INTERNATIONAL STANDARD

ISO 24615-2

First edition
2018-02

Language resource management — Syntactic annotation framework (SynAF) —

Part 2: XML serialization (Tiger vocabulary)

Gestion de ressources linguistiques — Cadre d'annotation syntaxique (SynAF) —

Partie 2: Sérialisation XML (vocabulaire Tiger)

Reference number
ISO 24615-2:2018(E)

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.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular the different approval criteria needed for the different types of ISO documents should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

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. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation on the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see the following URL: www.iso.org/iso/foreword.html.

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

A list of all parts in the ISO 24615 series can be found on the ISO website.

Introduction

The need for standardization of syntactic annotation was recognized and addressed in detail with the publication of ISO 24615-1:2014. As a result of the work on ISO 24615-1:2014, it was anticipated that such a reference model for syntactic annotation should be associated with a concrete XML serialization in order to meet the specific needs of such applications as syntactic parsers or syntactic treebanks, where representations have to be exchanged and reused. Furthermore, such a serialization should be independent from the theoretical orientation and specific details of any specific annotation scheme.

This document answers this need on the basis of the seminal work carried out on the TigerXML format[3]. This starting point was chosen as a reference because it is widely used as a de facto standard for unrelated XML treebanks, with the advantages in terms of interoperability offered by its XML-based representations, as opposed to other frequently used formats, in particular, the Penn Treebank bracketing format[5] or the CoNLL format for dependency structures (see Reference[4]).

The document is designed to complement ISO 24615-1:2014 and to coordinate closely with ISO 24610, ISO 24611, ISO 24612 and ISO 12620.

This document therefore extends ISO 24615-1:2014 with an XML model based upon the Tiger XML vocabulary for the interchange of syntactically annotated data which is both standardized as well as language- and theory-independent. The proposed format directly instantiates all features of the meta-model defined in ISO 24615-1 and defines concrete serialized interfaces to the complementary ISO 24611 and ISO 12620, which provides the background for the DatCatInfo data category registry.

Language resource management — Syntactic annotation framework (SynAF) —

Part 2: **XML serialization (Tiger vocabulary)**

1 Scope

This document describes an XML-conformant serialization of the ISO 24615-1 meta-model, with the objective of supporting interoperability across language resources or language processing components in the domain of syntactic annotations. As an extension of ISO 24615-1, this document is also coordinated with ISO 24612.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements 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 12620, *Terminology and other language and content resources — Data category specifications*

ISO 24610 (all parts), *Language resource management — Feature structures*

ISO 24611, *Language resource management — Morpho-syntactic annotation framework (MAF)*

ISO 24612, *Language resource management — Linguistic annotation framework (LAF)*

ISO 24615-1, *Language resource management — Syntactic annotation framework (SynAF) — Part 1: Syntactic model*

3 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO 12620, ISO 24610 (all parts), ISO 24611, ISO 24612 and ISO 24615-1 and the following apply.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

— IEC Electropedia: available at http://www.electropedia.org/

— ISO Online browsing platform: available at http://www.iso.org/obp

3.1
domain
class of elements to which a certain set of *labels* (3.2) can be assigned

Note 1 to entry: Domains can refer generally to the set of all edges, terminal nodes or non-terminal nodes.

3.2
label
unit of annotation consisting of the name of a feature and a value, which together can be applied to appropriate model elements and add arbitrary feature-value annotations to such elements

3.3
primary data
initial raw linguistic content that is being encoded

3.4
sequential representation
representation of annotation content where the XML element structure mirrors the sequence of linguistic objects in the primary source

4 Graph structure and meta-model

In the XML Tiger format, annotations are represented in a graph structure. The graph structure can be described as G = (V, E, A) with

— a set of nodes V,

— a set of edges E with e = (v ∈ V, v ∈ V) ∈ E,

— a set of annotations A, where an annotation a is defined by a feature-value pair, and

— a function *annot*: E ∪ V → A.

A graph represents a bundle of interrelated nodes and edges. It is not specified which parts of a primary text are covered by a single graph, e.g. a sentence, a sub-sentence, a chapter or a whole text. Linguistic annotations represented by labels can be attached to nodes as well as edges.

