical affix form in all scripts and representations. An *Affix Allomorph* is associated with *Condition* class instances that describe the phonological environment or other conditions (e.g. stem allomorph boundary) that resulted in the production of the allomorph.

K.3.12 Transform Category class

Transform Category is a class representing attributes that constrain or describe sets of features needed to manage morphological change.

EXAMPLE Conjugation classes in Spanish or stem groups in Arabic.

Annex L
(informative)

NLP morphological patterns examples

L.1 Absence versus presence of morphological patterns in a lexicon

Not all lexicons utilize the morphological pattern approach. Theoretically, it would be possible to list all forms in a lexicon, but the use of morphological patterns 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, or derived form is focused on a specific exemplary element instead of being spread throughout a great number of elements.

L.2 Example of class adornment

Classes may be adorned with the following attributes:

Class name	Example of attributes	Comment
Morphological Pattern	id comment example partOfSpeech patternType	A *Morphological Pattern* instance is designed to be shared and referred, thus it holds an identifier. A *Morphological Pattern* instance cannot be used for two different parts of speech (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 location agreement affix transformType	
Affix	writtenForm type	The type may be specified for instance with values like /prefix/ or /suffix/.
Affix Template	type	The type may be specified for instance with values like /prefix/ or /suffix/.
Affix Slot	id	
Template Slot	label position required	The /position/ specifies where the affix is to be set in the word form.
Affix Allomorph	writtenForm	
Transform Category	id comment	A *Transform Category* instance is designed to be shared and referred, thus it holds an identifier.
Grammatical Features	grammaticalNumber grammaticalGender grammaticalTense person	

L.3 Examples of lexeme description

L.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, stem-based approach [11], [12],[13], [14], [15], [17].

The following material deals with

— examples of inflection, beginning with simple phenomena and ending with more complex ones,

— examples of agglutination,

— example of derivation, and

— examples of composition.

Stem, *Affix*, *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.3.2 Inflection using an underspecified morphological pattern instance

In this example, the lemma *clergyman* is declared as conforming to the *Morphological Pattern* “asMan”. This *Morphological Pattern* instance has a name but is not analytically described within the lexicon, see Figure L.1 [10]. In other terms, the *Morphological Pattern* instance is considered as a mark. And for instance, the *Morphological Pattern* may be implemented by an external opaque algorithm.

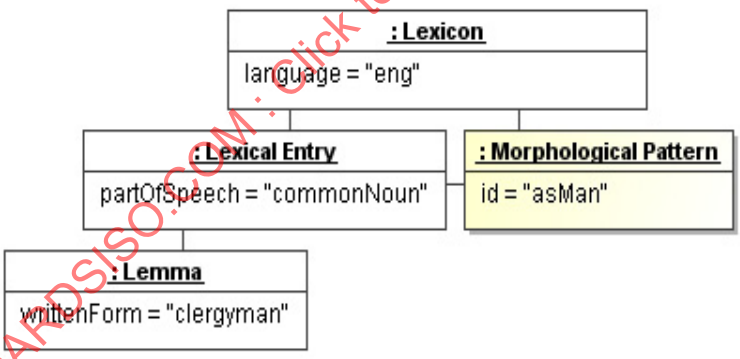

Figure L.1 — Inflection using an underspecified morphological pattern

10) In order to make this figure easier to read, shaded box outlines are used for the instances of the classes defined in the current package. The box outlines of the instances of the classes defined in another package are not shaded.

L.3.3 Inflection using the Transform Set class

This example implements a traditional lemma-based inflection using the English noun “table”. This lexeme is considered to be inflected according to the morphological pattern “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, see Figure L.2.

