ALVIS format of annotations
Task WP3.2
DRAFT - version 2.0

Guillaume Vauvert

October 22, 2004

Contents

Questions

This paper proposes a format of annotations and raises several issues:

1. About schemas (section 3.3): What is the more suitable language of schema (XML Schema, Relax NG) ? What tool to use to validate XML documents ? What's about our idea to use meta-schema ? How to validate partially annotated documents (partially may mean a part of the text, or a part of layers of annotations) ? How to organize partial schemas for partial validation ?

2. About processing: We assumed that a corpus is processed document by document. Do you validate this hypothesis ? Is it OK for statistical processing ?

3. About annotation content (section 1.3): What kind of semantic information do you need ?

4. About correction (section 3.2): Is our proposition enough powerfull and not too complex ?

5. About data volume (section 4.1): Be aware that this format considerably increases the document size. Is it OK ? Could compressed documents be used instead ?

6. About roles in Alvis annotating task (section **??**): WP5 and WP7 will do the following tasks: tagging of tokens, words, sentences, sections, paragraphs, titles, abstracts. From the Copenhaguen discussion, we assumed that WP7 is in charge of the structural annotations: sections, paragraphs, titles and abstracts (that requires to access to the original document). We propose that WP7 also assures tokenization (according to our definition, it is an automatic and not ambiguous process). The linguistic annotations must be processed in the following order: tokenization, named entities, other words, sentences. WP5 makes tag the words (which requires linguistic ressources). The main issue of tagging the sentences is to determine if a point (".") joined to a sequence of letters is a part of the word (Mr., www.ibm.com) or if it is a full stop marker. Tagging marks words or to tag sentences is the same issue: to determine the role of the point ".".

So, to make both in the same time is probably the best solution. Among these NLP steps, the precise role of WP7 and WP5 is not clear yet.

7. About the “light versions” of annotated documents (section **??**): How to access quickly (without parsing all the document) the desired annotations ? For validation, see “About schemas” above.

Chapter 1

Introduction

From a Natural Language Processing point of view, Alvis is mainly a process of annotation. Some packages add annotations (WP5 and WP7), some other use them for their tasks (WP2 and WP3 for classification and retrieval, WP6 for acquisition). So, the format of annotation is a central problem in Alvis.

1.1 Our goal

Our goal is to provide a format of annotation (called “alvis format”) that enables tools to exchange annotations. Actually, each tool provides some annotations in a *adhoc* format that may vary.
Tools are assumed to be impossible (close-source tools) or hardly (compexity of tools) to modify. So, the format translation must be transparent for them. Here is the annotation process we propose (see figure 1.1):

1. some annotations (that refer to a particular text) are available as input for a tool in “alvis format”;
2. a subset of annotations is selected:only a sub part of the annotation may be required (a level of annotation, a part of the text);
3. it is translated (to be understood by the tool) into the tool’s input format; some informations needed for further alignment may also be stored;
4. the tool processes the annotations;
5. it produces a new separate set of annotations in the tool’s output format (or a set of annotations that are mixed with the input annotations);
6. this output is translated into the “alvis format”, eventually using the alignment information stored before;

7. this output in “alvis format” is merged with the input in order to provide a unique annotation document.

So, we need:

1. A *selector* that selects what informations have to be used in order to make a projection. It selects a subset of annotations (a layer) and/or a subset of the text (a part). It is parameterized by a request in a declarative language.

2. A *translator* makes translation from the “alvis format” to the tool’s input format or from the tool’s output format to the “alvis format”. It is parameterized by an augmented grammar that describe the rules of transltation in a declarative way. The input translator has also to store some informations about alignment in order to allow the output translator to align the output annotations with the text.

3. A *joinor* that merges two disjoined or not annotations : if we assume that the output annotations contains only ouput annotations, its job is to concatenate two files (of annotations); else it has to separate pure new annotations from already known annotations, and then, it can concatenate them to the input annotations.

1.2 Needs

First, this section shown what kind of annotations are used, and then what kind of formalism is required to represent them.

The format of annotation may be the same for all these WPs. Some WPs may require a subset of annotations: some layer of annotations, annotations on a part of the text. So, we will be able to provide a sub-format for these sub-annotations.

1.2.1 Kind of annotations

Annotational units

This section details the kind of annotational units that are needed.
Usually, a token is a semantic unit that may vary according to the needs. This sens is used in this section. However, we will define later a token as a sequence of charater of the same category (**??**).

Segmentation A segmentation delimits some part of text and defines what kind of text it is. It may be a token, a phrase, a sentence ...
Here are two segmentations in tokens:
EXAMPLE. the / gerE protein / inhibits / transcription / in vitro / of / the / sigK gene / encoding / sigmaK

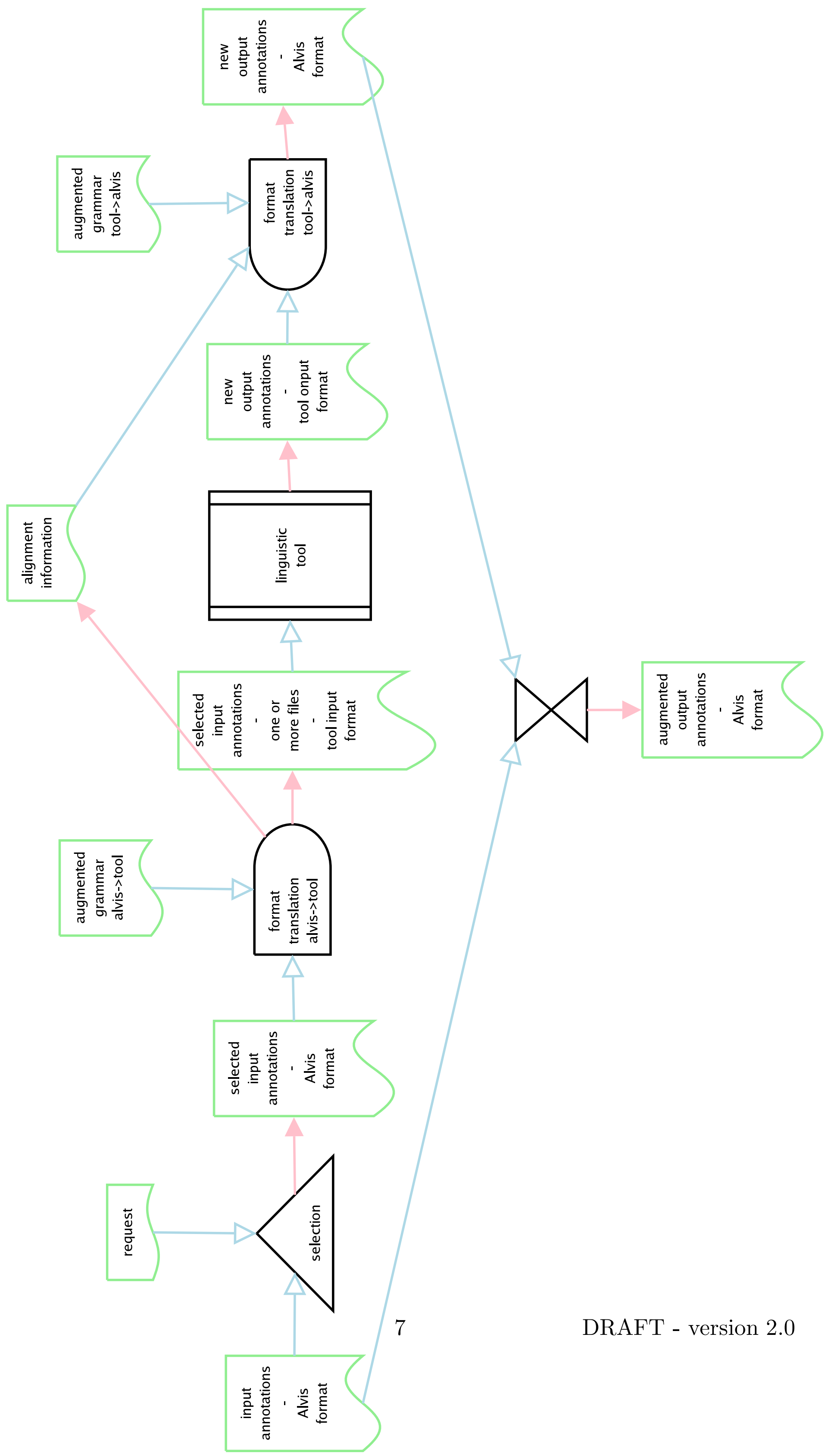

Figure 1.1: Architecture

EXAMPLE. the / gerE protein / inhibits / transcription in vitro / of / the / sigK gene / encoding / sigmaK

We may have multiple segmentations for the same document (but never all possible segmentations !).

Syntactic phrases An syntactic analysis provides a tree that represents the syntactic structure:
EXAMPLE. [the gerE protein] [inhibits [[transcription] [in vitro] [of [the sigK gene [encoding [sigmaK]]]]]]

Categories An annotation provides an information about a token:either syntactic (noun, verb, noun phrase....) or semantic (gene, action, person...).
There are categories of various granularity: small very specialised categories (synonymy or graphical variation as in sigmaK=Sigma K) or larger ones (person).
These categories may be hierarchically organized (for instance, a given unit can be a gene and a biological entity).

Discontinuous annotations Some annotations may have to annotate some discontinuous portions of text.
EXAMPLE. In french:
Je ne bouge pas.
(I do not move).
"ne ... pas" is the representation of the negation.

Co-occurrence If a token refers to a real-world entity, and another token refers to this token, then both tokens refers to the same real-world entity. However, the first token (that may be after the second) plays a particular role, because it defines the real-world entity. In this case, there is a a co-occurrence.
EXAMPLE. The cat eats the mouse.

$$Thecat$$

(agent)

$$eats$$

(object)

$$themouse$$

.

Punctuation Some symbols that are neither letter nor digits play a role in the text representation. They have to be taken into account in annotations.

Relations An annotation is not isolated, but in relation with some others. Most of time, one of the annotation plays a central role and may host the relation. But, sometimes, it is not the case. For example when two tokens are synonymous.

Way of annotation

Reliability Automatic process may rarely provide annotation without doubt. Most of time, it is correct with a degree of reliability.
Neither the limits of words/sentences/..., nor the annotations are known correctly and immediately (in fact, they are never known surely).

Partial annotation A part of the text is often selected before the annotational process begin. One determines which part will be analysed and which will not.

Granularity The granularity of annotations may vary, because some part of the text may required to be more or less deeply analyzed.

Concurrent annotation A tool may provide several annotations of the same type : for example, several syntactic structures are proposed, each one with a degree of reliability.
Two tools may provide the same type of annotation, but different annotations. For example, two tools may provide two different segmentation, each one based on a different information and on its own algorithm.

1.3 Some annotations

Linguistic annotation rely on entities and relations.

Basic linguistic entities The basic linguistic entities to encode are

- tokens. Tokens are not really linguistic entities. They result from the basic segmentation of the textual signal. They are designed for reference purposes (see below).
- Words. Basic linguistic entities corresponding to a given definition of what a word is (even if there is no definitive definition). A word may have the following properties:
  - Variation: any spelling correction
  - Lemma: the canonical form of the word (infinitive form for a verb; singular form for a noun...)
  - Semantics: the semantic information of a word may be of the following types (non-exclusively):

- * The word is a term, with a canonical form (to encore the regular spelling of naming form) and one or several source (to mention the certified terminology(ies) from which the terminological knowledge comes from).
  * The word is a named entity, with a canonical form (to encore the regular spelling of naming form) and one or several source (to mention the gazetter(s) from which the terminological knowledge comes from).
  * The word has one or several semantic types (which may be hierarchically structured). This must be further discussed and refined.
- Syntax:
  * The syntactic category of the word (part of speech: noun, verb, determinant...).
  * The morpho-syntactic features (plural/singular, tense...) of the word, the relevant features depending on the syntactic category of the word.
- Morphology: the morphologival information describes the internal structure of the word, either full or partial decomposition in a list of morphemes (root form, prefix, sufixes..) and gives its stem.

• Phrases. A phrase (noun phrase, adjectival phrase...) may have the following properties (non-exclusively):
  - Semantics:
    * Terminology: the same as for nouns.
    * Named entity: the same as for nouns.
    * Semantic types: the same as for nouns.
  - Syntax:
    * Morpho-syntactic features: (plural/singular, tense...) of the phrase, the relevant features depending on the syntactic category of the phrase.
    * Syntactic category: noun phrase, verbe phrase, clause...
    * Syntactic composition: the list of the phrase constituents, which are either phrases or words.

• Document structure elements:
  - Sentences. Sentences have the same structure as phrases. They could be considered as special types of phrases but they are often identified on a very surfarcical base whereas identifying phrases requires a syntactic analysis.
  - Other elements such as paragraphs, abstract, title, section... (to be further refined).
  - Eventually typographical elements such as italics (to be discussed).

Basic linguistic relations In addition to the above entities, it is necessary to encode some linguistic relations:

- Semantic relations: semantic roles between two words or phrases (in biology: agent or target of an interaction).
- Anaphora relations: relations that rely a pronoun (sometimes a noun phrase) to its antecedent as 'X' and 'its' in "the X protein and its regulator".
- Syntactic relations: dependancy relations that hold between two words or phrases (object, subject, modifier relations...).