The meta-model (see Figure 1) consists of three parts:

a) the structural organization of corpora and associated meta-data;

b) an annotation tagset definition;

c) the linguistic annotation graph.

The structural organization of corpora is represented by a recursively defined corpus element (Corpus) and its corresponding metadata (Meta). A corpus can contain subcorpora.

The annotation tagset definition is represented by a list of categories (Feature) containing the name of a category (Feature.name) and a list of category values (FeatureValue). Each FeatureValue contains a string representation of the value (FeatureValue.value). Together, both elements declare a tagset which is part of a specific corpus object. Such a tagset declaration is derivable, which means that all categories defined in a supercorpus object can also be used by its subcorpus objects. Further attributes are used to declare to which types of nodes and edges a category is applicable. Both elements allow reference to DatCatInfo entries in compliance with ISO 12620 via Unified Resource Identifiers (URIs) in the attributes Feature.dcrReference and FeatureValue.dcrReference.

The final part of the meta-model, the linguistic annotation graph, defines a set of elements containing the primary data and the annotation structure covering the primary data. It consists of the graph element itself (Graph), two classes of syntactic nodes (Terminal and NonTerminal), an edge element (Edge) and an annotation element (Annotation) realizing the *annot*-function and therefore referring to a feature name and its value. Graph is contained within Segment, which is a grouping mechanism to aggregate a set of syntactic nodes together. Such a group can have linguistic structural semantics, corresponding usually to a sentence, but possibly also to a line in a manuscript or other meaningful segments depending on the application and annotation scheme used by a specific project.

A terminal node (in ISO 24615-1 referred to as T_Node) constitutes the point of reference to the primary data. This can be a direct reference to a text span within the XML document or an indirect reference to an object outside the model, e.g. an element contained by a MAF file in compliance with ISO 24611, the Morpho-syntactic annotation framework. A non-terminal node is an inner node, referring directly or indirectly to a terminal node within the XML document.

An edge shall always have a source and a target node. Both of them can be either a terminal or a non-terminal node.

