Abstract: This paper presents the application of the <tiger2/> format to various linguistic scenarios with the aim of making it the standard serialisation for the ISO 24615 (SynAF) standard. After outlining the main characteristics of both the SynAF metamodel and the <tiger2/> format, as extended from the initial Tiger XML format (König & Lezius, 2000), we show through a range of different language families how <tiger2/> covers a variety of constituency and dependency based analyses.

1 From SynAF to <tiger2/>

In 2010, the International Organization for Standardization (ISO) published ISO 24615 (SynAF), which provides a reference model for the representation of syntactic annotations. This document elaborates on early proposals for the representation of syntactic information (Leech et al, 1995; Ide & Romary, 2003) to provide a comprehensive framework based on the following principles:

- Dealing with the representation of both syntactically annotated corpora and data resulting from an automatic syntactic analysis
- Equally covering dependency and constituency based representations
- Being flexible enough to deal with a variety of languages by relying on an open metamodel which can be parameterized by means of specific data categories

Still, the SynAF standard did not contain any specific XML serialisation. This prevented it from being sufficient for implementers who would want to ensure that their data be interoperable with other initiatives, at least for the same language. This is why ISO subcommittee TC 37/SC 4 initiated a new activity, in order to provide an additional part to SynAF on the basis of the Tiger XML format (König & Lezius, 2000). This format has been slightly adapted\(^1\) in the meantime under the code name <tiger2/>\(^2\), to comply with recent XML technology developments, to integrate some necessary features and to fully encompass dependency-based representations.

Before proceeding any further with this initiative, the SC4 experts who attended the ISO TC37 meeting in Madrid in June 2012 decided to put <tiger2/> to the test, by applying it to a variety of language samples. This should provide a concrete basis for the future committee work, which should integrate the feedback from the community interested in syntactic annotation and in particular colleagues involved in the development of treebanks. In this context, this paper is intended to provide a first overview of these experiments, with the aim of raising some interest and thereby getting some feedback from a variety of interested experts.

This paper will first of all describe the major characteristics of both the ISO SynAF standard and the <tiger2/> formats and then present a series of short sections demonstrating a) the possible application

---

\(^{1}\) All background information on <tiger2/> can be found at [http://korpling.german.hu-berlin.de/tiger2/](http://korpling.german.hu-berlin.de/tiger2/), comprising schemas, documentation and encoded examples.

\(^{2}\) Typically: generic XML attributes, schema technology, XML design good practices. Although it is not the aim of this paper to go in to the corresponding technical details, we encourage interested readers to refer to the <Tiger2/> documentation.
of <tiger2/> in specific linguistic contexts; and b) how SynAF can be used as an interchange format for existing tools. As mentioned, this paper discusses work in progress. Consequently, we expect it to lead other colleagues to start experimenting with <tiger2/> and, thus, provide some concrete feedback to the forthcoming standardisation work.

2 The SynAF metamodel

ISO 24615 (SynAF) provides a high-level metamodel for representing the syntactic annotation of linguistic data, with the objective of supporting interoperability across language resources or language processing components. SynAF aims at covering the two main functions of syntactic annotation in language processing, namely: a) to represent linguistic constituency, as in Noun Phrases (NP) etc., describing a structured sequence of morpho-syntactically annotated items (including, depending on the theory used, empty elements or traces generated by movements at the constituency level), as well as constituents built from non-contiguous elements; and b) to represent dependency relations, such as head-modifier relations, and also including relations between categories of the same kind (such as the head-head relations between nouns in appositions, or nominal coordinations in some formalisms).