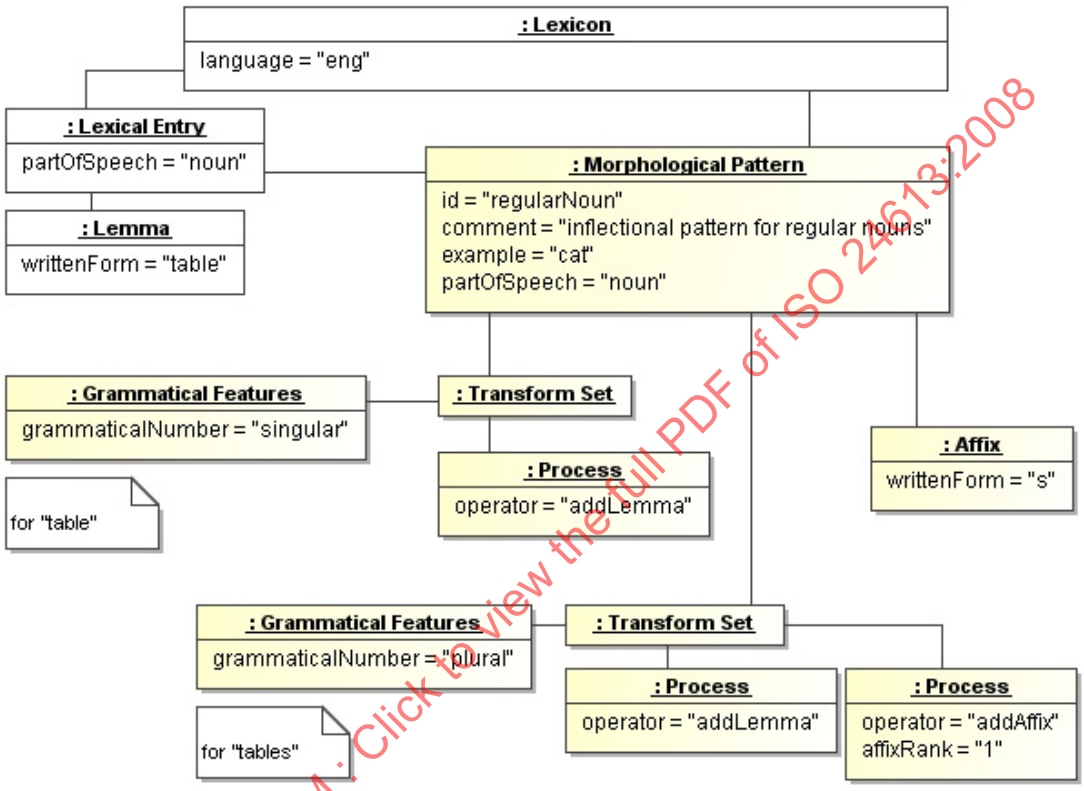

Figure L.2 — Inflection using Transform Set and Affix classes

It is possible to adopt an item and process approach in using an operation that codes the addition of “s” [28]. In this case, the use of an *Affix* instance is not needed, as in Figure L.3.

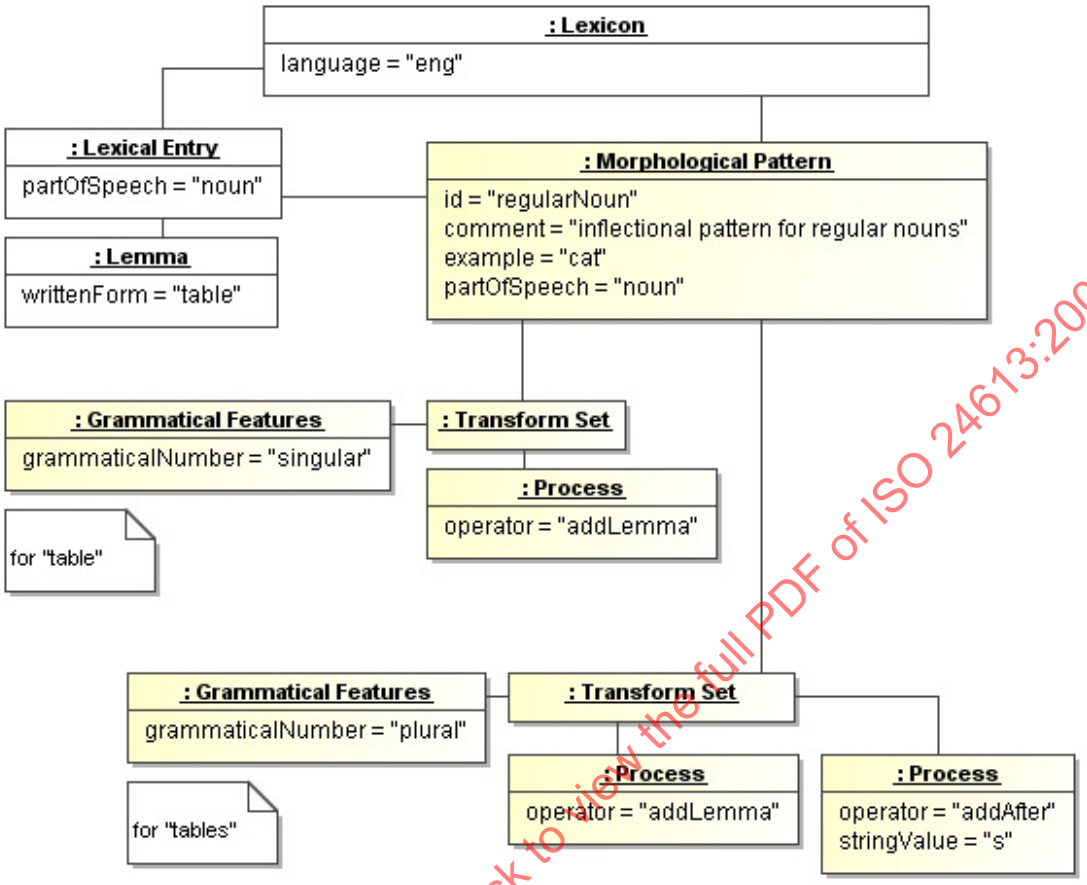

Figure L.3 — Inflection using a Transform Set class

L.3.4 Agglutination using the Affix Template class

This example uses an item and arrangement approach to implement a simplified Turkish verb conjugation. In the following diagram, the *Affix Template* manages a pattern of agglutinative affixes. Each *Template Slot* instance represents a different verbal aspect. This example shows the suffix for the past tense, using the *Affix* to represent the general word form and an associated *Grammatical Features* instance to indicate past tense. The *Affix Allomorph* represents the variant word forms (allomorphs) that are produced in different phonetic environments. In Figure L.4, the *Condition* object represents the left and right environments using regular expressions as values.