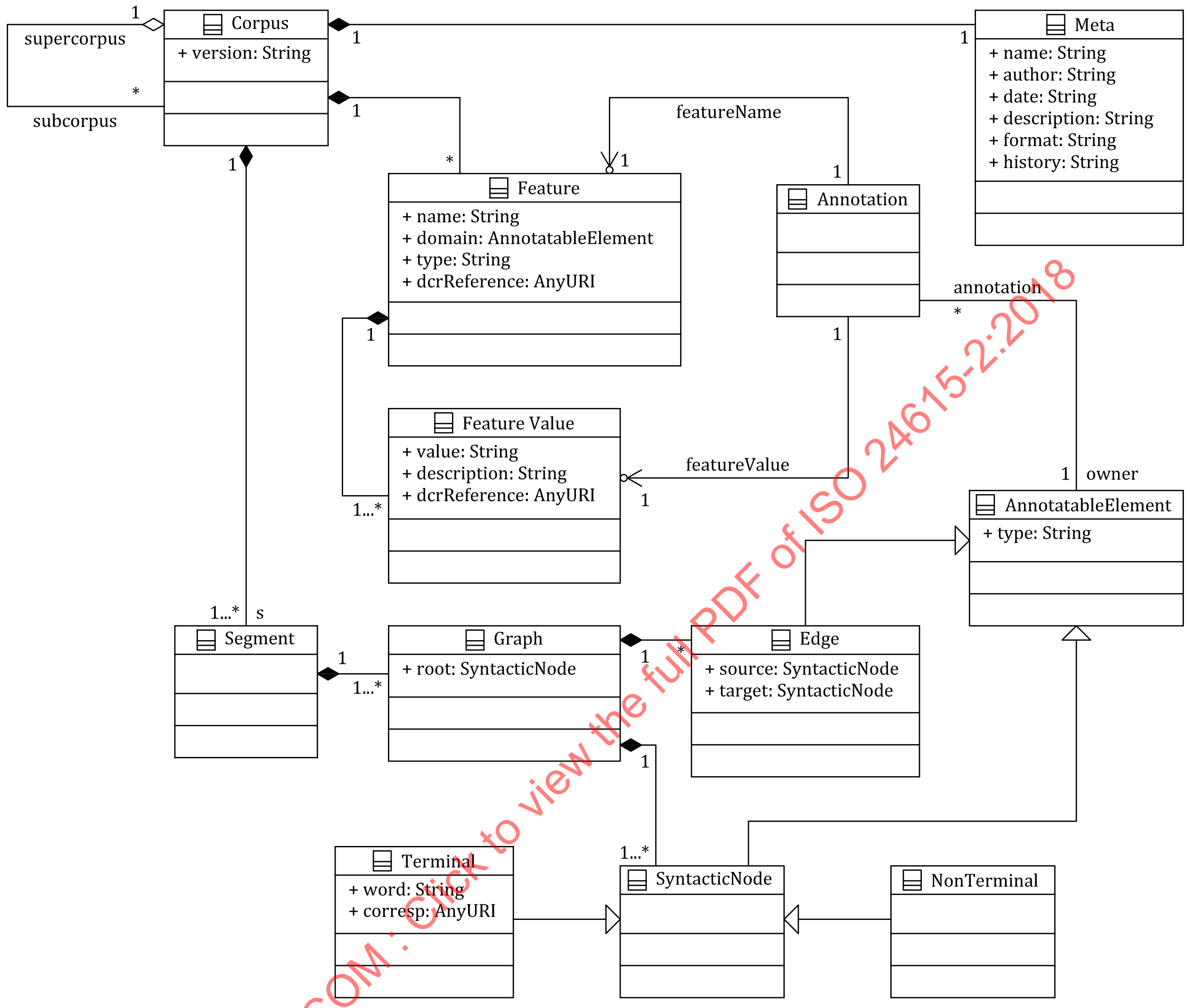

Figure 1 — Meta-model for the serialization format

5 Meta-model objects in XML serialization

The names of the XML elements and attributes follow those of the corresponding meta-model elements and attributes. All XML elements belong to the following namespace: http://www.clarin.eu/standards/ns/synaf. In this document, unless specified otherwise, all XML elements will be assumed to belong to this namespace.

Terminal nodes in the XML serialization are represented by the `<t>` element and are nested together in a `<terminals>` element.

A segment node is represented by the `<s>` element. The `<s>` element shall contain one or more `<graph>` elements, which may be used to express possible multiple annotation graphs alternating within a single segment or to represent a sequence of subgraphs.

A non-terminal node is represented by the `<nt>` element and is nested in the `<nonterminals>` element.

An edge is represented by the <edge> element and its source is given by the surrounding node element. An <edge> element is therefore always the child element of a <t> or an <nt> element. The target is given by the @target attribute and shall refer to a node within the same document.

The example shows the representation of the graph structure.

EXAMPLE Graph structure of a syntactic annotation.

```
<s>
    <graph xml:id="s1_g1">
        <terminals>
            <t xml:id="s1_t1"/>
            <t xml:id="s1_t2"/>
        </terminals>
        <nonterminals
            <nt xml:id="s1_nt1">
                <edge xml:id="s1_e1" target="#s1_t1" />
              <edge xml:id="s1_e2" target="#s1_t2" />
            </nt>
        </nonterminals>
    </graph>
</s>
```

6 Primary data and terminal representation

6.1 General

Terminal nodes and the primary data to which they refer can be defined in two different ways. The primary data can either be included within the XML document (sequential representation) or they can be specified in another file and referred to externally (standoff representation). The sequential representation makes it possible to have all data in just one XML document, whereas the standoff representation makes it possible to refer to other formats, for instance, a MAF file containing tokens and wordForms.

Both options are possible and the decision is largely made based on the needs of the corpus project.

6.2 Sequential representation

In a sequential representation, primary data is represented sequentially directly within the <t> elements of each <s> element in the document. As a result, all the primary data of a document is grouped together in one single XML document. This improves human readability and basic machine processing, but reduces representational flexibility. Tokens of the primary data are encoded as values of the @word attribute of the <t> element as illustrated in Example 1.