As a consequence, syntactic annotations must comply with a multi-layered annotation strategy, which interrelates syntactic annotation for both constituency and dependency, possibly simultaneously, as stated in the SynAF metamodel. The metamodel is based on the following components:

- **The T_node component** represents the terminal nodes of a syntactic tree, consisting of morpho-syntactically annotated word forms, as well as empty elements when appropriate. T_nodes are annotated with syntactic categories valid for the word level.
- **The NT_node component** represents the non-terminal nodes of a syntactic tree. The NT_nodes are annotated with syntactic categories that are valid at the phrasal, clausal and/or sentential levels.
- **The Edge component** represents a relation between syntactic nodes (both terminal and non-terminal nodes). For example, the dependency relation is binary, consisting of a pair of source and target nodes, with one or more annotations.

From this metamodel, a specific syntactic annotation model can be obtained by combining the above-mentioned components with data categories characterising or refining their semantics.

It should be noticed that the terminal node level in SynAF is strongly equivalent to the word form level in MAF (ISO 24611), for which we offer concrete interfaces below. It is thus left to the implementer to either separate or merge these two components depending on whether it is relevant or not, for instance, to clearly differentiate the data categories attached to word forms and terminals within a multi-layered annotated corpus.

3 <tiger2/> as a SynAF compliant Tiger

The main characteristics of the <tiger2/> datamodel can be summarized as follows:

- **Terminal nodes** are implemented as <t> elements, either referring to a textual segment or pointing to a word form in a morphosyntactically annotated corpus. Non-terminal nodes are implemented as <nt> elements and are used to represent hierarchical structures like syntactic trees.
- **The Edge component** is implemented by means of an <edge> element, which may relate either <nt> or <t> elements to each other.
• A generic @type attribute inside terminal, non-terminal and edge elements can further qualify
the meaning of these elements. For instance a non-terminal node can be qualified as a verbal
complex, a terminal can be specified as a compound node etc.
• The elements terminal, non-terminal and edge as well are used as placeholders for further
linguistic annotations. Such annotations can freely be added as generic attribute-value-pairs
with no restrictions to a specific linguistic school or theory (e.g. attributes like @cat,
@phrase, … or any user defined attributes).
• In addition to corpus metadata, an annotation block at the beginning of the document allows
to declare the particular descriptive features used in the annotation project.
• An important change from the initial Tiger format (König & Lezius, 2000) is that with
<tiger2/> it is possible to directly implement edges between terminal modes, thus
empowering the datamodel with a fully-fledged dependency-based representation. As
demonstrated in section 4.2 for Semitic languages, such edges may appear as children of <t>
elements. They may also be used for further sentential and even inter-sentential relations such
as binding and anaphora (see also subsections 4.1.4 and 4.3.5).

All these features have been implemented on the basis of a complete specification written in the TEI
ODD language from which one can generate both various schemas (DTD, RelaxNG, W3C) and the
actual documentation. Since not all the features integrated in <Tiger2/> may be needed at the same
time for a specific implementation, it is anticipated that project specific customizations, based on the
ODD developed made so far, will allow the community to converge on specific flavours of <Tiger2/>.

4 <tiger2/> at work – linguistic cases

At this stage, we do not want to discuss the actual coverage and expressive power of the <tiger2/>
proposal further without validating it across a variety of possible languages. To this end, the authors
of this paper have volunteered to test the <tiger2/> proposal on the languages where they had the
appropriate expertise. This is reflected in this section, where a quick overview of the main
characteristics and challenges is given for a sample of European, African and Asian languages,
followed by the explicit representation of an example in <tiger2/>. Even if the actual validation of the
model will only occur when wide-coverage syntactic analyses or treebanks will be transformed into
this format\(^1\), it will allow us to estimate in which direction the work on the future second part of
SynAF should go.

4.1 Spanish and Romance Languages

The set of syntactic phenomena that are present in Romance languages do not differ much from those
present in English. Hence, their representation does not require adding many other particular devices
to the ones included in SynAF and/or in <tiger2/>. However, they share some common particularities
that will be discussed next, namely enclitic objects and contractions, elliptic subjects and redundancy,
which can be described using a combination of MAF and SynAF (Alarcos-Llorach, 1999:2004; Real
Academia Española, 2009; Pineda & Meza, 2005). Each of them is presented in a dedicated
subsection below.