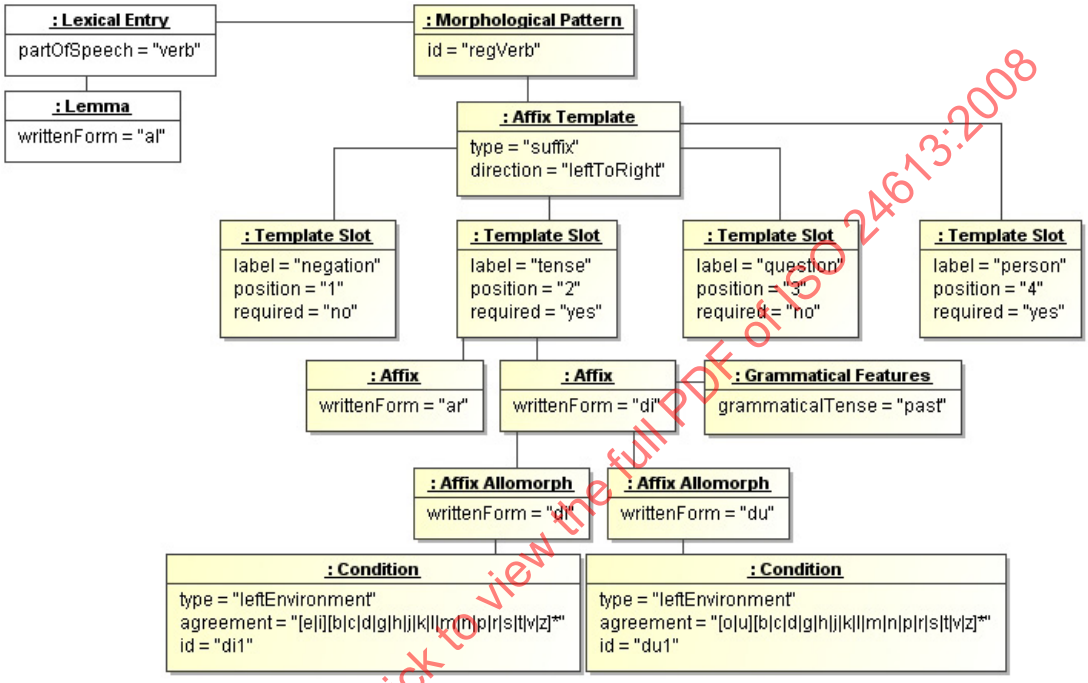

Figure L.4 — Inflection using a Transform Set class

L.3.5 Inflection using an Affix Template class

This example, as shown in Figure L.5, uses an item and arrangement approach to implement a simplified Bangla verb conjugation using an *Affix Template* to manage a pattern of inflectional affixes. This model differs significantly from the model for Turkish agglutination (as presented previously), where the *Affix Template* object managed an ordered set of *Template Slot* instances representing different verbal aspects. In this example, several *Affix Template* instances are used to represent different inflectional suffix patterns. The structure of the classes managing the sets of affixes is also more complex. For example, the *Affix Template* representing participial tenses references two *Template Slot* instances, one managing the perfect participial suffix and a second managing a set of verb suffixes in the perfective tenses. In Bangla, a subset of the simple verb suffixes also serves as components in the perfective verb tenses. In order to reduce the presence of redundant *Affix* objects in an implementation, the *Template Slot* instances manage the affixes indirectly through *Affix Slot* instances. The Affix Slot instances, in turn, reference shared *Affix* objects. The purpose of this design option is illustrated through the case of the 1st person present imperfect suffix shown in the diagram. This suffix form is used for the imperfect tense and as a component in the present perfect tense. Because the affix allomorphs in the different tenses have different phonetic environments, the *Condition* instances may reference sets of *Grammatical Features* instances as relevant constraints.