EXAMPLE 1 Simple sequential representation of word forms.

```
<t xml:id="s1_t1" word="two"/>
<t xml:id="s1_t2" word="words"/>
```

Example 1 shows two terminal nodes, the first containing the word "two" and the second containing the word "words". The order of the tokens in the primary data is mirrored by the order of the XML elements.

However, it should be pointed out that this representation does not need to fully preserve primary data, since only textual material that has a corresponding <t> element is preserved. This means that parts of an utterance that are not considered to be tokens (untokenized material), such as whitespaces between words or enumerations, non-linguistic characters, etc., will be lost. Example 2 and Example 3 illustrate this.

EXAMPLE 2 Primary data.

```
This is a sample.
```

EXAMPLE 3 Sequential representation.

```
<t xml:id="s1_t1" word="This"/>
<t xml:id="s1_t2" word="is"/>
<t xml:id="s1_t3" word="sample"/>
<t xml:id="s1_t4" word="."/>
```

In these examples, it is not possible to resolve whether there was originally a whitespace character between the tokens corresponding to the `@word` attributes of the terminals or not. In the case of the terminals "`s1_t1`" and "`s1_t2`" there was one, whereas between "`s1_t3`" and "`s1_t4`" there was no intervening whitespace. This can be handled by means of the `@join` attribute as provided by ISO 24611. It is also possible to omit textual material on purpose, leading to the text span corresponding to the word "`a`" becoming lost in the XML document.

Because of the restrictions of XML attributes, reserved XML signs such as angle brackets ("<") cannot be represented inside an XML attribute value. These are therefore escaped using the sequence "`&lt`"; (see Example 4), which, in some cases, can cause confusion in character-based offset calculations.

EXAMPLE 4 Sequential representation with character escaping.

```
<t xml:id="s1_t4" word="&lt; "/>
```

Cases where the primary data contains additional mark-up can be difficult to handle on a sequential encoding strategy.

6.3 Standoff representation

Problems concerning special characters or mark-up embedded in the primary data, described in 6.2, can be avoided by using a standoff representation. In a standoff representation, the `@corresp` attribute of the `<t>` element is used to point to a sequence of tokens or word forms in the primary data. The `@corresp` attribute replaces the `@word` attribute and contains a URI reference to another resource, for example, a MAF file. If both attributes are provided, the `@word` attribute takes precedence and the `@corresp` attribute can usually be ignored. The following examples show the use of the standoff representation with the `@corresp` attribute. For these examples, the token and wordForm elements from ISO 24611 data model are applied and the references into the primary data utilize anchors to refer to locations in between the base units of the data representation as specified in ISO 24612.

EXAMPLE 1 Locations in the primary data document.

```
<s xml:id="s1">We can…</s>
```

This sentence is associated with the following segmentation, with an inter-character numbering:

```
|W|e| |c|a|n|…
0 1 2 3 4 5 6
```

EXAMPLE 2 MAF standoff representation, in TEI flavour.

```
<w xml:id="t1" corresp="#string-range(s1,0,2)">
<w xml:id="t2" corresp="#string-range(s1,3,6)">

…

<span type="wordForm" xml:id="wf_1" lemma="we" corresp="#t1"/>
<span type="wordForm" xml:id="wf_2" lemma="can" corresp="#t2"/>
```

EXAMPLE 3 Standoff representation expressed in the Tiger vocabulary.

```
<terminals>
 <t xml:id="s9_1" corresp="example.maf#wf_1"/>
 <t xml:id="s9_2" corresp="example.maf#wf_2"/>
</terminals>
```

These examples refer to three different files: the primary text document, a MAF file (see ISO 24611) containing the tokenization and an aggregation of tokens to wordForms and an XML document referring to the MAF document.

6.4 Typing of nodes and edges

Nodes and edges can also be given types. A type is a special kind of annotation and is represented by the `@type` attribute contained in a `<t>`, `<nt>` or `<edge>` element. For example, the type of an edge can be "`dep`" to denote a dependency edge and to distinguish it from constituency edges.