4.1.1 Spanish enclitic objects

First, most Romance languages allow clitic constructions, much like Semitic languages (see Section
4.2). For example, the Spanish sentence ‘Díselo al médico’ (= ‘Tell [to him] it to the doctor) includes
a couple of enclitic pronouns (‘se’ and ‘lo’) attached to the word form ‘Dí’, which is the main verb of

\(^1\) and accordingly a comprehensive set of data categories for such languages will be made available in ISOCat
the sentence. This phenomenon can be found also in Italian and in Catalan, amongst others. The morphosyntactic segmentation of this sentence is shown in Example 1, using the encoding scheme of MAF, which also shows the particular morphosyntactic annotation of ‘Díselo’.

Example 1: Morphosyntactic segmentation of the Spanish sentence 'Díselo al médico' and morphosyntactic annotation of the clitic construction 'Díselo'

4.1.2 Contractions
Second, most Romance languages include some contractions, that is, some word forms consisting of the fusion of a preposition and an article. This is not particular to Romance languages, since English and German (for example) include this type of words too. Example 2 shows how the Spanish contraction ‘al’ (= ‘to the’, which corresponds also to ‘au’ in French, for instance) has been annotated conforming to MAF.

Example 2: Morphosyntactic segmentation of the Spanish contraction 'al' and morphosyntactic annotation of the preposition 'a' and the definite article 'el'
Example 2: Morphosyntactic annotation of the Spanish contraction 'al'

4.1.3 Elliptic subjects

Third, Romance languages are characterized by the agreement in person and in number between the subject and the main verb of the sentence, and some of them (like Spanish) also require the subject and the main verb to agree in gender in certain cases. Besides, the inflection of Spanish verbs usually identifies unambiguously the person and the number of the subject. Due to this, Spanish subjects are elliptic in most cases. In fact, if the subject is made explicit in such cases, it can be considered redundant and, hence, pragmatically emphasized. Therefore, a mechanism for making elliptic subjects explicit might be required in order to annotate the full structure and/or set of dependencies of Spanish sentences. This mechanism would be similar to the one shown in Example 1 and 2 for Buntu (see 4.3.5), which shows how some elements that are not present on the surface structure can be made explicit by means of the standard.

4.1.4 Grammar-required redundancy

Fourth, the enclitic constructions mentioned in Section 4.1.1 usually entail the inclusion of a redundant syntactic element, like the pronoun 'se' in the Spanish sentence ‘Déjalo al médico’. This a personal pronoun, which coreferences the nominal phrase ‘al médico’ and is, thus, redundant. Nevertheless, removing this pronoun would produce a wrong (that is, ungrammatical) sentence. So a coreference mechanism has to be allowed in order to tag this type of redundancies. This can be done also following the same mechanism used in Example 1 and 2, as shown in Example 3. The annotation associated to redundancy has been highlighted for convenience. The example only includes the section for terminals for the sake of space.
4.2 Semitic Languages

Semitic languages bring with them a particular set of challenges for modelling linguistic data. The following two subsections demonstrate how MAF and SynAF/<tiger/> can work together to represent two problematic constructions: dependencies for object clitics and constituents for construct state compounds.

4.2.1 Arabic enclitic objects

Much like Romance languages, both Arabic and Hebrew have the ability to express pronominal objects as enclitic pronouns that are written together with the verb, forming one word form on the orthographic level (see Ryding, 2005:305–307), e.g. in Standard Arabic:

\[
\text{رأيت الملك} \\
\text{ra'a'itu l-malik 'I saw the king'} \\
\text{رآيته} \\
\text{ra'a'itu-hu 'I saw him'}
\]

Modelling a syntax tree of the dependencies between verb and object can be difficult and usually leads to make the design decision of tokenizing enclitic object pronouns as separate word forms. Using MAF word forms it is possible to keep such orthographic strings together while referring directly to constituent elements in the SynAF representation. The following representation gives a possible solution for a unified representation of the examples above, similar to the Spanish case.