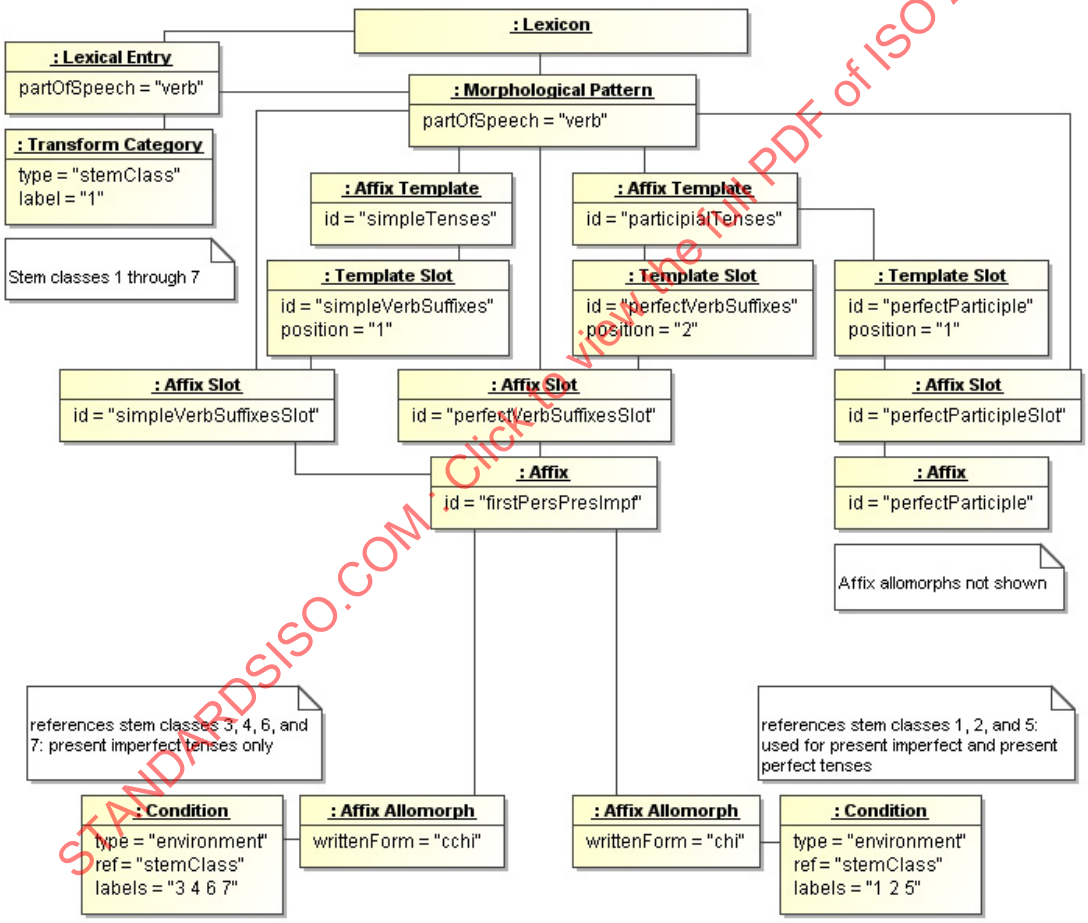

Figure L.5 — Inflection using Affix Template class

L.3.6 Inflection using a Transform Category class as an allomorphic condition

The following Figure L.6 deals with a simple example of a Spanish verb conjugation. *Transform Category* instances are used as a condition for organizing affix allomorphs and for associating categories of affix allomorphs with categories of stem allomorphs. The *Lexical Entry* is associated with two *Transform Category* instances, one of type “stemClass” and the other of type “conjugationClass”. Each *Affix* instance is associated directly with the *Morphological Paradigm* and is qualified by one or more *Grammatical Features* instances. The structure of the Spanish language is a key factor supporting this design approach, which is much less complex than the approach used for modeling Turkish and Bangla (as shown in previous examples). The phonetic value of the *Affix Allomorph* instance is determined by the value of the conjugation class, a subtype of the *Transform Category*. The association of an affix allomorph with a specific stem is managed through reference to a specific stem class, a subtype of *Transform Category*, and the order of the relevant stem in that specific stem class. In this example, the association between the affix /es/ and the stem /pesque/ is managed indirectly through the *Transform Category* instances. That is, one or more *Lexical Entry* instances may be associated with both the /ar/ conjugation class and the /buscar/ stem class. In each case, the second order *Stem* instance is associated with the affix allomorph /es/. This example also shows an alternative method for associating stem allomorphs with affix allomorphs by associating each with a *Grammatical Features* instance.