The domain of possible `@type` values is not defined here, although it is recommended to use DatCatInfo (an implementation of ISO 12620) for the identification of relevant data categories.

For a more specific definition of an `@type` value, it is possible to define the domain of the values via an annotation with a `<feature>` element (see detail in 7.6). If the `@type` attribute is not used within a `<t>`, `<nt>` or `<edge>` element, they will be implicitly set to a default value. In the case of an edge, the type will be set to "`edge`", in the case of a terminal node, it will be set to "`t`" and in the case of a non-terminal node, it will be set to "`nt`".

The `@type` attribute is also used for the declaration of the domain for annotation-feature names and the domain for annotation-feature values as described in Clause 7.

7 Annotations

7.1 General

Annotations are attribute-value pairs that can be appended to nodes and edges. This is mirrored by the names and values of XML attributes. In general, there are no restrictions for setting names or values for annotations depending on specific linguistic theories. Therefore, it is possible to append any feature-value pair to a `<t>`, `<nt>` or `<edge>` element, except for the reserved attributes of the Tiger vocabulary, for example, `@type`, `@word`, `@corresp` and `@domain`. These attributes have fixed semantics and cannot be overridden.

The example shows the annotation mechanism for nodes and edges.

EXAMPLE Annotation declaration for `<t>`, `<nt>` and `<edge>` elements.

```
<terminals>
    <t xml:id="s1_t1" word="I" lemma="I" pos="PP"/>
    <!-- ... -->
</terminals>
<nonterminals>
    <nt xml:id="s1_nt1" cat="NP">
        <edge xml:id="s1_e1" label="HD" target="#s1_t1"/>
    </nt>
    <!-- ... -->
</nonterminals>
```

In the example, the word "`I`" is annotated with the part-of-speech annotation "`PP`" (standing for "personal pronoun") and the lemma annotation "`I`". Neither "`pos`" nor "`lemma`" are reserved attribute names; they could also be named, e.g. "`part-of-speech`" or "`lem`", or any other non-reserved name could be used, so it is recommended to use references to the DatCatInfo Data Category Registry for disambiguation (see 7.5). The non-terminal "s1_nt1", annotated with the attribute @cat (for "category") with the value "NP" (for "nominal phrase") is syntactically headed by the terminal "s1_t1". This relationship is expressed by the edge "s1_e1" which has as its source "s1_nt1", and as its target "s1_t1", and bears the annotation ("label='HD'"), which indicates headedness.

7.2 Domain declaration

It is recommended to define a domain for annotation names. This should be done in the header section of the corpus, in the <annotation> element inside the element <head>. The @domain attribute indicates whether the annotation being defined should be applied to terminals, non-terminals or edges. For the annotations present in the example in 7.1, the relevant domains are set as shown in the example.

EXAMPLE Domain declaration for annotation attributes.

```
<head>
    <annotation>
        <feature xml:id="f1" name="pos" domain="t" />
        <feature xml:id="f2" name="lemma" domain="t" />
        <feature xml:id="f3" name="cat" domain="nt"/>
        <feature xml:id="f4" name="label" domain="edge"/>
    </annotation>
</head>
```

The mapping between the annotation attribute and the domain declaration is achieved by resolving the name of the XML attribute of the annotation and the @name attribute of the element <feature>. The attribute @domain specifies that the declared feature is applicable only to the given kind of XML elements. In the given example, this means that the feature declarations for "pos" and "lemma" can only be used by <t> elements, "cat" can only be used by <nt> elements and "label" can only be used by <edge> elements. The attribute @domain is optional and need not be set. If it is not set, it means that the feature declaration is available for all three kinds of elements.

7.3 Declaring annotations in an external file

It is also possible to declare annotations in a separate XML file referenced from within the corpus. This can be useful when authoring or curating large numbers of files which all have the same annotation scheme. In this case, the element <external> and the attribute @corresp should be used to specify the location of the annotation declaration file. Example 1 illustrates the use of external annotation declaration files.

EXAMPLE 1 Referencing external annotation declarations.

```
<corpus xmlns="http://www.clarin.eu/standards/ns/synaf">
   <head>
    <annotation>
           <external corresp="MyAnnotations.xml"/>
    </annotation>
   </head>
</corpus>
```

An external annotation declaration looks exactly like an annotation declaration (see 7.2). Example 2 shows how such a declaration works.