Example 4: MAF representation of the Arabic clitic examples above.

```xml
<s xml:id="s1">
  <graph>
    <!-- ra'a'itu = I saw -->
    <t xml:id="s1_t1" tiger2:corresp="arabic.maf.xml#s1wf1"/>
    <t xml:id="s1_t2" tiger2:corresp="arabic.maf.xml#s1wf2"/>
    <t xml:id="s1_t3" tiger2:corresp="arabic.maf.xml#s1wf3"/>
    <token xml:id="s1tk1">
      ra'a'itu
    </token>
    <token xml:id="s1tk2">
      l
    </token>
    <token xml:id="s1tk3">
      malik
    </token>
    <token xml:id="s1tk4">
      .
    </token>
    <wordForm xml:id="s1wf1" tokens="s1tk1"/>
    <wordForm xml:id="s1wf2" tokens="s1tk2 s1tk3"/>
    <wordForm xml:id="s1wf3" tokens="s1tk4"/>
  </graph>
</s>
```

Example 4: SynAF-TIGER2-conformant syntactic annotation of the Spanish sentence 'Díselo al médico.'
Example 5: \textit{<tiger2/>} representation of dependency trees for the Arabic examples above.

By pointing at both word forms and tokens, the SynAF representation can contain uniform syntax trees for both enclitic and separate objects.

4.2.2 Hebrew construct states

Both Hebrew and Arabic use a special left-headed construction similar to compounding (see Borer 2008) to form complex nouns. This construction has a number of special features. Most notably for the present discussion, it does not constitute an orthographic word form, but it allows only one article for the entire complex noun, placed between the head and the modifier (or before the last modifier in recursive complex nouns). The article, like all articles in both languages, is written together with the final modifier and does constitute one word form with it, but marks the entire construction as definite. The following examples from Hebrew illustrate the positioning of the relevant elements in definite and indefinite environments:

\begin{verbatim}
חולים
beit xolim 'hospital'
house sick.PL

החולים
beit ha-xolim 'the hospital'
house the-sick.PL
\end{verbatim}

As a result of this configuration, a syntactic analysis may choose to regard the interposed article as attached to the entire complex noun around it (‘hospital’), rather than the modifier to which it is attached (‘sick’). A MAF/SynAF representation reflecting this analysis is given below. Note that the MAF representation contains a discontinuous word form to represent the word for ‘hospital’ and its lemma, as well as a word form uniting ‘sick’ with the orthographically attached article, but without a lemma (since ‘the sick’ has no lexical status as a lemma).

Example 6: MAF representation of the Hebrew construct state examples above.
Example 7: <tiger2/> representation of a constituent tree for the definite construct state.

This way, the definite NP behaves in a usual manner by joining a noun with an article, and the complexity of the noun is hidden in the MAF representation. Another option is to model the entire graph within the SynAF level, as in the following representation.

Example 8: <tiger2/> representation of the definite construct state without a MAF layer.

In this representation there are two types of non-terminals and edges: a phrase node with primary dominance edges (‘prim’), and a construct node with ‘const’ edges to its constituents. These must be declared in the annotation block and may carry different annotations, e.g. the lemma for construct states.

4.3 Zulu

4.3.1 Zulu agreement and writing system, encoding in MAF

The Bantu language family, of which Zulu is a member, is characterized by an agreement system based on noun classification (Nurse and Philippson, 2003:103). The basic word order in Zulu is SVO, although this availability of grammatical agreement allows a fairly unrestricted word order. In Zulu, a verb hence usually contains a prefix that cross-references it to the subject noun. Zulu is a language written conjunctively, i.e. linguistic units are merged to morpho-phonetic units on the surface.
4.3.2 Cross-referencing noun phrases in verb phrases

In the example sentence shown in Table 1, the *wa-* prefix in the verb *wabuya* agrees with the subject noun *umfana*. Both are of noun class 1, which usually indicates that the noun is referring to a singular human. In Table 1, the first line shows the surface sentence, the second its underlying morphs, and the third line demonstrates its part-of-speech (or rather morph) categories.