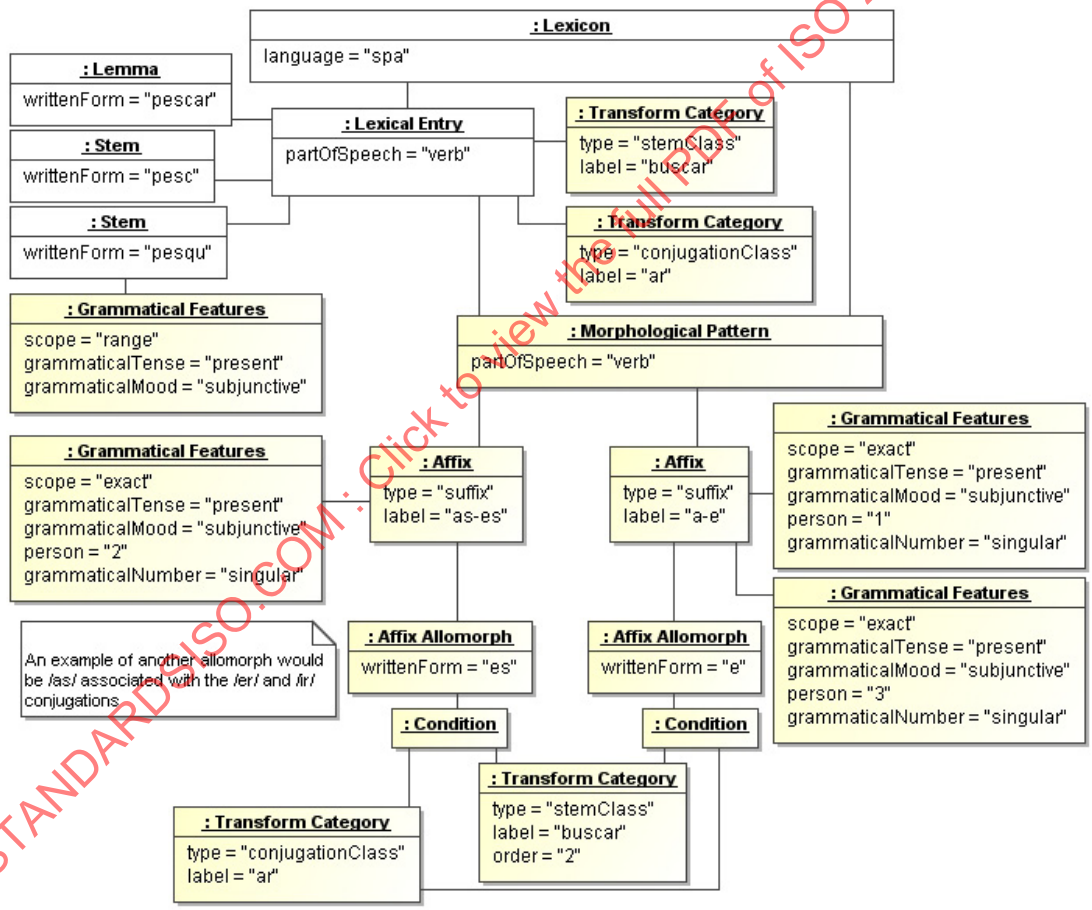

Figure L.6 — Inflection using a Transform Category class

L.3.7 Inflection based on allomorphy using Process classes

Frequently, lexicons do not contain examples of all the stem allomorphs needed to develop an implementation for intensional morphology. In such cases, it may be necessary to produce the stem allomorphs using phonetic rules and environmental conditions. The following figure illustrates one approach for producing stem allomorphs for Modern Standard Arabic in a phonetic transcription. In this example, the *Transform Set* instance manages three processes, each of which produces a different stem allomorph. The *Transform Set* instance is qualified by a *Transform Category* of subtype “stemClass” with a value of “defective3”. Each process is triggered by a specific condition, which in this case is determined by reference to a set of *Grammatical Features* instances. This process can be applied to all stems contained in *Lexical Entry* objects that are qualified by an association with a *Transform Category* instance of subtype “stemClass” with the value of “defective3”. This example, as shown in Figure L.7, could be used in conjunction with other design approaches to both produce stem allomorphs and associate those stem allomorphs with relevant affix allomorphs.

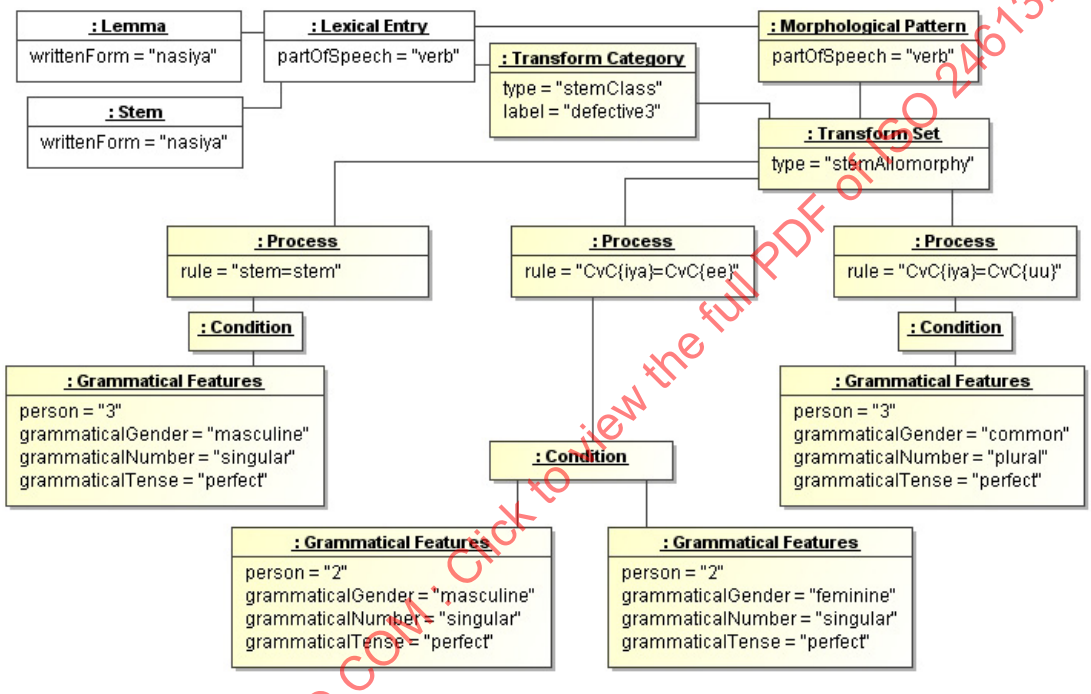

Figure L.7 — Inflection based on allomorphy using Process classes

L.3.8 Complex inflection: verbal forms in Tagalog

Verbs in Tagalog are difficult to describe by means of a pure item and arrangement approach. This particular example shows how to form the future tense by taking the first consonant, adding the first vowel and adding these letters to the left side of the lemma, see Figure L.8.