EXAMPLE 2 Content of an external declaration annotation.

```
<annotation xmlns="http://www.clarin.eu/standards/ns/synaf">
     <feature xml:id="f1" name="pos" domain="t" />
     <feature xml:id="f2" name="lemma" domain="t" />
     <feature xml:id="f3" name="cat" domain="nt"/>
     <feature xml:id="f4" name="label" domain="edge"/>
</annotation>
```

7.4 Feature values

Beyond domains for annotation names, it is also possible to restrict the use of annotation values. This can be achieved by using the <value> element inside the <feature> element. The <value> element has an @name attribute, which identifies the value name and also works as an anchor for annotation values. The element can also contain text, which acts as a description for the annotation-value. The example shows the use of an annotation value declaration.

EXAMPLE Referencing domains for annotation feature name and value.

```
<head>
    <annotation>
        <feature xml:id="f2" name="pos" domain="t">
            <value xml:id="f2_1" name="PP">Personal pronoun</value>
        </feature>
    </annotation>
</head>
<!-- ... -->
<terminals>
    <t id="s1_t1" word="I" pos="PP"/>
</terminals>
```

In the case of the example, the set of possible values for an annotation with the name "pos" only contains a "PP" element, standing for a personal pronoun. In this case, no other values are possible for this annotation of terminal nodes. To increase the set of possible annotation values, it is possible to add further <value> elements to the <feature> element. The number of possible annotation values is not limited. If there are no <value> elements given inside a <feature> element, the set of annotation values is not defined and can be any string legal as an XML-attribute value. In this case, validation of annotation values cannot take place. This is useful, e.g. for lemma annotations, for which it is typically not recommended to define the domain of possible annotation values.

7.5 Data category references

In many cases, annotation declarations are not only interesting for one specific corpus or project, but are of interest to a whole community. There is thus the possibility of linking annotations to DatCatInfo data categories (see also http://www.datcatinfo.net/). The attribute @datcat can be used in <feature> and <value> elements to refer to DatCatInfo categories via URI syntax. The example shows the use of the @datcat attribute.

EXAMPLE Using a data category registry in annotation declarations.

```
<corpus xmlns="http://www.clarin.eu/standards/ns/synaf" version="2.0.5"
   xml:id="c1" xmlns:dcr="http://www.datcatinfo.net/ns/dcr">
    <head>
        <annotation>
            <feature xml:id="f2" name="pos" domain="t"
             dcr:datcat="http://www.datcatinfo.net/datcat/DC-396">
                <value
                    xml:id="f2_1" name="PP"
                    dcr:datcat="http://www.datcatinfo.net/datcat/DC-1463"/>
            </feature>
        </annotation>
    </head>
    <!-- ... -->
    <terminals>
        <t id="s1_t1" word="I" pos="PP"/>
    </terminals>
</corpus>
```

NOTE In the preceding example, the reference to the namespace for the @datcat attribute is one of the possible instances. Depending on the context and the actual reference data category vocabulary that an implementer is deciding to use, this namespace URI can be changed as needed.

In the example, a terminal covering the word "I" is annotated by a part-of-speech annotation ("pos") with the value "personal pronoun". The domain for the annotation name and the annotation value is declared in the <annotation> element, as seen in the examples above. Additionally, in this example, the @datcat attribute is used to refer to a DatCatInfo data point. The annotation name is not only declared within the XML document, but also in the shared registry. This can also substitute the description inside the <value> element, provided that an adequate description is given by the DatCatInfo entry, although this document does not prohibit using both mechanisms concurrently.

7.6 Using the @type attribute

In cases where a finer granularity is required for the declaration of annotation domains, it is possible to use the @type attribute inside the <t>, <nt> and <edge> elements. As mentioned in 6.4, this attribute determines the type of the given element and also determines a correspondence to the declaration of the annotation domain. The element <feature> also contains an optional @type attribute, which can restrict the subset of nodes and edges defined by the annotation domain. Example 1 demonstrates this mechanism.