<table>
<thead>
<tr>
<th>Umfana</th>
<th>wabuya</th>
<th>kodwa</th>
<th>akayisizanga</th>
</tr>
</thead>
<tbody>
<tr>
<td>class 1</td>
<td>noun</td>
<td>subj-3rd-cl1 prefix</td>
<td>neg-subj-3rd-cl1 prefix</td>
</tr>
<tr>
<td>nom. marker</td>
<td>stem</td>
<td>past tense</td>
<td>verb ending</td>
</tr>
<tr>
<td>boy</td>
<td>return</td>
<td>but</td>
<td>(he) her assist not</td>
</tr>
</tbody>
</table>

Table 1: A brief description of the Zulu sentence 'Umefana wabuya kodwa akayisizanga' (The boy returned but he did not assist her)

While subject-verb agreement is obligatory, the verb-object phrase only shows grammatical agreement in certain cases, for instance, when the referent of the object is known information in the discourse and therefore omitted, as illustrated in the example sentence. The object-affix –yi- in *akayisizanga* is an example of pronominal incorporation in the verb, since it refers to the class 9 noun *intombazane* (girl), a discourse topic that has already been established. Note that, although it is a dependent morpheme, this object affix carries the grammatical function of an object (see also Bresnan, 2000:95 on Kichaga, another Bantu-language).

4.3.3 Inflection (negation and tense indication)

One way to negate a sentence in Zulu is to modify the verb. In this case, the sentence is negated by using two verbal affixes, i.e. the prefix *aka-* and the suffix *-anga* which also indicates past tense (*-i* would have marked present tense). In the first verb, *wa-buya*, past tense is indicated by the class 1 prefix *wa*; here, present tense would have been indicated by the prefix *u*.

4.3.4 Compound sentences

The sentence in Table 1 is an example of a compound sentence that links clauses of equal importance. In this case, the co-ordination is achieved by means of the co-ordinating conjunction *kodwa* (but). Our suggested encoding in MAF (the token definitions are found in Example 9) reflects the morphosyntactic facts: some of the units consisting of several morphs are formed on word level (see Example 10), while others form the predicate, i.e. a verbal phrase containing an agreement marker, an object, a verb stem and an inflectional element, cf. (Example 11).

Example 9: Excerpt of the MAF-encoding: token definitions

```xml
<token join="right" xml:id="t1" form="Um"/> <!-- um: noun class prefix class 1-->
<token xml:id="t2" form="fana"/> <!-- fana: noun stem common (boy)-->
<token join="right" xml:id="t3" form="wa"/> <!-- wa: subject prefix class 1-->
<token xml:id="t4" form="buya"/> <!-- buya: verb root (return) -->
<token join="left" xml:id="t5" form="a"/> <!-- -a: verb ending/extension -->
<token xml:id="t6" form="kodwa"/> <!-- kodwa: conjunction (but) -->
<token join="right" xml:id="t7" form="aka"/>
<token join="right" xml:id="t8" form="yi"/>
<token xml:id="t9" form="siz"/>
<token join="left" xml:id="t10" form="anga"/>
<token xml:id="t11" form=".">

Example 10: Excerpt of the MAF-encoding: token definitions

```
Example 10: Excerpt of the MAF-encoding: morphs forming a noun

Example 11: Excerpt of the MAF-encoding: units of the predicate

4.3.5 Encoding in SynAF

For space reasons, we focus on two of the phenomena described above: defining an anaphoric target in the second clause (see Example 12), and defining the predicate as a syntactic unit (a VP, see Example 13) formed by surface linguistic tokens, some of which are dependent morphs and others are free morphemes, i.e. lexical units.