NL Processing annotations It is important to keep track of the various processing steps that have been performed to obtain the described set of annotations. The following pieces of information must be stored, for each processing step:

- Type and name of the tool that has been used: segmentation in phrases... These types must be defined according to the specification of the NLP lines in various languages.
- Type and name of the resources that have been used.
- Segment of the document that has been annotated (token interval(s).
- List of types of the annotation entities and relation that have been added.
- Degree of reliability of the processing step.

Chapter 2

Related works

Researches in language field have produced a lot of formats of annotation. Over several hundreds bases have been built during the last 50 years [BL99]. Each base has been created for a specific goal, in an adequat format, near to the standards of the field. To adapt the tools for operations (creation, update, indexing, research, and visualization), requires to re-build some code.

2.1 Existing formats

To study the existing formats provides:

- ideas of interesting formats;
- a certain pallet of what it is necessary to represent;
- main principles of design.

Many information come from [BL99] and from its cited articles. It acts primarily of work based on the analysis of the speech, which could be seen as a defect (the problems seem different); but in fact, that obliges to separate the rough signal (for us, the text) from its annotations. Sometimes, the text is regarded as a sequence of words; but this level requires analysis that determines semantics of charaters. In fact, a texte is a sequence of character as a speech sample is a sequence of sampled frequencies. From this point of view, it is natural to use stand-off annotation (other reasons are provided in section 2.2).

GATE 1

In the user's documentation of GATE 2.1, one finds, page 71, of the examples of annotated documents.
GATE 1 is based on TIPSTER [Gri97] (see 2.1).
It thus acts of a table each line contains:

- an identifier of the line
- the type of the object
- the starting and end indexes of the interval
- annotations specific to the type of the object

Examples of annotations:

- token: the type of token (name, determiner...)
- name: type of name (of a person...)
- sentence: components

This formalism doesn't allow to represent all our needs, but it is interesting because it is a rare stand-off annotation.

GATE 2

GATE 2 is based on annotated graphs [BL99] (see section **??**), implemented in XML.

TIMIT

See [GLW93].
It is based on a specific data base, the first on the annotated word.
The format simple and clear, but is limited:

```
index\_start index\_end word
index\_start index\_end phonetic\_transcription\_
```

Each word is broken up into syllables, each one with its phonetic transcription.

Partitur

See [SBGW98].
Goal:

> An open (that is extensible), robust format to represent results from many different research labs in a common source.

Features:

- time-alignement
- multilevel description, links between units of the various levels independently of time

Description centered on “format of files”.

CHILDES

See [Whi95].
The greater scientifique database.
Format “CHAT”.
Not interesting.

LACITO

See [JML01]
Files with format XML; the words are separate and related to their French translation (integrated glossary), then the translation of the sentence is given in french and english. Only the beginnings and end of sentence are referred (temporally).

LCD Telephone Speech

Voice recognition: telephone conversation.
Whole sentences are translated (not the word for word one). There is a partial or total covering of sentences and thus of annotations.

NIST UTF

[NIS98]
Linguistic field: conversation
Goal:

> intended to provide an extensible universal format for transcription and annotation across many spoken language technology evaluation

Based on SGML, it must make it possible to represent the transcriptions and to be extensible.
The covering of the dialogue is managed by the separation of the “turns”. The beginnings and end of overlaps are noted in the turns.

Switchboard

[GHM92]
About the formats, see 2.1 and 2.1.
To read:[GB00]
Several annotations about the same corpus:

- Location of the words: beginning, end and transcription
- Part of speech: reference to the words by the word itself, in the order of the text
- Disfluency: the sentence is repeated to be annotated

- Treebank: the annotation is hierarchical (it is a tree); the morpho-syntactic categories and syntactic structure DNA are mixed in a structure XML-linke (in fact LISP-like). The words are also rewritten.

MUC-7 Coreference

[HC97]
Specialized in the extraction of information, in particular the location of entites and the coreference.
The initial text is marked out. The sections implied in the coreferences are taged and IDed to be REFed (XML). A TYPE attribute indicates the type of relation, the identity, which gives classes of equivalence (classes of equivalence are also found in **??**.

Emu

[CH00]
Speech database.
Hierarchical annotations in tables on several levels. A level is a linear order. The levels and the relations are customisables.

Festival

See [TBC00]
Applicability: synthesis of the voice.
Structure: "Heterogeneous Graph Relation": set of binary relations on structures of characteristics SC (represented as matrices attribute-value). Each SC describes the local properties of linguistic units (segment, syllable, sentence). An attribute may have as value an atom, a SC or a function. Functions make it possible to cross binary relations, for example to incorporate values of other characteristics (e.g. for information temporal).

MATE

See [KIM$^+$00]

MATE is a dialog annotation workbench. Each layer is in a separate XML file. Each layer may be a sequence of words or embeded tags representing a hierarchy. The references are done by hyper-links (one-to-many by a tag containing a list of hyperlinks). The composition is managed either by the embeded tags, or by the hyperlinks between layers. All is very configurable.

Tex Encoding Initiative

[**?**]
Very broad, to study.
There are certainly good councils to extract from TEI, but all the annotations

are in pure XML (chapter 15 Simple Analytic Mechanisms), what means that the text itself is marked out (and thus polluted). Moreover, the structures are inevitably hierarchical, which is very reducing.

> More precisely, the sentence: `The victim's friends told police that Kruger drove into the quarry and never surfaced.`
> is tagged :
>
> ```
> <s>The&AT; victim&NN1;'s&GEN; friends&NN2; told&VVD; police&NN2;
> that&CST; Krueger&NP1; drove&VVD; into&II; the&AT; quarry&NN1;
> and&CC; never&RR; surfaced&VVD;.&PUN;</s>
> ```
>
> The names used for these entity references have some significance for the human reader (AT for article, NN1 for singular noun, NN2 for plural noun, etc.), but their representation in the output from a system processing the document may be adjusted by modifying the entity declarations to suit the convenience of whatever analytic software is to be used. For example, if a parser operating on this sentence uses a set of entity declarations in the following form, then the word class tags will simply disappear from the output.
>
> ```
> <!ENTITY AT "">
> <!ENTITY NN1 "">
> <!ENTITY GEN "">
> <!-- ... -->
> ```
>
> Alternatively, suppose the entity set in use follows the following pattern:
>
> ```
> <!ENTITY AT  "[definite article]">
> <!ENTITY NN1 "[singular noun]">
> <!ENTITY GEN "[genitive suffix]">
> <!-- ... -->
> ```
>
> Then the sample sentence will be processed as if it began:
>
> ```
> The[definite article] victim[singular noun]'s[genitive suffix] ...
> ```

Rmq:23.4.1 Levels of Analysis is a piece of 15.4 Linguistic Annotation

> Non-nesting information poses fundamental problems for any encoding scheme, and it must be stated at the outset that no solution has yet been suggested which combines all the desirable attributes of formal simplicity, capacity to represent all occurring or imaginable kinds of structures, suitability for formal or mechanical validation, and clear identity with the notations needed for simpler cases (i.e. cases where the textual features do nest properly). The representation of non-hierarchical information is thus necessarily a matter of choices among alternatives, of tradeoffs between various sets of

different advantages and disadvantages.
There are several methods used within these Guidelines to handle non-nesting information:

- concur: an optional feature of SGML (not available in XML) which allows multiple hierarchies to be marked up concurrently in the same document
- milestone elements: empty elements which mark the boundaries between elements in a non-nesting structure : ¡qb who=”coin”/¿... ¡qe who=”coin”/¿ difficult to process; ¡qb id=”coin”/¿... ¡qe corr=”coin”/¿ not uniform representation and not high validation
- fragmentation of an item: the division of what logically is a single element into two or more parts, each of which nests properly within its context : ¡p¿ ¡q who=”coin”¿... ¡/q¿ ¡/p¿ ¡p¿ ¡q who=”coin”¿... ¡/q¿ ¡/p¿ (q continue over paragraphes) one hierarchie plays a particular role and can be processed normally but some units are not realized at all in the markup and automatic processing of these units is thus impossible.
- virtual joins: the recreation of a virtual element from fragments of text, possibly discontiguous or out of order: ¡q id=”qw1”¿...¡/q¿ ... ¡join targets=”qw1 qw2 qw3” result=”q”/¿
- redundant encoding of information in multiple forms : used by ??? (seen at MIG)

TIPSTER

Tipster is based on annotated graphs [BL99].
The used annotated graph are interesting but limited. See **??**.

2.2 Stand-off annotations

There are many reasons why separating markup from the material marked up (”standoff annotation”) may be an attractive proposition:

+ the base material may be read-only and/or very large, so copying it to introduce markup may be unacceptable;
+ distribution of the base document may be controlled, but the markup is intended to be freely available;
+ it doesn’t pollute the textual signal : it is always possible to refer to the original text;

+ it allows not embeded annotations (it is possible to have levels of annotation which have crossing branches, what is not normally possible in XML), needed by :

  concurrent annotations produced by several tools or a single tool;

  not linear annotations : "to ... decide" in "to completely decide" and "ne ...pas" in "je ne mange pas" are relevant linguistic entities;

  relations (grammatical functions, biological relations, ...) may rely entities at several levels in the hierarchy;

  complex structures: some annotations may begin in a structure (a sentence) and end in another one (another sentence)

+ new levels of annotation can be added without disturbing existing ones;

+ editing one level of annotation has minimal knock-on effects on others;

+ people can work on different levels at the same time without worrying about creating different versions;

+ access to results of alternative NLP components, for flexible use in applications;

+ each level of annotation (syntactic, dialog act, anaphoric) is stored indepedently;

- ... but it is harder to follow the textual flow: it requires to define at least one sequence of annotations.

With stand-off annotations, it is possible to put levels of annotation in separate documents:

+ don't keep all your data in one big document;

+ one document for each annotation level;

= links between documents;

- has several files may be confusing (however, they can be tared in a single file).

XML-based annotation:

+ reproducable

+ easy to go back and to correct mistakes

+ saves time and unnecessary work using standard preprocessing software

+ reporting automaticaly - allows detailed error analysis

+ standoff annotation

+ XSLT queries to XML standoff annotations for flexible, efficient integration
- still a lot of work
- difficult to get overview because view restricted to window on the screen
- usually only one kind of annotation at one time

The text is regarded as a signal; so, it cannot be modified.

2.3 Standard of annotation

The annotation representation problem is not new. It has been widely studied since the begining of nineties and several *ad hoc* formats have been proposed. Efforts to unify these formats in order to allow interoperability among NLP tools is recent and has lead to two projects that are going to merge :

- the TEI that is an international and interdisciplinary standard that helps people to represent all kinds of literary and linguistic texts for online research and teaching, using an encoding scheme that is maximally expressive and minimally obsolescent;
- an ISO proposition (TC37SC4/TEI), composed by : Feature Structure Representation, Morphho-Syntactic Annotation Framework, Category Data Repositery, Linguistic Annotation Framework, Lexical Markup Framework, Data Category S-Electronic Lexical Ressources

Our proposition is based on the same ideas, but:

- current drafts of ISO provide principles that are far from definitive propositions; they may evolve in a significatively different way.
- Our goal is a subset of the more general goal of TC37SC4/TEI. To follow exactly the norm leads to a high complexity, whereas our need can be satisfied with a light version.

So, we decide to follow the principles of the TC37SC4/TEI (but we don't try to stick to their current proposition):

- Each annotations (tokens, words, phrases, sentences, function relation, biologic relation, ...)
- Each annotation is provided with references to some other annotations.