EXAMPLE 1 Using the @type attribute for annotation domain declaration.

```
<head>
   <annotation>
      <feature xml:id="f2" type="wordform" name="lemma" domain="t">
         <value xml:id="f2_1" name="PP">Personal pronoun</value>
      </feature>
   </annotation>
</head>
...
<s>
    <graph>
        <terminals>
                <t id="s1_t1" word="I" type="wordform" lemma="I" pos="PP"/>
        </terminals>
        …
    </graph>
</s>
```

In Example 1, an annotation name for terminal "s1_t1" is defined by the feature "f2". The @type attribute guarantees that the feature "f2" only declares an annotation name for terminals of the type "wordform". A terminal without this type cannot be referenced within that domain. If no @type attribute is given in the <feature> element, the domain works for all elements respecting the domain declaration (<t>, <nt> or <edge> elements).

It is also possible to declare the values of the @type attribute. The relevant mechanism is the same as for declaring annotation values, namely via the <feature> and <value> element. Example 2 shows the declaration of the domain for types.

EXAMPLE 2 Declaration of values of the @type attribute.

```
<feature xml:id="t1" name="type" domain="t">
    <value xml:id="t1_1" name="wordform">Abstraction over token(s), corresponding to a
linguistic entity, cf. wordForm in ISO 24611
</value>
    ...
</feature>
<feature xml:id="t2" name="type" domain="edge">
    <value xml:id="t2_1" name="dep">Grammatical dependency</value>
    ...
</feature>
```

Example 2 declares sample values of the @type attribute within the domains of terminals and edges and provides glosses for these values.

8 Corpus structure

It is possible to represent a complete corpus structure containing several subcorpora in a single XML document. But in each such document, only one root corpus may exist, represented by the `<corpus>` element. The `<corpus>` element shall specify the format version of the data using the `@version` attribute. A subordinate corpus structure, or more precisely a subcorpus, can be created by using the `<subcorpus>` element inside the `<corpus>` element. `@version` is inherited by subcorpora and should not be specified individually for each subcorpus. The `<subcorpus>` element is a recursive structure which allows for an unbounded depth of embedded subcorpora. Example 1 shows the corpus structure represented in XML.

EXAMPLE 1   Corpus structure with subcorpora.

```
<corpus xml:id="c1" version="2.0.5" xmlns="http://www.clarin.eu/standards/ns/synaf">
    <head>
    ...
    </head>
    <body/>
    <subcorpus xml:id="c2">
           <head/>
           <body/>
           <subcorpus xml:id="c3">
               <head/>
               <body/>
           </subcorpus>
    </subcorpus>
    <subcorpus xml:id="c4">
           <head/>
           <body/>
    </subcorpus>
</corpus>
```

Example 1 shows a corpus “`c1`” containing in total three subcorpora. Two of them (“`c2`” and “`c4`”) are directly contained in “`c1`”, and “`c3`” is directly contained in “`c2`”, and therefore, indirectly contained in “`c1`”.

The element `<head>` groups the corpus metadata, while the element `<body>` contains segments (`<s>`), which may contain one or more `<graph>` elements; see Example 2.

EXAMPLE 2   The content of `<body>`.

```
<body>
    <s xml:id="s1">
        <graph xml:id="s1_g1"/>
        <graph xml:id="s1_g2"/>
        ...
    </s>
    <s xml:id="s2">
        <graph xml:id="s2_g1"/>
        <graph xml:id="s2_g2"/>
        ...
    </s>
...
<body>
```

The relationship between the contents of the `<graph>` elements is not prescribed by this document. They may cover exactly the same primary data, creating alternative annotation graphs, they may cover separate spans of `<s>` and they may also partially overlap.

A corpus or subcorpus can be annotated by meta-annotations. Using meta-annotations differs from the way nodes and edges are annotated. This document provides a fixed set of possible meta-annotations, although it is possible to link a given document in a standoff fashion to an XML file specifying meta-annotations in either the serialization defined here or even a different format. Meta-annotations are limited to the header of a corpus or subcorpus. They are identified by the enclosing element `<meta>`. The