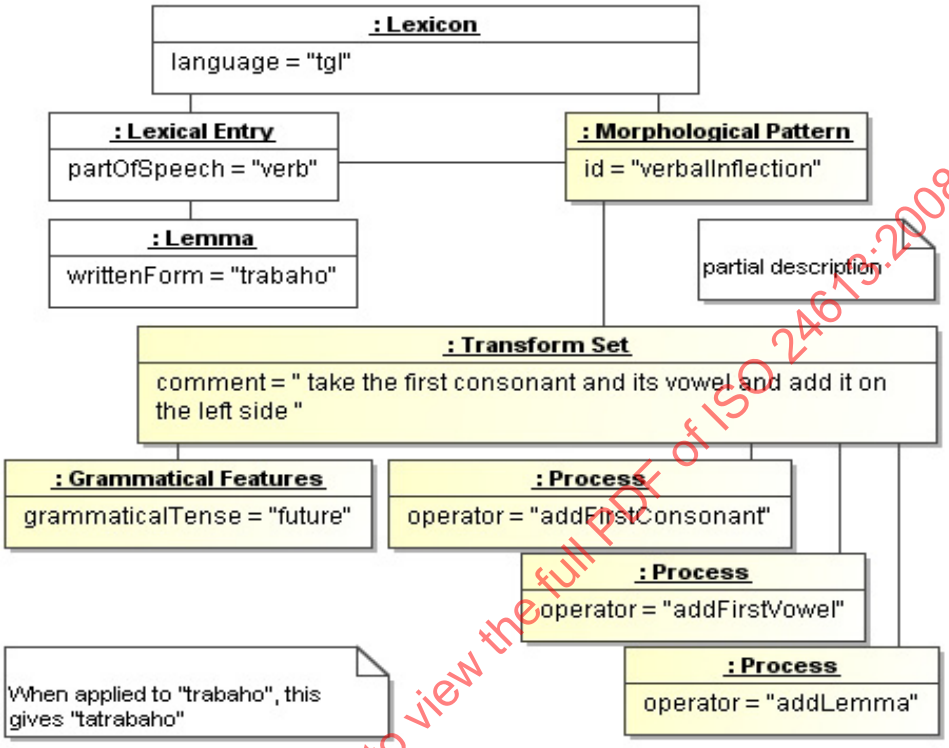

Figure L.8 — Verbal forms in Tagalog

L.3.9 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 subclauses deal with inflectional languages. On the contrary, the following example is about Hungarian, an agglutinative language. In Hungarian, agglutination is governed 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 [27].

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. These exceptions must be recorded in *Morphological Pattern* instances. For example, imported lexemes 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, see Figure L.9.

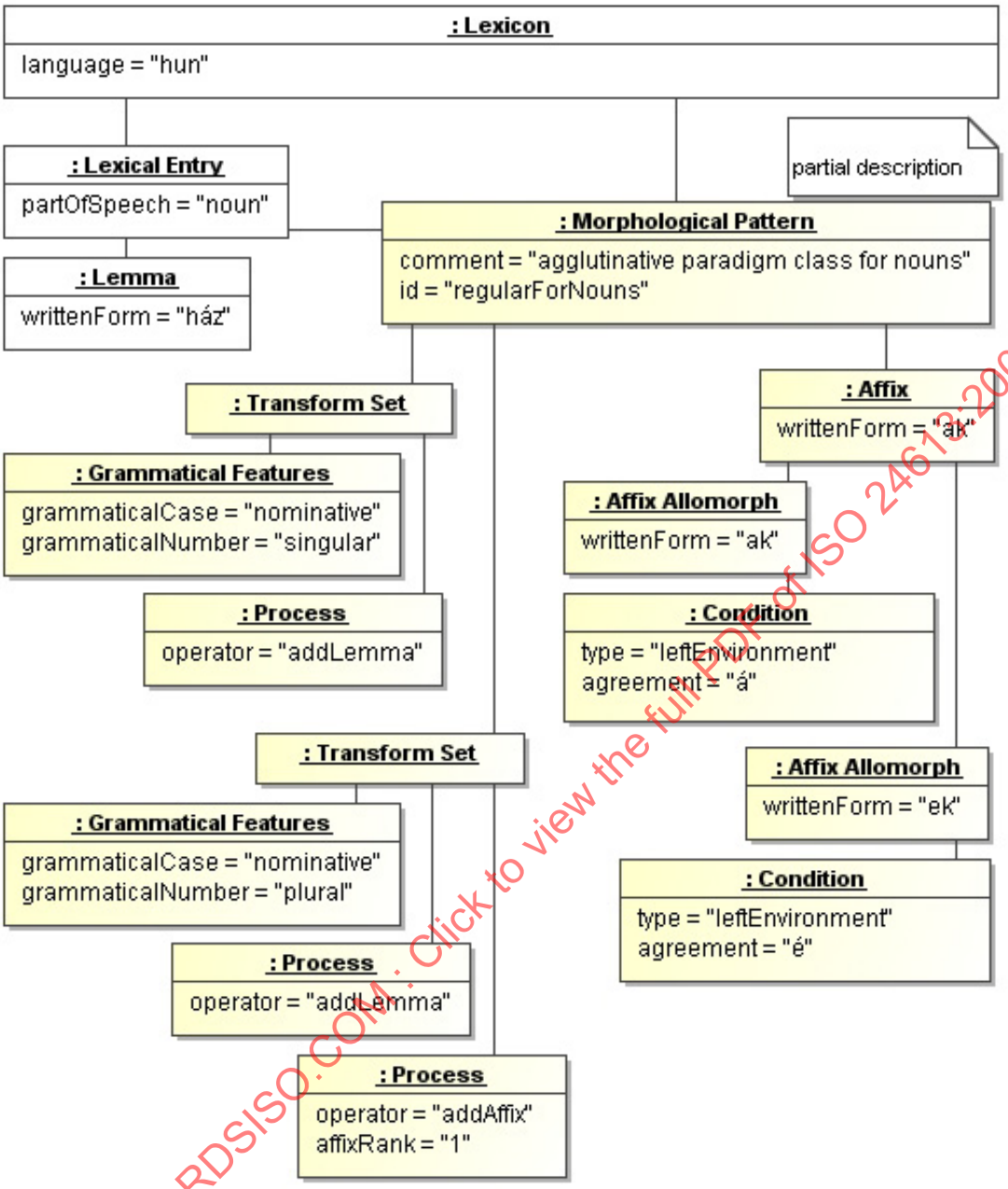

Figure L.9 — Lemma-based agglutination

It is worth noting 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 explicitly in the lexicon (like in Annex A) produces unmanageable documentation. Usually, a strategy based on morphological patterns is preferred.