Chapter 3

Proposal

3.1 Token layer

It is necessary to have a unique and common reference. As shown in **??**, this reference must not require semantic analysis. For instance, bounds of sentences and bounds of words are not free from mistake. It could be:

- the character
  - [+] no semantics
  - [-] offsets are very large
- the words
  - [-] semantics
  - [-] the concept of word is not normalized
  - [-] so, the bounds of words are not unique
- the sentence, and words or characters
  - [-] no semantics
  - [-] the bounds of sentences are not known at the beginning

With characters, offset are very large, but with the next layer (word), a sematnic analysis is required. So, we propose to use an intermediary level, called the token level. A token is defined here without semantics as a sequence of charater of the same type. We propose to define four kind of tokens:

- `alpha`:sequence of letters (a-z and A-Z)
- `num`:sequence of digits (0-9)
- `sep`:sequence of separators (space, return, ...)
- `symb`:any other character

The level of token has been chosen by the workgroup TC37SC4/TEI too, but they decide to include "TEI element ptr" in order to mark places in the text. These marks are used to identify the token (between two "ptrs"). We decide to don't modify the original text.
The tokenization is the first stage of text analysis. Token are numbered from 1 for the first token. Then, all other annotations refer directly or indirectly to this layer. Ex: "The gerE protein inhibits transcription in vitro of the sigK gene encoding sigmaK."

```
<c_alpha>
  <id>token1</id>
  <from>0</from>
  <to>2</to>
  <cont>The</cont>
</c_alpha>
<c_sep>
  <id>token2</id>
  <from>3</from>
  <to>3</to>
  <cont> </cont>
</c_sep>
<c_alpha>
  <id>token3</id>
  <from>4</from>
  <to>7</to>
  <cont>gerE</cont>
</c_alpha>
<c_sep>
  <id>token4</id>
  <from>8</from>
  <to>8</to>
  <cont> </cont>
</c_sep>
<c_alpha>
  <id>token5</id>
  <from>9</from>
  <to>15</to>
  <cont>protein</cont>
</c_alpha>
<c_sep>
  <id>token6</id>
  <from>16</from>
  <to>16</to>
  <cont> </cont>
</c_sep>
<c_alpha>
  <id>token7</id>
  <from>17</from>
  <to>24</to>
  <cont>inhibits</cont>
</c_alpha>
<c_sep>
  <id>token8</id>
  <from>25</from>
  <to>25</to>
  <cont> </cont>
</c_sep>
<c_alpha>
  <id>token9</id>
  <from>26</from>
  <to>38</to>
  <cont>transcription</cont>
</c_alpha>
<c_sep>
  <id>token10</id>
  <from>39</from>
  <to>39</to>
  <cont> </cont>
</c_sep>
<c_alpha>
  <id>token11</id>
  <from>40</from>
  <to>41</to>
  <cont>in</cont>
</c_alpha>
<c_sep>
  <id>token12</id>
  <from>42</from>
  <to>42</to>
  <cont> </cont>
</c_sep>
<c_alpha>
  <id>token13</id>
  <from>43</from>
  <to>47</to>
  <cont>vitro</cont>
</c_alpha>
<c_sep>
  <id>token14</id>
  <from>48</from>
  <to>48</to>
  <cont> </cont>
</c_sep>
<c_alpha>
  <id>token15</id>
  <from>49</from>
  <to>50</to>
  <cont>of</cont>
</c_alpha>
<c_sep>
  <id>token16</id>
  <from>51</from>
  <to>51</to>
  <cont> </cont>
</c_sep>
<c_alpha>
  <id>token17</id>
  <from>52</from>
  <to>54</to>
  <cont>the</cont>
</c_alpha>
<c_sep>
  <id>token18</id>
  <from>55</from>
  <to>55</to>
  <cont> </cont>
</c_sep>
<c_alpha>
  <id>token19</id>
  <from>56</from>
```

```
  <to>59</to>
  <cont>sigK</cont>
</c_alpha>
<c_sep>
  <id>token20</id>
  <from>60</from>
  <to>60</to>
  <cont> </cont>
</c_sep>
<c_alpha>
  <id>token21</id>
  <from>61</from>
  <to>64</to>
  <cont>gene</cont>
</c_alpha>
<c_sep>
  <id>token22</id>
  <from>65</from>
  <to>65</to>
  <cont> </cont>
</c_sep>
<c_alpha>
  <id>token23</id>
  <from>66</from>
  <to>73</to>
  <cont>encoding</cont>
</c_alpha>
<c_sep>
  <id>token24</id>
  <from>74</from>
  <to>74</to>
  <cont> </cont>
</c_sep>
<c_alpha>
  <id>token25</id>
  <from>75</from>
  <to>80</to>
  <cont>sigmaK</cont>
</c_alpha>
<c_symb>
  <id>token26</id>
  <from>81</from>
  <to>81</to>
  <cont>.</cont>
</c_symb>
<c_sep>
  <id>token27</id>
  <from>82</from>
  <to>82</to>
  <cont>\n</cont>
</c_sep>
```

3.2 Corrections

Examples below are based on the following sentence that contains a mistake: the letter "i" between offsets 13 and 14 has been omitted (the correct word is protein):

```
0000000000111111111122222222223333333333...
01234567890123456789012345678901234567...
The gerE proten inhibits transcription...
```

Orthographic correction may be done:

1. before all NLP, the signal is corrected:

   the orginal signal is lost

   ```
   00000000001111111111222222222233333333333...
   012345678901234567890123456789012345678...
   The gerE protein inhibits transcription...
   ```

   the corrected signal is the single used signal

   a trace of correction is kept:

   zone [9,15] was "proten"

2. during NLP:

   the signal is not modified

   several words can be proposed for a single token

Example before NLP:

```
<correction>
  <id>correct1</id>
  <from>9</from> <!-- the new offsets -->
  <to>15</to>    <!-- the new offsets -->
  <was>proten</was> <!-- the old value mistake -->
</correction>
```

Example during NLP:
`word3` and `word33` are two possible words for the token `token5` ("proten").

```
    <c_alpha>
      <id>token5</id>
      <from>9</from>
      <to>14</to>
      <cont>proten</cont>
    </c_alpha>

    <c_word>
      <id>word3</id>
      <list-refid-token>
        <refid-token>token5</refid-token>
      </list-refid-token>
      <!--variante></variante-->
      <lemma>
        <canonical_form>protein</canonical_form>
        <source></source>
      </lemma>
      <semantics>
        <is_term>F</is_term>
        <is_named_entity>F</is_named_entity>
        <semantical_features>
          <object_category>protein</object_category>
        </semantical_features>
      </semantics>
      <morphology>
        <composition>protein</composition>
        <stem>protein</stem>
      </morphology>
      <morpho-syntactic>
        <category>noun</category>
        <morpho-syntactic_features>
          <number>singular</number>
        </morpho-syntactic_features>
      </morpho-syntactic>
```

```
</c_word>

<c_word>
  <id>word33</id>
  <list-refid-token>
    <refid-token>token5</refid-token>
  </list-refid-token>
  <!--variante></variante-->
  <lemma>
    <canonical_form>proton</canonical_form>
    <source></source>
  </lemma>
  <semantics>
    <is_term>F</is_term>
    <is_named_entity>F</is_named_entity>
    <semantical_features>
      <object_category>proton</object_category>
    </semantical_features>
  </semantics>
  <morphology>
    <composition>proton</composition>
    <stem>proton</stem>
  </morphology>
  <morpho-syntactic>
    <category>noun</category>
    <morpho-syntactic_features>
      <number>singular</number>
    </morpho-syntactic_features>
  </morpho-syntactic>
</c_word>
```

Another stage:

```
<c_alpha>
  <id>token5</id>
  <from>9</from>
  <to>15</to>
  <cont>protein</cont> <!-- corrected mistake -->
</c_alpha>
```

3.3 Schemas and schemas of schema

3.3.1 Principles

Some principles of annotation have been proposed:

- there is a finite set of annotations: concepts (token, word, phrase, sentence, ...) and relations (phrase function, biologic relation, ...)
- a concept or a relation is a structure with:

  an id field that contains a unique identifier;

  another fields with unique names (in this structure) which types are defined below.
- a hierarchy of objects may be defined

To represent these annotations, we need types that may be:

- basic types (or subtypes):number, string, refid (a reference to a unique identifier, a kind of pointer)
- list of data with the same type; ex:list of sizes, of addresses, ...
- set of data, each one with a type that is not the same than others

3.3.2 Schema of schema

These types have to be defined in an XML schema in order to validate XML documents.
Because of the complexity of XML schema languages, to write a schema is not easy. Moreover, the schema must follow the principles defined above, and that must be verified. So, we propose to allow the user to define its set of annotations through an XML file `def_docann.xml`. It will be validated by given XML schema `deftype.xsd` that represents the principles. It will then be automatically translated into a XML schema `docann.xsd`, that may validate an annotated document contained in an XML file `annotated_doc.xml`.
Figure 3.1 describes the way to define the language of annotations and to use it:

1. The principles proposed above are described in the schema `deftype.xsd`: it defines the expressivity power of the language. This schema should not change.
2. The particular set of authorized annotations that will be used are defined in an XML file `def_docann.xml` that must be validated by the previous schema `deftype.xsd`.
3. This XML file `def_docann.xml` is transformed automatically into a schema `docann.xsd` in order to validate annotations.
4. The schema `docann.xsd` allows to validate the annotated document contained in a XML file `annotated_doc.xml`.

To summarize, the `def_docann.xml` is the meta-schema (validated by `deftype.xsd`) that produces a schema `docann.xsd` used to validate the annotated document (for example `annotated_doc.xml`).

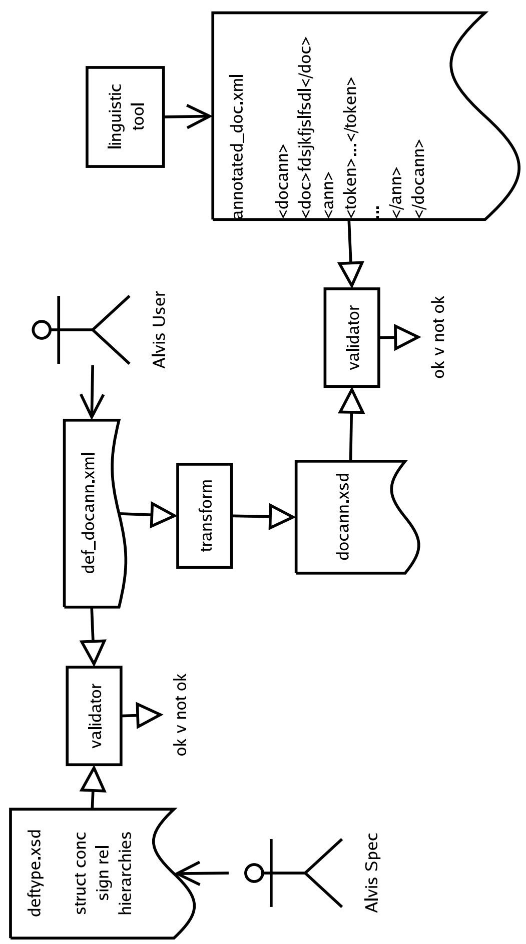

Figure 3.1: Schema and meta-schema

3.3.3 Selection of sub-part of document

(This subsection is not finalized. Several points are not clear.)
In section **??** and in figure **??** show the need of a selector. This section show how to validate this documents.
A subpart of an input annotation (provided by previous steps of NLP) may have to be chosen to produce a selected input annotation. This selection requires several documents (see figure 3.2):

1. `restriction.xlst`, that describes the way to transform the input annotation into the selected input annotation;
2. `docann_ia.xsd`, to validate the input annotation;
3. `docann_sia.xsd`, to validate the input annotation;
4. `def_docann_ia.xml` that describes `docann_ia.xsd`;
5. `def_docann_sia.xml` that describes `docann_sia.xsd`;

In fact, `def_docann_ia.xml` generates `docann_ia.xsd` and `def_docann_sia.xml` generates `docann_sia.xsd`.
The transformation from `input annotation` to `selected input annotation` and the transformation from `def_docann_ia.xml` to `def_docann_sia.xml` are similar. So, they could use the same description of the restriction (`restriction.???`).
Now, the difficulty is to write the two magic tools : the first that produce `restriction.xslt` using `restriction.???`; the second that tranform `def_docann_ia.xml` into `def_docann_sia.xml` using using `restriction.???`.

3.4 Additional remarks

1. Ambiguities are not explicit in the annotated document. There is an ambiguity if two different words (for instance, the noun 'ports' and the verb 'ports') refer to the same token or if two words or phrases refer to partially overlapping sequences of tokens (such as one is not included in the other).
2. Annotation errors. Annotations that have been added at a step i of NL processing may be deleted at step i+1. It may be necessary to keep track of deleted annotations so as to avoid circularity in processing.

3.5 Format

3.5.1 Annotations : separated vs embedded

The annotations are disjoined from the text (seen as a signal):

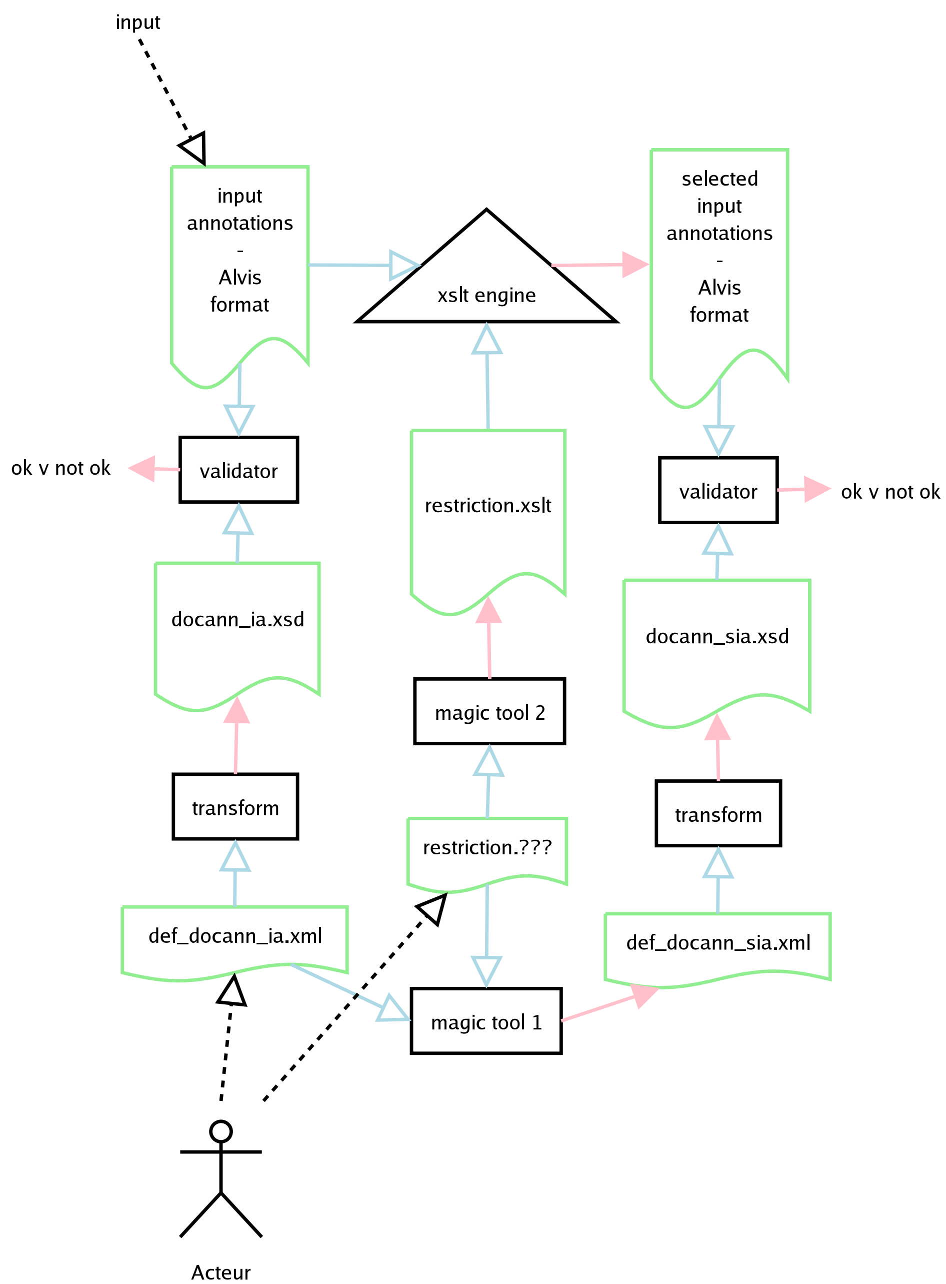

Figure 3.2: selector validation

(+) easy to represent all kinds of annotations (non hierarchical)