Example 12: Excerpts of our SynAF-encoding demonstrating the capability of the standard to represent anaphoric elements, non-existent on the surface.
4.4 Chinese

The following encodings illustrate the treatment of an example sentence in Chinese. The sentence reads “她们正在学习古代汉语。” ("They are currently studying classical Chinese."). It contains a total of 11 Chinese characters including the sentence-end full stop, which are segmented into 6 lexical units as terminals. Syntactically speaking, the string, like most other sentences in Chinese, exhibits an SVO construction, with “她们” (They) as the subject NP, “学习” (studying) as the verb and “古代汉语” (classical Chinese) as the direct object NP. “正在” (currently) is analyzed as an adverb, forming part of the VP.

There is no explicit tense marking in the sentence. The progressive aspect is lexically expressed through the adverb “正在” (currently), reaffirming the fact that there is no morphological change for the Chinese verb. As a matter of fact, except for a few dependent morphemes indicating plurality, Chinese nouns do not have inflectional changes. In this sense, the attribute “lemma” in <terminals> below is not really necessary. In addition, the plural personal pronoun “她们” (they) is feminine, illustrating one of the very few personal pronouns carrying gender information. As a general rule, there is no gender marking for Chinese nouns. It is thus questionable whether gender marking should be a default attribute. In the current example, such marking is omitted for the sake of simplicity and clarity.

<s xml:id="s1">
<graph root="s1_ROOT" discontinuous="true">
<terminals>
<!-- aka-yi-siz-anga -->
</terminals>
</graph>
</s>
Example 14: example sentence in Chinese

4.5 Korean

Korean is known to be an agglutinative head-final language, much like Japanese (see Chang (1996) and Sohn (1999) for details).

4.5.1 Agglutination

Basic sentential segments, separated by whitespace, are called *eojeol* in Korean. Each *eojeol* is formed through a series of so-called *agglutination* processes. Each agglutination process concatenates a stem with a suffix, either nominal or predicate, and then this process may repeat, as in \[[[타]verbStem-[^ㅆ]pas^stem[^던]rx]eojeol.\]

미아가 탔던 차

Mia+NOM use+PAS+RX car

‘car (which) Mia used to ride’

where NOM is a nominative case marker, PAS a past tense marker, and RX a relativizing (retrospective) suffix, called *adnominalizer*.

This string consists of three *eojeol*: (eo1) 미아가, (eo2) 탔던, and (eo3) 차. Some *eojeol*, such as (eo2) and (eo3), are word forms by themselves, whereas the first *eojeol* (eo1) is segmented into two word forms, 미아 ‘Mia’ and -가, a nominative case marker. Each word form is then segmented into one or more tokens.

The process of segmenting a textual string into tokens or that of combining them into word forms can be represented in conformance to MAF:

<s xml:id="s1" lang="kr">
  미아가 탔던 차
</s>

<maf>
  <token xml:id="s1_tk1">미아</token> ‘Mia’ -->
  <token xml:id="s1_tk2" join="left">가</token> nominative case marker -->
  <token xml:id="s1_tk3">타</token> 'ride' -->
  <token xml:id="s1_tk4" join="overlap">ㅆ</token> past tense marker -->
  <token xml:id="s1_tk5" join="left">던</token> relativizing suffix -->
  <token xml:id="s1_tk6">차</token> 'car' -->
  <wordForm xml:id="s1_wf1" lemma="미아" tag="#pos.properName" tokens="s1_tk1" />
  <wordForm xml:id="s1_wf2" lemma="-가" tag="#pos.case_marker" tokens="s1_tk2" />
  <wordForm xml:id="s1_wf3" lemma="타다" form="טופ+RX" tag="#pos.verb" tokens="s1_tk3 s1_tk4 s1_tk5" />
  <wordForm xml:id="s1_wf4" lemma="차" tag="#pos.noun" tokens="s1_tk6" />
</maf>