L.3.10 Derivation using reduplication

In Thai, the derivation is a frequent mechanism, thus it is time consuming to record a separate entry for the derived form. Reduplication is used to modify the sense of a lexeme by some operation to repeat the sound of the lemma [25],[26].

The types of derivation using reduplication are:

— AA type. The form of reduplication is generated by attaching a character symbol 'Mai Yamok' (ๆ) to produce a reduplicated sound of the lemma. For instance, the lemma "ดำ" (to be pronounced "dam0" and that means "black") can be modified to give "ดำๆ" (to be pronounced "dam0-dam0" and that means "blackish") in order to express a generalization.

— A'A type (tone change in the first syllable), for instance, "ด๊ำดำ" (to be pronounced "dam3-dam0" and that means "extremely black") for intensification.

— AA'A type (triplication), for instance, "กินกิ๊นกิน" (to be pronounced "kin0-kin4-kin0" and that means "eat like a horse") for intensification.

— More complex mechanisms like AABB or AB'AB types.

The two following diagrams show different options for representing derivation. It is worth noting in these examples, in contrast to the other examples, two different *Morphological Pattern* instances are attached to the given *Lexical Entry* instance.

The first approach is simply to mark the *Morphological Pattern* instance by means of a reduplication type attribute, as presented in Figure L.10.

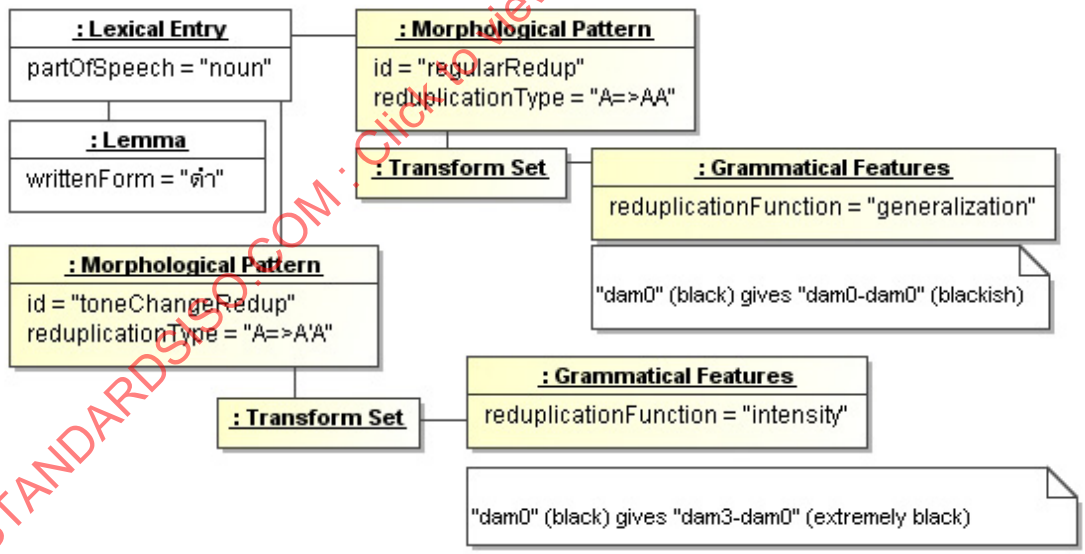

Figure L.10 — Derivation using reduplication (simple approach)

A more complex and detailed approach is to fully describe the reduplication by means of *Process* instances, as presented in Figure L.11.