(-) difficult to generate a clean text (to discuss)

3.5.2 Annotations : Database vs XML vs dedicated format

How to implement the formalism ? Three main ways : Database:

(-)requires a middleware (to access the database), what make it more complex to access data

(-) is designed for simultaneous access of persistent data. we do not need simultaneous access. over complex.

(-) add a network layer that speed down the access (required for simultaneous access)

(-) relationel database: the efficiency is provided by the use of rectangular tables, attributes and values. it is not easy to use with very different and irrgular formats

Intermediate solution:
storage in database, generate xml when required.
XML with tagged text:

(-) non hierarchical data very complex to represent and to process

(+) humanly readable, but why ? manual correction ? perly easily readable ?

(+) tools to process XML

(+) validation

(+) usage

XML could be use without tagging text, by using pointers on the contents. It is not certain that it is the most adequate format, especially because many tools existing will not be usable. But it acts of a format perennial, verifiable, for which one can develop specific tools while resting on API what exists (DOM, SAX...).
Dedicated format:

(+) may represent non hierachical annotations

(-) may be seen as another format

(+) optimize to our needs

(-) require tools to access

(-) require tools to

3.5.3 Human readability

A software is always necessary to access to a document, even if it may be a common word processor. It may be sufficient/enough for a small amount of data, but when it becomes too large, a special software is necessary. The human readability may allow a user to modify the data directly. It is often the case when an operation has not been implemanted.
What is necessary, it is a software-easy-processability, universal, without the need of a middleware (a perlability ?).
There is another problem, the compacity of the annotation.

3.5.4 XML

XML is a rich language which makes it possible to represent clarly complex structures, not always very regular, but these structures must be hierarchical (in standard use, i.e. in annotating texts with tags).
The documents can be validated, initially by the compliance with the rules of syntax XML (approximately, a correct beaconing), and then of grammar by a DTD or XShema (better conception).
Thanks to XPointer, one can reach any point of any document.
With XSL/XSLT, one can transform document in XML, either to make it visible under certain other forms (most promising being XFO), or to transform it into another document XML (handling of arborescent stucture).
We will see later how to use XML to represent our formalism. One can either be in particular satisfied with the standard tools (XSLT primarily) to ensure the handling of our data, or to conceive an application which will carry out particular actions according to certain tags or attributes.

3.5.5 XML-like vs DB

XML allow to manipulate/process data without intermediate layer:perl without a DB interface. To add new annotations is easy: some lines in the DTD vs the add of several tables in the DB. Moreover, the structure is more flexible : data may be irregular, since in DB, the only available object is the table, that is rectangle and regular.
However, the process of DB is very fast, thanks to the regularity of structures of data.

3.6 Some examples

See examples at the end of the document, annex A.

Chapter 4

Conclusion

4.1 Data size

An annotated document is much highter than the original document. The figure 4.1 provides some sizes. The first colum “Document” is the name of the original document. The second column is the level of annotated layer: “token” means token only, “word” means token and word, “sentence” means token and word and sentence, “all” means token and word and sentence and phrases (lemma, semantics, syntax) and functional relations and biologic relations. The third column is the size of the document (without annotation). The fourth column is the size of the annotated document, and the fifth, the ratio = size of annotated document / size of document (the growth of size). The seventh column is the size of the gzipped annotated document, and the fifth, the ratio = size of gzipped annotated document / size of document (the growth of size with gzip). The last column is the ratio of the two sizes of annotated document, non gzipped / gzipped.
The six first rows provide sizes for a corpus of a single sentence that has been annotated by hand. The layer “All-min” is the same anntotation than the layer “All”, but named have been simplified in order to save space (however, it is no more humanly readable).
The four next rows provide sizes for a corpus of carnivor plants. Values of the layer “All (extrapolated)” are extrapolated from values “1 sentence / All” and values of “Carnivor / Sentence”. Compression ratio are not significative for small sizes. As supposed, to simplify the names of elements does not significatively decrease the size of the gzipped file (compare “All” and “All-min”).
Between “1 sentence” and “carnivor”, non-gzipped ratio are similar for token (43), words (60), sentences (70). Extrapolating these results leads to the “all layer” annotations provides the last line. A document which size is 3.1Mo produces an annotated document which size is around 880Mo (ratio 282).
This size may be divided by 2 by using more short names than those chosen, and to remove all non-significant spaces and end-of-lines.

Document	Layer	Size	Non-Gzipped		Gzipped		Compression
			Size	Ratio	Size	Ratio	ratio
1 Sentence	Token	83	3472	41,83	670	8,07	5,18
	Word	83	5557	66,95	826	9,95	6,73
	Sentence	83	6160	74,22	982	11,83	6,27
	All	83	23438	282,39	2126	25,61	11,02
	All-min	83	6938	82,6	1368	16,29	5,07
Carnivor	Token	3104488	137750409	44,37	12565647	4,05	10,96
	Word	3104488	171848400	55,35	16009125	5,16	10,73
	Sentence	3104488	202894549	65,36	17385830	5,6	11,67
	All (extrapolated)	3104488	876662527	282,39	75120331	24,2	11,67

Figure 4.1: Sizes of annotated documents

However, rich annotations are required only for acquisition (WP5 and WP6), and acquisition use small volumes of documents. For information retrieval, data volume is large, but there is less annotation (there is no syntactic analysis). The sizes of annotations in real process will be smaller than those provided here.

4.2 Questions

This paper proposes a format of annotations and raises several issues:

1. About schemas (section 3.3): What is the more suitable language of schema (XML Schema, Relax NG) ? What tool to use to validate XML documents ? What's about our idea to use meta-schema ? How to validate partially annotated documents (partially may mean a part of the text, or a part of layers of annotations) ? How to organize partial schemas for partial validation ?

2. About processing: We assumed that a corpus is processed document by document. Do you validate this hypothesis ? Is it OK for statistical processing ?

3. About annotation content (section 1.3): What kind of semantic information do you need ?

4. About correction (section 3.2): Is our proposition enough powerfull and not too complex ?

5. About data volume (section 4.1): Be aware that this format considerably increases the document size. Is it OK ? Could compressed documents be used instead ?

6. About roles in Alvis annotating task (section **??**): WP5 and WP7 will do the following tasks: tagging of tokens, words, sentences, sections, paragraphs, titles, abstracts. From the Copenhaguen discussion, we assumed that WP7 is in charge of the structural annotations: sections, paragraphs, titles and abstracts (that requires to access to the original document). We propose that WP7 also assures tokenization (according to our definition, it is an automatic and not ambiguous process). The linguistic annotations must be processed in the following order: tokenization, named entities, other words, sentences. WP5 makes tag the words (which requires linguistic ressources). The main issue of tagging the sentences is to determine if a point (".") joined to a sequence of letters is a part of the word (Mr., www.ibm.com) or if it is a full stop marker. Tagging marks words or to tag sentences is the same issue: to determine the role of the point ".". So, to make both in the same time is probably the best solution. Among these NLP steps, the precise role of WP7 and WP5 is not clear yet.

7. About the "light versions" of annotated documents (section **??**): How to access quickly (without parsing all the document) the desired annotations ? For validation, see "About schemas" above.

Bibliography

[BL99] Steven Bird and Mark Liberman. Annotation graphs as a framework for multidimensional linguistic data analysis. In NJ: Association for Computational Linguistics, editor, *Towards Standards and Tools for Discourse Tagging - Proceedings of the Workshop*, pages 1–10, Somerset, 1999.

[CH00] Steve Cassidy and Jonathan Harrington. Multi-level annotation of speech : an overview of the emu speech database management. In *Speech Communication*, 2000.

[GB00] David Graff and Steven Bird. Many uses, many annotations for large speech corpora: Switchboard and tdt as cas studies. In *Proceedings of the Second International Conference on Language Resources and Evaluation*, Paris : European Language Resources Association, 2000.

[GHM92] J.J. Godfrey, E.C. Holliman, and J. McDaniel. Switchboard : a telephone speech corpus for research and develpment. In *Proceedings of the IEEE COnference on Acoustics, Speech and Signal Processing*, volume 1, pages 517–520, 1992.

[GLW93] John S. Garofolo, Lori F. Lamel, and William. The darpa timit acoustic-phonetic continuous speech corpus cdro. (printed documentation for NIST Speech Disc 1-1.1) NTIS order number PB91-100354, 1993.

[Gri97] Grishman. Tipster architecture design document version 2.3. Technical report, DARPA, 1997.

[HC97] Lynette Hirshman and Nancy Chinchor. Muc-7 coreference task definition. In *Message Understanding Conference Proceedings*. Published online, 1997.

[JML01] Michel Jacobson, B Michailovsky, and J. B. Lowe. Linguistic documents synchronizing sound and text. In *Speech Communication*, volume 33, pages 79–96, 2001.

[KIM+00] David Mac Kelvie, Amy Isard, Andreas Mengel, M. Moller, M. Grosse, and Marion Klein. The mate workbench: an annotation tool for xml coded speech corpora. In *Speech Communication*, 2000.

[NIS98] NIST. A universal transcription format (utf) annotation specification for evaluation of spoken language technology corpora. www.nist.gov/speech/hub4_98/utf-1.0-v2.ps, 1998.

[SBGW98] F. Schiel, S. Burger, A. Geumann, and K. Weilhammer. The partitur format at bas. In *Proceedings of the First International Conference on Language Resources and Evaluation*, Granada - Spain, 1998.

[TBC00] Paul Taylor, Alan W. Black, and Richard Caley. Heterogeneous relation graphs as a formalism for representing linguistic information. In *Speech Communication*, 2000.

[Whi95] Brian Mac Whinney. *The CHILDES Project: Tools for Analysing Talk*, volume second edition. Lawrence Erlbraum, Mahwah, 1995.

Appendix A

Examples

A.1 Examples of annotations

This section provides two examples.

A.1.1 Annotation of a word

Here is an annotation of a word:

```
<c_word>
  <id>word4</id>
  <list-tokens>
    <id_token>7</id_token>
  </list-tokens>
  <lemma>
    <canonical_form>inhibit</canonical_form>
  </lemma>
  <semantics>
    <is_term>F</is_term>
    <is_named_entity>F</is_named_entity>
    <semantical_features>
      <gene_action>inhibition</gene_action>
      <sign_action>positive</sign_action>
    </semantical_features>
  </semantics>
  <morphology>
    <composition>in,hibit,s</composition> <!-- prefix,root,suffix -->
    <stem>inhibit</stem>
  </morphology>
  <morpho-syntactic>
    <category>verb</category>
```

```
        <morpho-syntactic_features>
          <tense>present</tense>
          <aspect>simple</aspect>
          <person>3rd</person>
          <number>singular</number>
          <voice>active</voice>
        </morpho-syntactic_features>
      </morpho-syntactic>
    </c_word>
```

The word with id `word4` is composed of one token (the `token7` is `inhibits`). Its lemma-part describes its canonical form, `inhibit`. Its semantic part states that it is not a term neither a named entity, and that it describes a gene action that is positive. Then, the morphology part show its composition (prefix, root, suffix) and its stem (a kind of suffix-cutting). Finally, the morpho-syntactic part gives the category (it is a `verb`), and some features of the verb.