Example 15: Tokens and Word Forms in MAF

4.5.2 Head-final Feature

Korean is an SOV language with predicate forms at the end of the clause. The primary data given above shows a relative clause construction ending in a verb form 탔던 ‘used to ride’ (relativized form) and followed by its nominal head 차 ‘car’. This syntactic formation can be represented as follows:

<s xml:id="s1">
  <graph>
    <terminals>
      <t xml:id="sl_t1" tiger2:corresp="korean.maf.xml#sl_wf1"/>
      <t xml:id="sl_t2" tiger2:corresp="korean.maf.xml#sl_wf2"/>
    </terminals>
  </graph>
</s>
Example 16: Relative Clause Construction in Korean

5 Generating <tiger2/> data automatically

One of the goals of the <tiger2/> development was not just to provide a new XML format, but also a human and machine readable metamodel and a processable API. For this purpose we used the Eclipse Modeling Framework (EMF). We generated the API in Java, because of its wide use and operating system interoperability. Since the core of the <tiger2/> metamodel is a general graph structure of nodes, edges and labels on both, it is possible, to access the <tiger2/> model with the usual graph processing mechanisms such as traversal etc.

Another part of the <tiger2/> development was its compatibility to the TigerXML model, i.e making sure that existing data in TigerXML can be converted to <tiger2/> without loss of information. To test this, we implemented a mapping from TigerXML to the <tiger2/> API. This allows us to import already existing data in TigerXML into the API and export it to <tiger2/>, and vice versa, though this may lead to information losses in some cases. Testing native TigerXML files and converting them back and forth, we can ensure no information is lost.

Another interesting prospect is converting data from other graph-based formats into <tiger2/>. We therefore used the SaltNPepper framework (Zipser & Romary, 2010), a universal importer to convert a wide range of formats into each other. Pepper is a plug-in based converter which uses the intermediate metamodel Salt to make a direct conversion between several formats. Using SaltNPepper and the <tiger2/> API we created a module mapping <tiger2/> to the Salt metamodel which we plugged into the framework. This step has allowed us to benefit from all already existing SaltNPepper modules and to convert data e.g. from and into the CoNLL (http://ilk.uvt.nl/conll/#dataformat) format, the PAULA XML format (Dipper, 2005), the GrAF format (Ide, & Suderman, 2007) and more.

The CoNLL format, is a field-based format largely used in international parsing evaluations. It is dependency-based, usually restricted to projective dependencies, with several fields such as FORM, LEMMA, CPOSTAG, POSTAG, FEATS, etc. A conversion to <tiger2/> is rather straightforward, with no need for nt elements (see Example 17 and 18).

<table>
<thead>
<tr>
<th></th>
<th>il</th>
<th>il</th>
<th>CL</th>
<th>CLS</th>
<th>2</th>
<th>suj</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>mange</td>
<td>manger</td>
<td>V</td>
<td>V</td>
<td>0</td>
<td>root</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>une</td>
<td>un</td>
<td>D</td>
<td>DET</td>
<td>4</td>
<td>det</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>pomme</td>
<td>pomme</td>
<td>N</td>
<td>NC</td>
<td>2</td>
<td>obj</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>.</td>
<td>.</td>
<td>PONCT</td>
<td>PONCT</td>
<td>2</td>
<td>ponct</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example 17: CoNLL output for “he eats an apple”

```xml
<t form="il" lemma="il" cpostag="CL" postag="CLS" xml:id="1">
  <edge label="suj" target="2"/>
</t>
<t form="mange" lemma="manger" cpostag="V" postag="V"/>
```

Example 18: Fragment of a possible <tiger2/> representation for CoNLL

When using the CoNLL format, some problems may arise from the fact that it does not follow the two level segmentation model of MAF (with tokens and word forms), leading to compound POS such as P+D for the agglutinate des (‘de’ + ‘les’ = of the).