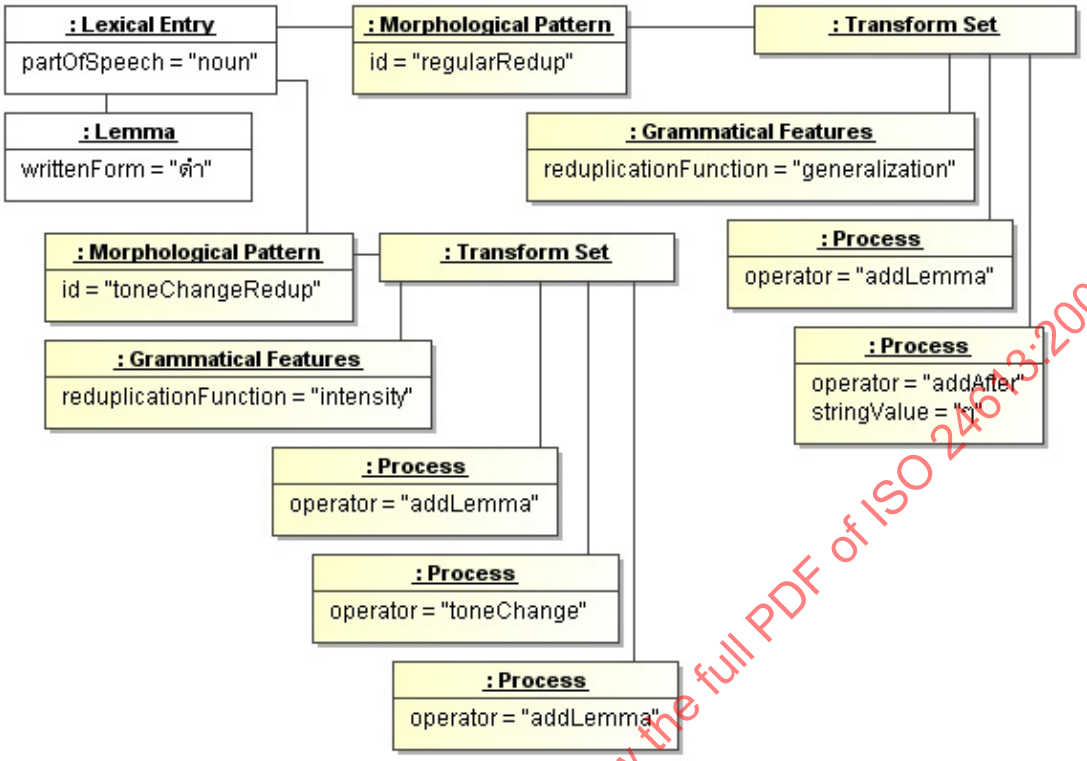

Figure L.11 — Derivation using reduplication (complex approach)

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

L.3.11 Lemma-based composition: Anwendungsprogramm

Composition is a bit different from inflection, agglutination and derivation. In these latest examples, each form is defined from the lemma (or one stem) of the entry with various types of operations, such as adding affixes. The description involves only one entry.

In the composition process, each component form of a compound entry is defined in the *List of Components* instance. In this way, the process is dependent on separate morphological behaviour of each component.

The following diagram illustrates a German example of an inflectional morphological pattern applied to a compound that involves 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 morphological pattern that describes how to build the compound. The first component is selected, an “s” is added, then the second component is added with the initial letter transformed into a lowercase letter, see Figure L.12.

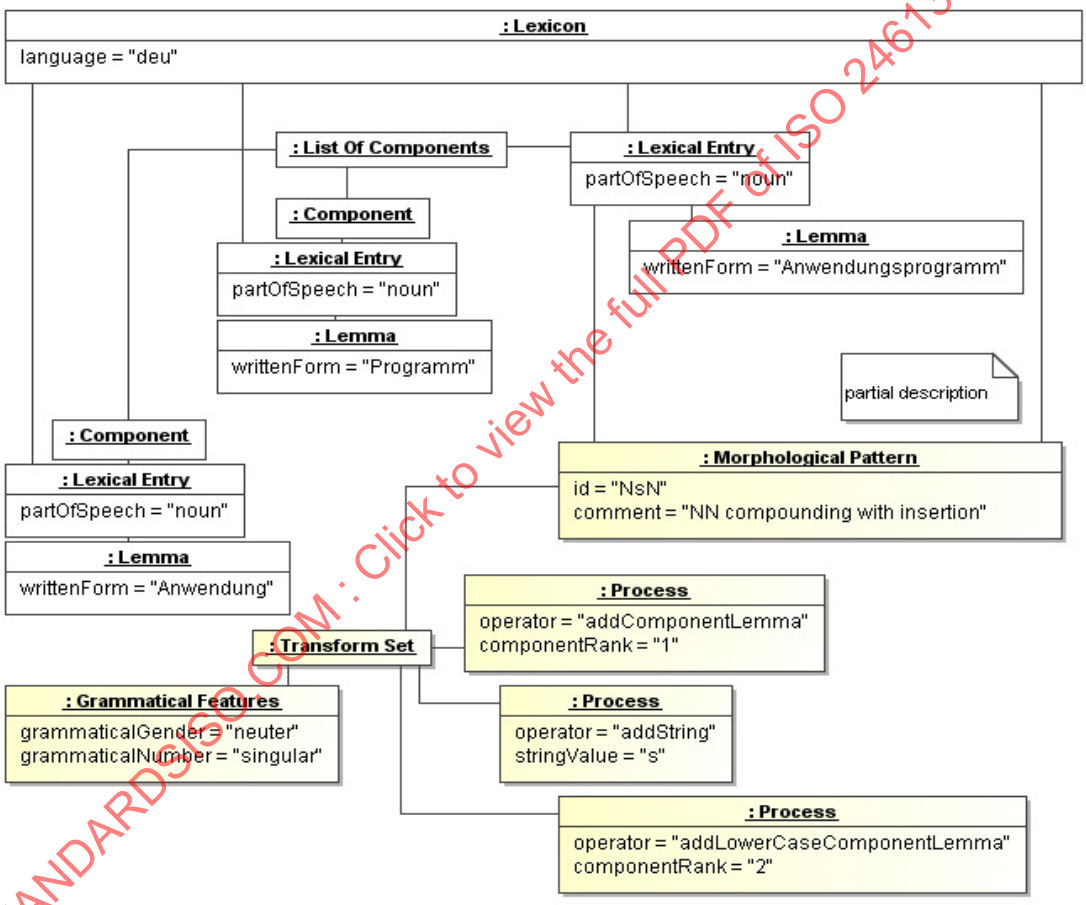

Figure L.12 — Example of a lemma-based composition