A.1.2 Complete example

Here is a complete enough example of linguistically annotated text:

```
<?xml version="1.0"?>

<docann
  xmlns="http://www-lipn.univ-paris13.fr/~vauvert/alvis/xml"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www-lipn.univ-paris13.fr/~vauvert/alvis/xml docann.xsd">

  <doc>The gerE protein inhibits transcription in vitro of the sigK gene encoding sigmaK.
</doc>
  <ann>
    <!--++++++++++++++-->
    <!-- token layer -->
    <!--++++++++++++++-->

    <c_alpha>
      <id>token1</id>
      <from>0</from>
      <to>2</to>
      <cont>The</cont>
    </c_alpha>
    <c_sep>
      <id>token2</id>
      <from>3</from>
      <to>3</to>
      <cont> </cont>
```

```
  </c_sep>
  <c_alpha>
    <id>token3</id>
    <from>4</from>
    <to>7</to>
    <cont>gerE</cont>
  </c_alpha>
  <c_sep>
    <id>token4</id>
    <from>8</from>
    <to>8</to>
    <cont> </cont>
  </c_sep>
  <c_alpha>
    <id>token5</id>
    <from>9</from>
    <to>15</to>
    <cont>protein</cont>
  </c_alpha>
  <c_sep>
    <id>token6</id>
    <from>16</from>
    <to>16</to>
    <cont> </cont>
  </c_sep>
  <c_alpha>
    <id>token7</id>
    <from>17</from>
    <to>24</to>
    <cont>inhibits</cont>
  </c_alpha>
  <c_sep>
    <id>token8</id>
    <from>25</from>
    <to>25</to>
    <cont> </cont>
  </c_sep>
  <c_alpha>
    <id>token9</id>
    <from>26</from>
    <to>38</to>
    <cont>transcription</cont>
  </c_alpha>
  <c_sep>
    <id>token10</id>
    <from>39</from>
```

```
    <to>39</to>
    <cont> </cont>
  </c_sep>
  <c_alpha>
    <id>token11</id>
    <from>40</from>
    <to>41</to>
    <cont>in</cont>
  </c_alpha>
  <c_sep>
    <id>token12</id>
    <from>42</from>
    <to>42</to>
    <cont> </cont>
  </c_sep>
  <c_alpha>
    <id>token13</id>
    <from>43</from>
    <to>47</to>
    <cont>vitro</cont>
  </c_alpha>
  <c_sep>
    <id>token14</id>
    <from>48</from>
    <to>48</to>
    <cont> </cont>
  </c_sep>
  <c_alpha>
    <id>token15</id>
    <from>49</from>
    <to>50</to>
    <cont>of</cont>
  </c_alpha>
  <c_sep>
    <id>token16</id>
    <from>51</from>
    <to>51</to>
    <cont> </cont>
  </c_sep>
  <c_alpha>
    <id>token17</id>
    <from>52</from>
    <to>54</to>
    <cont>the</cont>
  </c_alpha>
  <c_sep>
```

```
    <id>token18</id>
    <from>55</from>
    <to>55</to>
    <cont> </cont>
  </c_sep>
  <c_alpha>
    <id>token19</id>
    <from>56</from>
    <to>59</to>
    <cont>sigK</cont>
  </c_alpha>
  <c_sep>
    <id>token20</id>
    <from>60</from>
    <to>60</to>
    <cont> </cont>
  </c_sep>
  <c_alpha>
    <id>token21</id>
    <from>61</from>
    <to>64</to>
    <cont>gene</cont>
  </c_alpha>
  <c_sep>
    <id>token22</id>
    <from>65</from>
    <to>65</to>
    <cont> </cont>
  </c_sep>
  <c_alpha>
    <id>token23</id>
    <from>66</from>
    <to>73</to>
    <cont>encoding</cont>
  </c_alpha>
  <c_sep>
    <id>token24</id>
    <from>74</from>
    <to>74</to>
    <cont> </cont>
  </c_sep>
  <c_alpha>
    <id>token25</id>
    <from>75</from>
    <to>80</to>
    <cont>sigmaK</cont>
```

```
    </c_alpha>
    <c_symb>
      <id>token26</id>
      <from>81</from>
      <to>81</to>
      <cont>.</cont>
    </c_symb>
    <c_sep>
      <id>token27</id>
      <from>82</from>
      <to>82</to>
      <cont>\n</cont>
    </c_sep>

    <!--+++++++++++++-->
    <!-- word layer -->
    <!--+++++++++++++-->

    <c_word>
      <id>word1</id>
      <list-refid-token>
<refid-token>token1</refid-token>
      </list-refid-token>
      <!--variante></variante-->
      <lemma>
<canonical_form>the</canonical_form>
<source></source>
      </lemma>
      <semantics>
<is_term>F</is_term>
<is_named_entity>F</is_named_entity>
<semantical_features>
  <feature></feature>
</semantical_features>
      </semantics>
      <morphology>
<composition>the</composition> <!-- prefix*,(infix|root)*,root,(infix|root)*,suffix* -->
        <stem></stem>
      </morphology>
      <morpho-syntactic>
<category>definite article</category> <!-- kind of determiner -->
<morpho-syntactic_features>
  <number>singular</number>
</morpho-syntactic_features>
      </morpho-syntactic>
    </c_word>
```

```
    <c_word>
      <id>word2</id>
      <list-refid-token>
<refid-token>token3</refid-token>
      </list-refid-token>
      <!--variante></variante-->
      <lemma>
<canonical_form>gerE</canonical_form>
<source></source>
      </lemma>
      <semantics>
<is_term>F</is_term>
<is_named_entity>T</is_named_entity>
<semantical_features>
  <object_category>protein name</object_category> <!-- protein name < biologic name -->
</semantical_features>
      </semantics>
      <morphology>
      </morphology>
      <morpho-syntactic>
<category>noun</category>
<morpho-syntactic_features>
  <number>singular</number>
</morpho-syntactic_features>
      </morpho-syntactic>
    </c_word>

    <c_word>
      <id>word3</id>
      <list-refid-token>
<refid-token>token5</refid-token>
      </list-refid-token>
      <!--variante></variante-->
      <lemma>
<canonical_form>protein</canonical_form> <!-- protein < biologic -->
<source></source>
      </lemma>
      <semantics>
<is_term>F</is_term>
<is_named_entity>F</is_named_entity>
<semantical_features>
  <object_category>protein</object_category> <!-- protein < biologic -->
</semantical_features>
      </semantics>
      <morphology>
```

```
<composition>protein</composition> <!-- prefix*,(infix|root)*,root,(infix|root)*,suffix* -->
        <stem>protein</stem>
      </morphology>
      <morpho-syntactic>
<category>noun</category>
<morpho-syntactic_features>
  <number>singular</number>
</morpho-syntactic_features>
      </morpho-syntactic>
    </c_word>

    <c_word>
      <id>word4</id>
      <list-refid-token>
<refid-token>token7</refid-token>
      </list-refid-token>
      <!--variante></variante-->
      <lemma>
<canonical_form>inhibit</canonical_form>
<source></source>
      </lemma>
      <semantics>
<is_term>F</is_term>
<is_named_entity>F</is_named_entity>
<semantical_features>
  <gene_action>interaction</gene_action>
  <sign_action>negative</sign_action>
</semantical_features>
      </semantics>
      <morphology>
<composition>in,hibit,s</composition> <!-- prefix*,(infix|root)*,root,(infix|root)*,suffix* -->
        <stem>inhibit</stem>
      </morphology>
      <morpho-syntactic>
<category>verb</category>
<morpho-syntactic_features>
  <tense>present</tense>
  <aspect>simple</aspect>
  <person>3rd</person>
  <number>singular</number>
  <voice>active</voice>
</morpho-syntactic_features>
      </morpho-syntactic>
    </c_word>

    <c_word>
```

```
      <id>word5</id>
      <list-refid-token>
<refid-token>token9</refid-token>
      </list-refid-token>
      <!--variante></variante-->
      <lemma>
<canonical_form>transcription</canonical_form>
<source></source>
      </lemma>
      <semantics>
<is_term>T</is_term>
<is_named_entity>F</is_named_entity>
<semantical_features>
  <feature></feature>
</semantical_features>
      </semantics>
      <morphology>
<composition>trans,script,ion</composition> <!-- prefix*,(infix|root)*,root,(infix|root)*,suffix* -->
        <stem>transcript</stem>
      </morphology>
      <morpho-syntactic>
<category>noun</category>
<morpho-syntactic_features>
  <number>singular</number>
</morpho-syntactic_features>
      </morpho-syntactic>
    </c_word>

    <c_word>
      <id>word6</id>
      <list-refid-token>
<refid-token>token11</refid-token>
<refid-token>token12</refid-token>
<refid-token>token13</refid-token>
      </list-refid-token>
      <!--variante></variante-->
      <lemma>
<canonical_form>in vitro</canonical_form>
<source></source>
      </lemma>
      <semantics>
<is_term>F</is_term>
<is_named_entity>T</is_named_entity>
<semantical_features>
  <feature></feature>
</semantical_features>
```

```
      </semantics>
      <morphology>
<composition></composition> <!-- prefix*,(infix|root)*,root,(infix|root)*,suffix* -->
        <stem></stem>
      </morphology>
      <morpho-syntactic>
<category>adverb</category>
<morpho-syntactic_features>
  <kind>manner</kind>
</morpho-syntactic_features>
      </morpho-syntactic>
    </c_word>

    <c_word>
      <id>word7</id>
      <list-refid-token>
<refid-token>token15</refid-token>
      </list-refid-token>
      <!--variante></variante-->
      <lemma>
<canonical_form>of</canonical_form>
<source></source>
      </lemma>
      <semantics>
<is_term>F</is_term>
<is_named_entity>F</is_named_entity>
<semantical_features>
  <feature></feature>
</semantical_features>
      </semantics>
      <morphology>
<composition></composition> <!-- prefix*,(infix|root)*,root,(infix|root)*,suffix* -->
        <stem></stem>
      </morphology>
      <morpho-syntactic>
<category>preposition</category>
<morpho-syntactic_features>
  <feature></feature>
</morpho-syntactic_features>
      </morpho-syntactic>
    </c_word>

    <c_word>
      <id>word8</id>
      <list-refid-token>
<refid-token>token17</refid-token>
```

```
      </list-refid-token>
      <!--variante></variante-->
      <lemma>
<canonical_form>the</canonical_form>
<source></source>
      </lemma>
      <semantics>
<is_term>F</is_term>
<is_named_entity>F</is_named_entity>
<semantical_features>
  <feature></feature>
</semantical_features>
      </semantics>
      <morphology>
<composition></composition> <!-- prefix*,(infix|root)*,root,(infix|root)*,suffix* -->
        <stem></stem>
      </morphology>
      <morpho-syntactic>
<category>definite article</category> <!-- kind of determiner -->
<morpho-syntactic_features>
  <feature></feature>
</morpho-syntactic_features>
      </morpho-syntactic>
    </c_word>

    <c_word>
      <id>word9</id>
      <list-refid-token>
<refid-token>token19</refid-token>
      </list-refid-token>
      <!--variante></variante-->
      <lemma>
<canonical_form>sigK</canonical_form>
<source></source>
      </lemma>
      <semantics>
<is_term>F</is_term>
<is_named_entity>T</is_named_entity>
<semantical_features>
  <object_category>gene name</object_category> <!-- gene name < biologic name -->
</semantical_features>
      </semantics>
      <morphology>
<composition></composition> <!-- prefix*,(infix|root)*,root,(infix|root)*,suffix* -->
        <stem></stem>
      </morphology>
```

```
      <morpho-syntactic>
<category>noun</category>
<morpho-syntactic_features>
  <number>singular</number>
</morpho-syntactic_features>
      </morpho-syntactic>
    </c_word>

    <c_word>
      <id>word10</id>
      <list-refid-token>
<refid-token>token21</refid-token>
      </list-refid-token>
      <!--variante></variante-->
      <lemma>
<canonical_form>gene</canonical_form>
<source></source>
      </lemma>
      <semantics>
<is_term>F</is_term>
<is_named_entity>F</is_named_entity>
<semantical_features>
  <object_category>gene</object_category>
</semantical_features>
      </semantics>
      <morphology>
<composition>gene</composition> <!-- prefix*,(infix|root)*,root,(infix|root)*,suffix* -->
        <stem></stem>
      </morphology>
      <morpho-syntactic>
<category>noun</category>
<morpho-syntactic_features>
  <number>singular</number>
</morpho-syntactic_features>
      </morpho-syntactic>
    </c_word>

    <c_word>
      <id>word11</id>
      <list-refid-token>
<refid-token>token23</refid-token>
      </list-refid-token>
      <!--variante></variante-->
      <lemma>
<canonical_form>encode</canonical_form>
<source></source>
```

```
      </lemma>
      <semantics>
<is_term>F</is_term>
<is_named_entity>F</is_named_entity>