In a second experiment, a dataset was converted from the Passage format (Paroubek & al. 2009) produced by the French parser FRMG (Villemonte de la Clergerie, 2010) into the <tiger2/> format. Though there is no module for SaltNPepper supporting the Passage format at present, a dedicated <tiger2/> generator was constructed for this purpose. The Perl conversion script was easily developed, and, while it remains to test it on complex cases, it demonstrates the potential of <tiger2/> to encode relatively complex syntactic annotation formats.

The Passage format was developed for the French evaluation campaign also called Passage (http://atoll.inria.fr/passage/). Accordingly, the Passage format is currently produced by several French parsers, and is relatively rich in information. It is a successor of the EASy format, extended to better conform the recommendations of ISO TC37 SC4, in particular with the MAF format (at the tokenization level with T and W elements) and with the use of ISO data categories. Passage is both constituency-based, with 6 kinds of chunks (called G) and dependency-based, with 14 kinds of relations anchored by either word forms W or chunks G. The T and W elements are straightforwardly converted into MAF elements and a correspondence (with attribute @corresp) is established at the level of <tiger2/> elements t towards the MAF word forms. The chunks become nt elements and the relations become edge elements either within t or nt elements. However, strictly speaking, Passage relations are not oriented (since they entail no explicit notion of governor and governor), but role-based. Furthermore, the COORD relation (for the coordinations) is ternary and has 3 roles, namely coordonnant (coordinator), coord-g (left coordinated) and coord-d (right coordinated). To fit in the <tiger2/> metamodel, it was therefore needed to orient the relations (by choosing a governor and orienting the edge from the governor to its governor) and binarize the COORD relation (using the coordinator as governor). Fortunately, no information is lost in the process. This is achieved using a @label attribute on edge, which combines the name of the relation with a role name (such as the SUJ-V sujet label for the subject in a SUJ-V relation). Constituency is represented by edges labelled comp within nt elements. Finally, the additional information carried by Passage W elements, such as form, lemma and mstag, are moved to the MAF wordForm elements, with a conversion of the mstag flat feature structure (using tags) into a deep expanded feature structure (based on the FSR standard).

Example 21 shows the original Passage fragment and Example 19 and 20 the corresponding representation in MAF and <tiger2/>.

```
<maf>
  <token xml:id="E1T1">ce</token> <!-- this -->
  <token xml:id="E1T2">matin</token> <!-- morning -->
  <token xml:id="E1T3"></token> <!-- , -->
  <token xml:id="E1T4">il</token> <!-- he -->
  <token xml:id="E1T5">a</token> <!-- has -->
</maf>
```

FRMG may be tried online at http://alpage.inria.fr/parserdemo with the possible production of 3 formats (DepXML, Passage, CoNLL). The <tiger2/> format should be soon added, thanks to the script presented in this paper.
Example 19: Fragment of the MAF part (sample_passage.maf.xml) produced from PASSAGE
Example 20: Fragment of the <tiger2/> part (sample_passage.tiger2.xml) produced from PASSAGE and relying on the MAF part and an external PASSAGE tagset (PassageTAGSET.xml), which specifies the possible features (pos, cat, lemma, form, ...) and values (in particular for the edge labels and for the part-of-speech).

Example 21: Fragment of the original PASSAGE file used to produce the MAF and <tiger2/> parts.

6 Next steps

Even if the <tiger2/> proposal, building upon the already quite mature TigerXML format, does already present all the technical features to cover the characteristics of the SynAF metamodel, making it an ISO standard, usable for the largest possible scientific community will require some additional efforts. In particular, beyond the provision of simple examples, we will need to put <tiger2/> to the test by confronting it with large-scale treebanks as they are available in the various languages tackled in this paper. In particular, this will bring more ideas as to the data categories that should be added to ISOCat in order to get a fully-fledged environment for the description of annotation schemes for syntactic data.

7 References