<semantical_features>
  <feature></feature>
</semantical_features>
      </semantics>
      <morphology>
<composition>en,cod,ing</composition> <!-- prefix*,(infix|root)*,root,(infix|root)*,suffix* -->
        <stem>encod</stem>
      </morphology>
      <morpho-syntactic>
<category>verb</category>
<morpho-syntactic_features>
  <form>present participle</form> <!-- form>-ing</form !--> <!-- form>participle -ing</form !-->
</morpho-syntactic_features>
      </morpho-syntactic>
    </c_word>

    <c_word>
      <id>word12</id>
      <list-refid-token>
<refid-token>token25</refid-token>
      </list-refid-token>
      <!--variante></variante-->
      <lemma>
<canonical_form>sigmaK</canonical_form>
<source></source>
      </lemma>
      <semantics>
<is_term>T</is_term>
<is_named_entity>T</is_named_entity>
<semantical_features>
  <object_category>protein name</object_category> <!-- protein name < biologic name -->
</semantical_features>
      </semantics>
      <morphology>
<composition></composition> <!-- prefix*,(infix|root)*,root,(infix|root)*,suffix* -->
        <stem></stem>
      </morphology>
      <morpho-syntactic>
<category>noun</category>
<morpho-syntactic_features>
  <number>singular</number>
</morpho-syntactic_features>
```

```
      </morpho-syntactic>
    </c_word>

    <c_word>
      <id>word13</id>
      <list-refid-token>
<refid-token>token27</refid-token>
      </list-refid-token>
      <!--variante></variante-->
      <lemma>
<canonical_form>at</canonical_form>
<source></source>
      </lemma>
      <semantics>
<is_term>F</is_term>
<is_named_entity></is_named_entity>
<semantical_features>
  <feature></feature>
</semantical_features>
      </semantics>
      <morphology>
<composition></composition> <!-- prefix*,(infix|root)*,root,(infix|root)*,suffix* -->
        <stem></stem>
      </morphology>
      <morpho-syntactic>
<category>preposition</category>
<morpho-syntactic_features>
  <type>simple</type>
</morpho-syntactic_features>
      </morpho-syntactic>
    </c_word>

    <c_word>
      <id>word14</id>
      <list-refid-token>
<refid-token>token29</refid-token>
<refid-token>token30</refid-token>
      </list-refid-token>
      <!--variante></variante-->
      <lemma>
<canonical_form>0.3</canonical_form>
<source></source>
      </lemma>
      <semantics>
<is_term>F</is_term>
<is_named_entity>F</is_named_entity>
```

```
<semantical_features>
  <feature></feature>
</semantical_features>
      </semantics>
      <morphology>
<composition></composition> <!-- prefix*,(infix|root)*,root,(infix|root)*,suffix* -->
        <stem></stem>
      </morphology>
      <morpho-syntactic>
<category>cardinal numeral</category> <!-- sub-category of numeral, that is a sub-category of noun !-->
<morpho-syntactic_features>
  <feature></feature>
</morpho-syntactic_features>
      </morpho-syntactic>
    </c_word>

    <!--+++++++++++++++-->
    <!-- phrase layer -->
    <!--+++++++++++++++-->

    <c_phrase>
      <id>phrase1</id>
      <semantics>
<is_term>T</is_term>
<term_lemma>
  <canonical_form>gerE protein</canonical_form>
  <source></source>
</term_lemma>
<semantical_features>
  <object_category>named protein</object_category> <!-- named protein < named biologic -->
</semantical_features>
      </semantics>
      <syntax>
<cat>noun phrase</cat>
<!-- composition -->
<list-refid-word_or_phrase>
  <refid-word_or_phrase>word2</refid-word_or_phrase>
  <refid-word_or_phrase>word3</refid-word_or_phrase>
</list-refid-word_or_phrase>
<!-- /composition -->
<morpho-syntactic_features>
  <number>singular</number>
</morpho-syntactic_features>
      </syntax>
    </c_phrase>
```

```
    <c_phrase>
      <id>phrase2</id>
      <semantics>
<is_term>T</is_term>
<term_lemma>
  <canonical_form>sigK gene</canonical_form>
  <source></source>
</term_lemma>
<semantical_features>
  <object_category>named gene</object_category>
</semantical_features>
      </semantics>
      <syntax>
<cat>noun phrase</cat>
<!-- composition -->
<list-refid-word_or_phrase>
  <refid-word_or_phrase>word9</refid-word_or_phrase>
  <refid-word_or_phrase>word10</refid-word_or_phrase>
</list-refid-word_or_phrase>
<!-- /composition -->
<morpho-syntactic_features>
  <number>singular</number>
</morpho-syntactic_features>
      </syntax>
    </c_phrase>

    <c_phrase>
      <id>phrase3</id>
      <semantics>
<is_term>T</is_term>
<term_lemma>
  <canonical_form>sigmaK</canonical_form>
  <source></source>
</term_lemma>
<semantical_features>
  <object_category>named protein</object_category>
</semantical_features>
      </semantics>
      <syntax>
<cat>noun phrase</cat>
<!-- composition -->
<list-refid-word_or_phrase>
  <refid-word_or_phrase>word12</refid-word_or_phrase>
</list-refid-word_or_phrase>
<!-- /composition -->
<morpho-syntactic_features>
```

```
  <number>singular</number>
</morpho-syntactic_features>
      </syntax>
    </c_phrase>

    <c_phrase>
      <id>phrase4</id>
      <semantics>
<is_term></is_term>
<term_lemma>
  <canonical_form></canonical_form>
  <source></source>
</term_lemma>
<semantical_features>
  <feature></feature>
</semantical_features>
      </semantics>
      <syntax>
<cat></cat>
<!-- composition -->
<list-refid-word_or_phrase>
  <refid-word_or_phrase>word13</refid-word_or_phrase>
  <refid-word_or_phrase>word14</refid-word_or_phrase>
</list-refid-word_or_phrase>
<!-- /composition -->
<morpho-syntactic_features>
  <feature></feature>
</morpho-syntactic_features>
      </syntax>
    </c_phrase>

    <c_phrase>
      <id>phrase5</id>
      <semantics>
<is_term></is_term>
<term_lemma>
  <canonical_form></canonical_form>
  <source></source>
</term_lemma>
<semantical_features>
  <feature></feature>
</semantical_features>
      </semantics>
      <syntax>
<cat>noun phrase</cat>
<!-- composition -->
```

```
<list-refid-word_or_phrase>
  <refid-word_or_phrase>word1</refid-word_or_phrase>
  <refid-word_or_phrase>phrase1</refid-word_or_phrase>
</list-refid-word_or_phrase>
<!-- /composition -->
<morpho-syntactic_features>
  <number>singular</number>
</morpho-syntactic_features>
      </syntax>
    </c_phrase>

    <c_phrase>
      <id>phrase6</id>
      <semantics>
<is_term></is_term>
<term_lemma>
  <canonical_form></canonical_form>
  <source></source>
</term_lemma>
<semantical_features>
  <feature></feature>
</semantical_features>
      </semantics>
      <syntax>
<cat>verb phrase</cat>
<!-- composition -->
<list-refid-word_or_phrase>
  <refid-word_or_phrase>word11</refid-word_or_phrase>
  <refid-word_or_phrase>phrase3</refid-word_or_phrase>
  <refid-word_or_phrase>phrase4</refid-word_or_phrase>
</list-refid-word_or_phrase>
<!-- /composition -->
<morpho-syntactic_features>
  <tense>present</tense>
  <aspect>simple</aspect>
  <person>3rd</person>
  <number>singular</number>
  <voice>active</voice>
</morpho-syntactic_features>
      </syntax>
    </c_phrase>

    <c_phrase>
      <id>phrase7</id>
      <semantics>
<is_term></is_term>
```

```
<term_lemma>
  <canonical_form></canonical_form>
  <source></source>
</term_lemma>
<semantical_features>
  <feature></feature>
</semantical_features>
      </semantics>
      <syntax>
<cat>noun phrase</cat>
<!-- composition -->
<list-refid-word_or_phrase>
  <refid-word_or_phrase>phrase2</refid-word_or_phrase>
  <refid-word_or_phrase>phrase6</refid-word_or_phrase>
</list-refid-word_or_phrase>
<!-- /composition -->
<morpho-syntactic_features>
  <number>singular</number>
</morpho-syntactic_features>
      </syntax>
    </c_phrase>

    <c_phrase>
      <id>phrase8</id>
      <semantics>
<is_term></is_term>
<term_lemma>
  <canonical_form></canonical_form>
  <source></source>
</term_lemma>
<semantical_features>
  <feature></feature>
</semantical_features>
      </semantics>
      <syntax>
<cat>noun phrase</cat>
<!-- composition -->
<list-refid-word_or_phrase>
  <refid-word_or_phrase>word8</refid-word_or_phrase>
  <refid-word_or_phrase>phrase7</refid-word_or_phrase>
</list-refid-word_or_phrase>
<!-- /composition -->
<morpho-syntactic_features>
  <number>singular</number>
</morpho-syntactic_features>
      </syntax>
```

```
    </c_phrase>

    <c_phrase>
      <id>phrase9</id>
      <semantics>
<is_term></is_term>
<term_lemma>
  <canonical_form></canonical_form>
  <source></source>
</term_lemma>
<semantical_features>
  <feature></feature>
</semantical_features>
      </semantics>
      <syntax>
<cat>prepositional phrase</cat>
<!-- composition -->
<list-refid-word_or_phrase>
  <refid-word_or_phrase>word7</refid-word_or_phrase>
  <refid-word_or_phrase>phrase8</refid-word_or_phrase>
</list-refid-word_or_phrase>
<!-- /composition -->
<morpho-syntactic_features>
  <feature></feature>
</morpho-syntactic_features>
      </syntax>
    </c_phrase>

    <c_phrase>
      <id>phrase10</id>
      <semantics>
<is_term></is_term>
<term_lemma>
  <canonical_form></canonical_form>
  <source></source>
</term_lemma>
<semantical_features>
  <feature></feature>
</semantical_features>
      </semantics>
      <syntax>
<cat>noun phrase</cat>
<!-- composition -->
<list-refid-word_or_phrase>
  <refid-word_or_phrase>word5</refid-word_or_phrase>
  <refid-word_or_phrase>word6</refid-word_or_phrase>
```

```
  <refid-word_or_phrase>phrase9</refid-word_or_phrase>
</list-refid-word_or_phrase>
<!-- /composition -->
<morpho-syntactic_features>
  <number>singular</number>
</morpho-syntactic_features>
      </syntax>
    </c_phrase>

    <c_phrase>
      <id>phrase11</id>
      <semantics>
<is_term></is_term>
<term_lemma>
  <canonical_form></canonical_form>
  <source></source>
</term_lemma>
<semantical_features>
  <feature></feature>
</semantical_features>
      </semantics>
      <syntax>
<cat>verb phrase</cat>
<!-- composition -->
<list-refid-word_or_phrase>
  <refid-word_or_phrase>word4</refid-word_or_phrase>
  <refid-word_or_phrase>phrase10</refid-word_or_phrase>
</list-refid-word_or_phrase>
<!-- /composition -->
<morpho-syntactic_features>
  <feature></feature>
</morpho-syntactic_features>
      </syntax>
    </c_phrase>

    <!--+++++++++++++++++-->
    <!-- sentence layer -->
    <!--+++++++++++++++++-->

    <c_sentence>
      <id>sentence1</id>
      <semantics>
<semantical_features>
  <feature></feature>
</semantical_features>
      </semantics>
```

```
      <syntax>
<cat></cat>
<!-- composition -->
<list-refid-word_or_phrase>
  <refid-word_or_phrase>phrase5</refid-word_or_phrase>
  <refid-word_or_phrase>phrase10</refid-word_or_phrase>
</list-refid-word_or_phrase>
<!-- /composition -->
<morpho-syntactic_features>
  <feature></feature>
</morpho-syntactic_features>
      </syntax>
    </c_sentence>

    <!--++++++++++++++++++++++++++-->
    <!-- function relation layer -->
    <!--++++++++++++++++++++++++++-->

    <r_function>
      <id>function1</id>
      <type>object</type>
      <refid-agent>phrase10</refid-agent>
      <refid-object>word4</refid-object>
    </r_function>

    <r_function>
      <id>function2</id>
      <type>subject</type>
      <refid-agent>phrase5</refid-agent>
      <refid-object>phrase10</refid-object>
    </r_function>

    <!--++++++++++++++++++++++++++-->
    <!-- biologic relation layer -->
    <!--++++++++++++++++++++++++++-->

    <r_biologic>
      <id>biologic1</id>
      <type>interaction</type>
      <sign>negative</sign>
      <refid-agent>phrase5</refid-agent> <!-- source ? -->
      <refid-object>phrase10</refid-object>
      <refid-verb>word4</refid-verb>
    </r_biologic>

  </ann>
```

```
</docann>
```

A.1.3 Concurrent annotation

Here is an example of concurrently annotated text that contains the two part-of-speech tagging versions, without explicite representation of alternative. This example use another version of the format: there is no refid in concept, only in relation, what leads to increase the set of annotations.

```
<?xml version="1.0"?>

<docann
  xmlns="http://www-lipn.univ-paris13.fr/~vauvert/alvis/schema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www-lipn.univ-paris13.fr/~vauvert/alvis/schema docann.xsd">

  <doc>La belle porte le voile.</doc>
  <ann>

    <!-- segmentation : token -->
    <c_alpha>
      <id>token1</id>
      <from>0</from>
      <to>1</to>
      <cont>La</cont>
    </c_alpha>
    <c_sep>
      <id>token2</id>
      <from>2</from>
      <to>2</to>
      <cont> </cont>
    </c_sep>
    <c_alpha>
      <id>token3</id>
      <from>3</from>
      <to>7</to>
      <cont>belle</cont>
    </c_alpha>
    <c_sep>
      <id>token4</id>
      <from>8</from>
      <to>8</to>
      <cont> </cont>
    </c_sep>
    <c_alpha>
      <id>token5</id>
```

```
      <from>9</from>
      <to>13</to>
      <cont>porte</cont>
    </c_alpha>
    <c_sep>
      <id>token6</id>
      <from>14</from>
      <to>14</to>
      <cont> </cont>
    </c_sep>
    <c_alpha>
      <id>token7</id>
      <from>15</from>
      <to>16</to>
      <cont>le</cont>
    </c_alpha>
    <c_sep>
      <id>token8</id>
      <from>17</from>
      <to>17</to>
      <cont> </cont>
    </c_sep>
    <c_alpha>
      <id>token9</id>
      <from>18</from>
      <to>22</to>
      <cont>voile</cont>
    </c_alpha>
    <c_symb>
      <id>token10</id>
      <from>23</from>
      <to>23</to>
      <cont>.</cont>
    </c_symb>

    <!-- segmentation : tokenization / words -->
    <c_word>
      <id>word1</id>
      <token>token1</token>
      <cont>La</cont>
    </c_word>
    <c_word>
      <id>word2</id>
      <token>token3</token>
      <cont>belle</cont>
    </c_word>
```

```
<c_word>
  <id>word3</id>
  <token>token5</token>
  <cont>porte</cont>
</c_word>
<c_word>
  <id>word4</id>
  <token>token7</token>
  <cont>le</cont>
</c_word>
<c_word>
  <id>word5</id>
  <token>token9</token>
  <cont>voile</cont>
</c_word>

<!-- segmentation : punctuation -->
<c_punct>
  <id>punct1</id>
  <token>token10</token>
  <cont>.</cont>
</c_punct>


<!-- segmentation : sentence -->
<c_sent>
  <id>sent1</id>
  <fromtk>token0</fromtk>
  <totk>token10</totk>
</c_sent>
<c_seqword>
  <id>seqword1</id>
  <seq>word1 word2 word3 word4 word5 word6</seq>
</c_seqword>
<r_seqword>
  <id>rseqword1</id>
  <sent>sent1</sent>
  <seqword>seqword1</seqword>
</r_seqword>

<!-- tagging : categories -->

<!-- tagging : la -->
<c_cat>
  <id>cat1</id>
  <val>determiner</val>
```

```
</c_cat>
<c_cat>
  <id>cat2</id>
  <val>pronoun</val>
</c_cat>
<r_cat>
  <id>rcat1</id>
  <cat>cat1</cat>
  <word>word1</word>
</r_cat>
<r_cat>
  <id>rcat2</id>
  <cat>cat2</cat>
  <word>word1</word>
</r_cat>

<!-- tagging : belle -->
<c_cat>
  <id>cat3</id>
  <val>adjective</val>
</c_cat>
<c_cat>
  <id>cat4</id>
  <val>noun</val>
</c_cat>
<r_cat>
  <id>rcat3</id>
  <cat>cat3</cat>
  <word>word2</word>
</r_cat>
<r_cat>
  <id>rcat4</id>
  <cat>cat4</cat>
  <word>word2</word>
</r_cat>

<!-- tagging : porte -->
<c_cat>
  <id>cat5</id>
  <val>noun</val>
</c_cat>
<c_cat>
  <id>cat6</id>
  <val>verb</val>
</c_cat>
<r_cat>
```

```
    <id>rcat5</id>
    <cat>cat5</cat>
    <word>word3</word>
  </r_cat>
  <r_cat>
    <id>rcat6</id>
    <cat>cat6</cat>
    <word>word3</word>
  </r_cat>

  <!-- tagging : le -->
  <c_cat>
    <id>cat7</id>
    <val>determiner</val>
  </c_cat>
  <c_cat>
    <id>cat8</id>
    <val>pronoun</val>
  </c_cat>
  <r_cat>
    <id>rcat7</id>
    <cat>cat7</cat>
    <word>word4</word>
  </r_cat>
  <r_cat>
    <id>rcat8</id>
    <cat>cat8</cat>
    <word>word4</word>
  </r_cat>

  <!-- tagging : voile -->
  <c_cat>
    <id>cat9</id>
    <val>noun</val>
  </c_cat>
  <c_cat>
    <id>cat10</id>
    <val>verb</val>
  </c_cat>
  <r_cat>
    <id>rcat9</id>
    <cat>cat9</cat>
    <word>word5</word>
  </r_cat>
  <r_cat>
    <id>rcat10</id>
```

```
    <cat>cat10</cat>
    <word>word5</word>
  </r_cat>

  <!-- tagging : point . -->
  <c_cat_punct>
    <id>catpunct1</id>
    <cat>end</cat>
  </c_cat_punct>
  <r_cat_punct>
    <id>rcatpunct1</id>
    <cat_punct>catpunct1</cat_punct>
    <punct>punct1</punct>
  </r_cat_punct>

  <!-- tagging : sequences of tags -->
  <c_seqcat>
    <id>seqcat1</id>
    <seq>cat1 cat3 cat5 cat7 cat9 punc1</seq>
  </c_seqcat>
  <r_seqcat>
    <id>rseqcat1</id>
    <seqword>seqword1</seqword>
    <seqcat>seqcat1</seqcat>
  </r_seqcat>

  <c_seqcat>
    <id>seqcat2</id>
    <seq>cat2 cat4 cat6 cat8 cat10 punc1</seq>
  </c_seqcat>
  <r_seqcat>
    <id>rseqcat2</id>
    <seqword>seqword1</seqword>
    <seqcat>seqcat2</seqcat>
  </r_seqcat>

  <!-- phrase analysis -->
  <!-- [la belle porte] [[le] [voile]]-->

  <c_seqword>
    <id>seqword2</id>
    <seq>word2</seq>
  </c_seqword>

  <c_adjective_phrase>
    <id>adjective_phrase1</id>
```

```
    <assume>seqcat1</assume>
    <seq>seqword2</seq>
    <head>word2</head>
  </c_adjective_phrase>

  <c_seqword>
    <id>seqword3</id>
    <seq>word1 word2 word3</seq>
  </c_seqword>

  <c_noun_phrase>
    <id>noun_phrase1</id>
    <assume>seqcat1</assume>
    <seq>seqword2</seq>
    <head>word3</head>
  </c_noun_phrase>

  <c_seqword>
    <id>seqword4</id>
    <seq>word4</seq>
  </c_seqword>

  <c_noun_phrase>
    <id>noun_phrase2</id>
    <assume>seqcat1</assume>
    <seq>seqword4</seq>
    <head>word4</head>
  </c_noun_phrase>

  <c_seqword>
    <id>seqword5</id>
    <seq>word4 word5</seq>
  </c_seqword>

  <c_verb_phrase>
    <id>verb_phrase1</id>
    <assume>seqcat1</assume>
    <seq>seqword5</seq>
    <head>word5</head>
  </c_verb_phrase>

  <!-- [la belle] [porte [le voile]]-->

  <c_seqword>
    <id>seqword6</id>
    <seq>word1 word2</seq>
```

```
    </c_seqword>

    <c_noun_phrase>
      <id>noun_phrase3</id>
      <assume>seqcat2</assume>
      <seq>seqword6</seq>
      <head>word2</head>
    </c_noun_phrase>

    <c_noun_phrase>
      <id>noun_phrase2</id>
      <assume>seqcat2</assume>
      <seq>seqword5</seq>
      <head>word5</head>
    </c_noun_phrase>

    <c_seqword>
      <id>seqword6</id>
      <seq>word3 word4 word5</seq>
    </c_seqword>

    <c_verb_phrase>
      <id>verb_phrase1</id>
      <assume>seqcat2</assume>
      <seq>seqword6</seq>
      <head>word3</head>
    </c_verb_phrase>

    <!-- clause -->
    <c_clause>
      <id>clause1</id>
      <seq>seqword1</seq>
    </c_clause>

  </ann>
</docann>
```

A.2 Schema of documents

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://www-lipn.univ-paris13.fr/~vauvert/alvis/xml"
  xmlns="http://www-lipn.univ-paris13.fr/~vauvert/alvis/xml"
  elementFormDefault="qualified">
```

```
  <xs:element name="docann">
    <xs:complexType>
      <xs:sequence>
<xs:element name="doc" type="xs:string"/>
<xs:element name="ann">
  <xs:complexType>
    <xs:sequence maxOccurs="unbounded">

      <xs:element ref="c_token" minOccurs="0"/>

      <xs:element ref="c_word" minOccurs="0"/>

      <xs:element ref="c_punct" minOccurs="0"/>

      <xs:element ref="c_sent" minOccurs="0"/>

      <xs:element name="c_seqword" minOccurs="0">
<xs:complexType>
  <xs:sequence>
    <xs:element name="id" type="id_c_seqword"/>
    <!-- xs:element name="seq" type="t_seqword"/ -->
    <xs:element name="seq" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
      </xs:element>

      <xs:element name="r_seqword" minOccurs="0">
<xs:complexType>
  <xs:sequence>
    <xs:element name="id" type="id_r_seqword"/>
    <xs:element name="sent" type="idref_c_sent"/>
    <xs:element name="seqword" type="idref_c_seqword"/>
  </xs:sequence>
</xs:complexType>
      </xs:element>

      <xs:element name="c_cat" minOccurs="0">
<xs:complexType>
  <xs:sequence>
    <xs:element name="id" type="idccat"/>
    <xs:element name="val" type="xs:string"/>
    <!-- xs:element name="val" type=enumeration/ -->
  </xs:sequence>
</xs:complexType>
      </xs:element>
```

```
      <xs:element name="r_cat" minOccurs="0">
<xs:complexType>
  <xs:sequence>
    <xs:element name="id" type="id_r_cat"/>
    <xs:element name="cat" type="idref_c_cat"/>
    <xs:element name="word" type="idref_c_word"/>
  </xs:sequence>
</xs:complexType>
      </xs:element>

      <xs:element name="c_cat_punct" minOccurs="0">
<xs:complexType>
  <xs:sequence>
    <xs:element name="id" type="id_c_cat_punct"/>
    <xs:element name="cat" type="xs:string"/>
    <!-- xs:element name="val" type=enumeration/ -->
  </xs:sequence>
</xs:complexType>
      </xs:element>

      <xs:element name="r_cat_punct" minOccurs="0">
<xs:complexType>
  <xs:sequence>
    <xs:element name="id" type="id_r_cat_punct"/>
    <xs:element name="cat_punct" type="idref_c_cat_punct"/>
    <xs:element name="punct" type="idref_c_punct"/>
  </xs:sequence>
</xs:complexType>
      </xs:element>

      <xs:element name="c_seqcat" minOccurs="0">
<xs:complexType>
  <xs:sequence>
    <xs:element name="id" type="id_c_seqcat"/>
    <xs:element name="seq" type="xs:string"/>
    <!-- xs:element name="val" type=liste/ -->
  </xs:sequence>
</xs:complexType>
      </xs:element>

      <xs:element name="r_seqcat" minOccurs="0">
<xs:complexType>
  <xs:sequence>
    <xs:element name="id" type="id_r_seqcat"/>
    <xs:element name="seqword" type="idref_c_seqword"/>
```

```
    <xs:element name="seqcat" type="idref_c_seqcat"/>
  </xs:sequence>
</xs:complexType>
      </xs:element>

              <xs:element name="c_adjective_phrase" type="type_c_phrase" minOccurs="0"/>
              <xs:element name="c_noun_phrase" type="type_c_phrase" minOccurs="0"/>
              <xs:element name="c_verb_phrase" type="type_c_phrase" minOccurs="0"/>
              <xs:element name="c_adverb_phrase" type="type_c_phrase" minOccurs="0"/>
              <xs:element name="c_prepositional_phrase" type="type_c_phrase" minOccurs="0"/>

              <xs:element name="c_clause" minOccurs="0">
<xs:complexType>
                  <xs:sequence>
                    <xs:element name="id" type="id_c_clause"/>
    <xs:element name="seq" type="idref_c_seqword"/>
  </xs:sequence>
</xs:complexType>
              </xs:element>

              <xs:element name="r_ordinate_clause" minOccurs="0">
<xs:complexType>
                  <xs:sequence>
                    <xs:element name="id" type="id_r_ordinate_clause"/>
    <xs:element name="subordinate" type="id_c_clause"/>
    <xs:element name="superordinate" type="id_c_clause"/>
  </xs:sequence>
</xs:complexType>
              </xs:element>

    </xs:sequence>
  </xs:complexType>
</xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:complexType name="interval">
    <xs:sequence>
      <xs:element name="id" type="id_c_interval"/>
      <xs:element name="from" type="offset"/>
      <xs:element name="to" type="offset"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="t_c_token">
```

```
    <xs:sequence>
      <xs:element name="id" type="id_c_token"/>
      <xs:element name="from" type="offset"/>
      <xs:element name="to" type="offset"/>
      <xs:element name="cont" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>

  <xs:element name="c_token" type="t_c_token" abstract="true"/>

  <xs:element name="c_symb" substitutionGroup="c_token">
  </xs:element>

  <xs:element name="c_alpha" substitutionGroup="c_token">
  </xs:element>

  <xs:element name="c_num" substitutionGroup="c_token">
  </xs:element>

  <xs:element name="c_sep" substitutionGroup="c_token">
  </xs:element>

  <xs:element name="c_word">
    <xs:complexType>
      <xs:sequence>
<xs:element name="id" type="id_c_word"/>
<xs:element name="token" type="idref_c_token"/>
<xs:element name="cont" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name="c_sent" >
    <xs:complexType>
      <xs:sequence>
<xs:element name="id" type="id_c_sent"/>
<xs:element name="fromtk" type="idref_c_token"/>
<xs:element name="totk" type="idref_c_token"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name="c_punct" minOccurs="0">
    <xs:complexType>
      <xs:sequence>
<xs:element name="id" type="id_c_punct"/>
```

```
<xs:element name="token" type="idref_c_token"/>
<xs:element name="cont" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:simpleType name="offset">
    <xs:restriction base="xs:integer">
      <xs:minInclusive value="0"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="id_c_interval">
    <xs:restriction base="xs:ID">
      <xs:pattern value="interval([0-9])*"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="id_c_token">
    <xs:restriction base="xs:ID">
      <xs:pattern value="token([0-9])*"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="idref_c_token">
    <xs:restriction base="xs:IDREF">
      <xs:pattern value="token([0-9])*"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="id_c_word">
    <xs:restriction base="xs:ID">
      <xs:pattern value="word([0-9])*"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="idref_c_word">
    <xs:restriction base="xs:IDREF">
      <xs:pattern value="word([0-9])*"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="id_c_symb">
    <xs:restriction base="xs:ID">
      <xs:pattern value="symb([0-9])*"/>
    </xs:restriction>
```

```
</xs:simpleType>

<xs:simpleType name="idref_c_symb">
  <xs:restriction base="xs:IDREF">
    <xs:pattern value="symb([0-9])*"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="id_c_sent">
  <!-- CONSTITUENTS. A constituent is defined as a word or a group of words which acts syntactically as a unit. -->
  <!-- begins with a capital letter and ends with a full stop (period), a question mark, or an exclamation mark.-->
  <!-- types: -->
  <!-- simple sentence, complex sentence (matrix+subordinates), compound sentence (clauses that are coordinated) -->
  <!-- Discourse Functions -->
  <!-- declarative, interrogative, imperative, exclamative -->
  <!-- interrogative : YES/NO INTERROGATIVES, ALTERNATIVE INTERROGATIVES, WH- INTERROGATIVES, TAG QUESTIONS -->
  <!-- hierarchy : sentence, clause, phrase, word -->
  <!-- 1. Although clauses are higher than phrases in the hierarchy, clauses can occur within phrases -->
  <!-- -->
  <xs:restriction base="xs:ID">
    <xs:pattern value="sent([0-9])*"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="idref_c_sent">
  <xs:restriction base="xs:IDREF">
    <xs:pattern value="sent([0-9])*"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="id_c_seqword">
  <xs:restriction base="xs:ID">
    <xs:pattern value="seqword([0-9])*"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="idref_c_seqword">
  <xs:restriction base="xs:IDREF">
    <xs:pattern value="seqword([0-9])*"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="id_r_seqword">
  <xs:restriction base="xs:ID">
    <xs:pattern value="rseqword([0-9])*"/>
  </xs:restriction>
```

```
</xs:simpleType>

<xs:simpleType name="idccat">
  <xs:restriction base="xs:ID">
    <xs:pattern value="cat([0-9])*"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="idref_c_cat">
  <xs:restriction base="xs:IDREF">
    <xs:pattern value="cat([0-9])*"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="id_r_cat">
  <xs:restriction base="xs:ID">
    <xs:pattern value="rcat([0-9])*"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="id_c_punct">
  <xs:restriction base="xs:ID">
    <xs:pattern value="punct([0-9])*"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="idref_c_punct">
  <xs:restriction base="xs:IDREF">
    <xs:pattern value="punct([0-9])*"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="id_c_cat_punct">
  <xs:restriction base="xs:ID">
    <xs:pattern value="catpunct([0-9])*"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="idref_c_cat_punct">
  <xs:restriction base="xs:IDREF">
    <xs:pattern value="catpunct([0-9])*"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="id_r_cat_punct">
  <xs:restriction base="xs:ID">
```

```
      <xs:pattern value="rcatpunct([0-9])*"/>
    </xs:restriction>
  </xs:simpleType>

  <!-- remplacer cat par tiquette morpho synt ou catpunct ? -->

  <xs:simpleType name="id_c_seqcat">
    <xs:restriction base="xs:ID">
      <xs:pattern value="seqcat([0-9])*"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="idref_c_seqcat">
    <xs:restriction base="xs:IDREF">
      <xs:pattern value="seqcat([0-9])*"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="id_r_seqcat">
    <xs:restriction base="xs:ID">
      <xs:pattern value="rseqcat([0-9])*"/>
    </xs:restriction>
  </xs:simpleType>

  <!-- remplacer val de cat et punct par enumeration -->

  <xs:simpleType name="category">
    <xs:restriction base="xs:string">
      <xs:enumeration value="verb"/>
      <!--Verb   be, drive, grow, sing, think -->
      <!-- characteristics : -->
      <!-- baseform+inflection -->
      <!-- depends on tense, person x singular|plural form -->
      <!-- agreement or concord -->
      <!-- To-infinitive Bare infinitive -->
      <!-- -ING PARTICIPLE form and -ED PARTICIPLE form. -->
      <!-- Verbs which have the past or the present form are called FINITE verbs. Verbs in any other form (infinitive, -ing, or -ed) are called NONFINITE verbs -->
      <xs:enumeration value="noun"/>
      <!--Noun  brother, car, David, house, London -->
      <!-- pronoun  here, common noun / proper noun -->
      <!-- characteristics : singular/plural, count noun / non-count nouns, -->
      <!-- personal pronouns : person (1st,2nd,3rd), case (subjective={I, you, he}, objective={me, you, him})-->
      <!-- case and number and genre : personnal, possessive, reflexive -->
      <!-- other pronouns: -->
      <!-- possessive (mine, ours) -->
      <!-- reflexive (myself, ourselves) -->
```

```
    <!-- reciprocal (each other, one another) -->
    <!-- relative (that, which, who, whose, whom, where, when) -->
    <!-- demontrative (this, that, these, those) -->
    <!-- interrogative (who, what, why, where, when, whatever) -->
    <!-- indefinite (any...,some...,no...)x(thing,body,one) -->
    <!-- noun -->
    <!-- genitive : 's -->
    <!-- numeral -->
    <!-- ordinal or cardinal -->
    <!-- noun (plural, ..) or determiner -->
    <xs:enumeration value="determiner"/>
    <!--Determiner  a, an, my, some, the -->
    <!-- close -->
    <!-- characteristics : -->
    <!-- predeterminer : multiplying, fractions, aal et both, -->
    <!-- central determiner : the, an, possessive, demonstrative -->
    <!-- postdeterminer : cardinal et ordinal numeral, general ordinals (next), quantifying expression; can cooccurs -->
    <!-- definite or indefinite article -->
    <!-- personal pronouns and determiner -->
    <!-- that et this are determiner et pronouns -->
    <!-- personal pronouns (I, you, he, etc) cannot be determiners -->
    <!-- possessive pronouns (mine, yours, his/hers, ours, and theirs) have corresponding forms which are determiners -->
    <!-- definite and the indefinite articles can never be pronouns, they are always determiners. -->
    <xs:enumeration value="adjective"/>
    <!--Adjective  big, foolish, happy, talented, tidy -->
    <!-- characteristics : -->
    <xs:enumeration value="adverb"/>
    <!--Adverb  happily, recently, soon, then, there -->
    <!-- characteristics : -->
    <xs:enumeration value="preposition"/>
    <!--Preposition  at, in, of, over, with -->
    <!-- close -->
    <!-- characteristics : -->
    <xs:enumeration value="conjunction"/>
    <!--Conjunction  and, because, but, if, or -->
    <!-- close -->
    <!-- characteristics : -->
  </xs:restriction>
</xs:simpleType>

<xs:complexType name="type_c_phrase">
  <xs:sequence>
    <xs:element name="id" type="id_c_phrase"/>
    <xs:element name="assume" type="xs:string"/>
    <xs:element name="seq" type="idref_c_seqword"/>
    <xs:element name="head" type="idref_c_word"/>
```

```
    </xs:sequence>
  </xs:complexType>

  <xs:simpleType name="id_c_phrase">
    <xs:restriction base="xs:ID">
      <xs:pattern value="(noun|verb|adjective|adverb|prepositional)_phrase([0-9])*"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="phrase_type">
    <!-- Phrase : pre-head head post-head -->
    <xs:restriction base="xs:string">
      <xs:enumeration value="noun_phrase"/>
      <!-- Noun Phrase   Noun   [the children in class 5] -->
      <!-- pre-head = adjective, determiner; head=noun,pronoun,number; post-head mya be long -->
      <xs:enumeration value="verb_phrase"/>
      <!-- Verb Phrase  Verb  [play the piano] -->
      <!-- Verbs which require a post-Head string are called TRANSITIVE verbs -->
      <!-- Some verbs are never followed by a direct object : These are known as INTRANSITIVE VERBS. -->
      <!-- pre-head=negative word (not never), adverb; head=verb; post-head= -->
      <xs:enumeration value="adjective_phrase"/>
      <!-- Adjective Phrase  Adjective  [delighted to meet you] -->
      <!-- pre-head=adv phrase; head=adj; post-head=string-->
      <xs:enumeration value="adverb_phrase"/>
      <!-- Adverb Phrase  Adverb  [very quickly] -->
      <!-- pre-head=adv phrase;head=adv;post-head=rare -->
      <xs:enumeration value="prepositional_phrase"/>
      <!-- Prepositional Phrase  Preposition  [in the garden] -->
      <!-- pre-head=rare;head=proposition;post-head=string -->
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="id_c_clause">
    <!-- a finite clause contains a finite verb (conjugu) -->
    <!-- a matrix clause contains a finite verb (conjugu) -->
    <!-- a subordinate clause may contain a finite or nonfinite verb (conjugu) : -->
    <!-- TO-INFINITIVE CLAUSE, BARE INFINITIVE CLAUSE, -ING PARTICIPLE CLAUSE, -ED PARTICIPLE CLAUSE, IF-CLAUSE -->
    <!-- THAT-CLAUSE (that may be ellipted), RELATIVE CLAUSE, REDUCED RELATIVE CLAUSE, NOMINAL RELATIVE CLAUSES, SMALL CLAUSE -->
    <!-- semantic type -->
    <!-- Temporal, Conditional, Concessive, Reason, Result, Comparative -->
    <xs:restriction base="xs:ID">
      <xs:pattern value="clause([0-9])*"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="id_r_ordinate_clause">
```

```
    <xs:restriction base="xs:ID">
      <xs:pattern value="r_ordinate_clause([0-9])*"/>
    </xs:restriction>
  </xs:simpleType>

  <!-- Functional analysis -->
  <!-- functions in sentences -->
  <!-- Subject + predicate -->
  <!-- Subject : -->
  <!-- Clauses functioning as Subject -->
  <!-- Finite : That-clause, Nominal Relative clause -->
  <!-- Nonfinite : To-infinitive clause, -ing clause -->
  <!-- some of these Subject clauses have Subjects of their own -->
  <!-- Prepositional Phrases functioning as Subject : -->
  <!-- Less commonly, the Subject may be realised by a prepositional phrase: Prepositional phrases as Subject typically refer to time or to space. -->
  <!-- Some Unusual Subjects : it and there -->
  <!-- Predicate : -->
  <!-- predicator : always realized by a verb -->
  <!-- direct object : NP or clauses (finite=that-clause or nominal relative clause, nonfinite=to-infinitive or bare infinitive or -ing or -ed) -->
  <!-- indirect object (before direct) : NP or clauses -->
  <!-- verb with indirect and direct object = ditransitive -->
  <!-- verb with can take only direct object = monotransitive -->
  <!-- adjuncts (how,when,where) : NP,AdvP,PP,clause (finite, Nonfinite = To-infinitive or Bare infinitive or -ing or -ed  or Small) -->

  <!-- un -->
  <!-- Extraposition (the typical declarative order has been disrupted for stylistic effect, when the sentence is introduced by anticipatory it) -->
  <!-- cleft sentences : It + be + focus + clause -->

  <!-- functions in phrases-->
  <!-- in VP -->
  <!-- post-head sringS (not always) complementS (not always direct objects) -->
  <!-- copular verbs : be, become, seem, felt, sound -->
  <!-- copular verb => copular complement -->
  <!-- Noun Phrase : noun : PP or clause -->
  <!-- Verb Phrase : verb : NP or clause or PP-->
  <!-- Adjective Phrase : adjective : clause or PP -->
  <!-- Adverb Phrase : adverb : PP -->
  <!-- Prepositional Phrase : preposition or NP or PP -->
  <!-- Adjuncts in phrases : -->
  <!-- Noun Phrase : noun : PP or AP clause -->
  <!-- Verb Phrase : verb : AdvP or PP -->
  <!-- Adjective Phrase : adjective : AdvP -->
  <!-- Prepositional Phrase : preposition : AdvP -->
  <!-- 1. Complements immediately follow the Head -->
  <!-- 2. Adjuncts are "stackable" -->
  <!-- adjuncts may appear before the Head of the phrase -->
```

```
  <!-- Specifier -->
  <!-- remaining part of the pre-head string -->
  <!-- Noun Phrase : noun : determiners -->
  <!-- Verb Phrase : verb : ‘negative’ elements -->
  <!-- Adjective Phrase : adjective : AdvP -->
  <!-- Prepositional Phrase : preposition : AdvP -->
  <!-- In functional terms, then, the three-part structure of a phrase can be summarised as -->
  <!-- (Specifier) - [Head - (Complement)] -->

</xs:schema>

<!-- remplacer seq par liste types -->
```