FoLiA: Format for Linguistic Annotation

Maarten van Gompel

24-03-2011

ILK Research Group
Induction of Linguistic Knowledge.

Tilburg University

Introduction

Annotation formats in the field

- Many ad-hoc and old-style annotation formats (CGN, Tadpole column format)
- Many theoretic and specialised annotation formats with limited scope
- Overly rich document encoding formats (TEI)
- Many conversions needed
- De-facto-standard for various ILK projects: SoNaR/DCOI

Limits Tadpole/Frog columned format

- Simplistic format, lacking expressiveness of XML
- Six/seven columns currently, covering full screen width
- Lacking even expressiveness to fully output Frog's data!

Limits of DCOI

- Not expressive enough for many kinds of annotation (such as sense annotation, correction annotation)
- Can't encode annotators
- Can't encode alternatives

Objectives

Objectives (1/2)

Objectives: Expressability, Extensability, Uniformity

- One generalised rich common XML-based format, supporting almost all we do at ILK
- Uniform and consistent paradigm for annotations of various kinds
- Built from a bottom-up application-oriented perspective
- Not committing to any particular tagset or language
- Encoding many different annotation aspects similtaneously in a single XML file
- Support for sense annotation (DutchSemCor)

Objectives

Objectives (2/2)

Objectives: Expressability, Extensability, Uniformity

- Support for complex corrections (Valkuil + Ticcl)
- Support for NER (HITIME)
- Support in Frog for reading and writing this format
- Founded on the DCOI format (our de-facto standard)
- Inter-operability with ISO Data Category Registry
- Unicode compliant
- Fully open-source

Annotations

Supported Annotations (1/2)

FoLiA supports the following linguistic annotations:

- Part-of-Speech tags (with features)
- Lemmatisation
- Spelling corrections on both a tokenised as well as an untokenised level.
- Domain tagging
- Semantic sense annotation (to be used in DutchSemCor)

Annotations

Supported Annotations (2/2)

FoLiA supports the following linguistic annotations:

- Named Entities / Multi-word units
- Syntactic Parses
- Dependency Relations
- Chunking
- Corrections

More to be added when needed (in collaboration with partners)

Format: Overall skeleton

```
<?xml version="1.0" encoding="utf-8"?>
<FoLiA xmlns="http://ilk.uvt.nl/FoLiA"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xml:id="example">
   <metadata>
     <annotations>
        ...
     </annotations>
     <!-- (Here IMDI or CMDI metadata can be inserted) -->
   </metadata>
   <text xml:id="example.text">
      ...
   </text>
</FoLiA>
```

Characteristic of overall format

1. Unique identifier for document as a whole
2. Metadata
   - Annotations - Declaration of all used annotations
   - May hold IMDI or CMDI metadata
3. Text

Format: Basic structure

```
<p xml:id="TEST.p.1">
    <t>This is a test. It has two sentences.</t>
    <s xml:id="TEST.p.1.s.1">
        <t>This is a test.</t>
        <w xml:id="TEST.p.1.s.1.w.1"><t>This</t></w>
        <w xml:id="TEST.p.1.s.1.w.2"><t>is</t></w>
        ..
    </s><s xml:id="TEST.p.1.s.2">...</s>
</p>
```

Characteristics of basic structure

1. **Structure Elements**: Paragraphs, Sentences, Words/Tokens
2. More: Division, Head, List, ListItem, Figure, Gap...
3. Unique identifiers (DCOI style by convention)
4. Text element (t) holds actual text. May occur untokenised on higher levels as well.

Paradigm: Annotation Categories

Three categories of annotation:

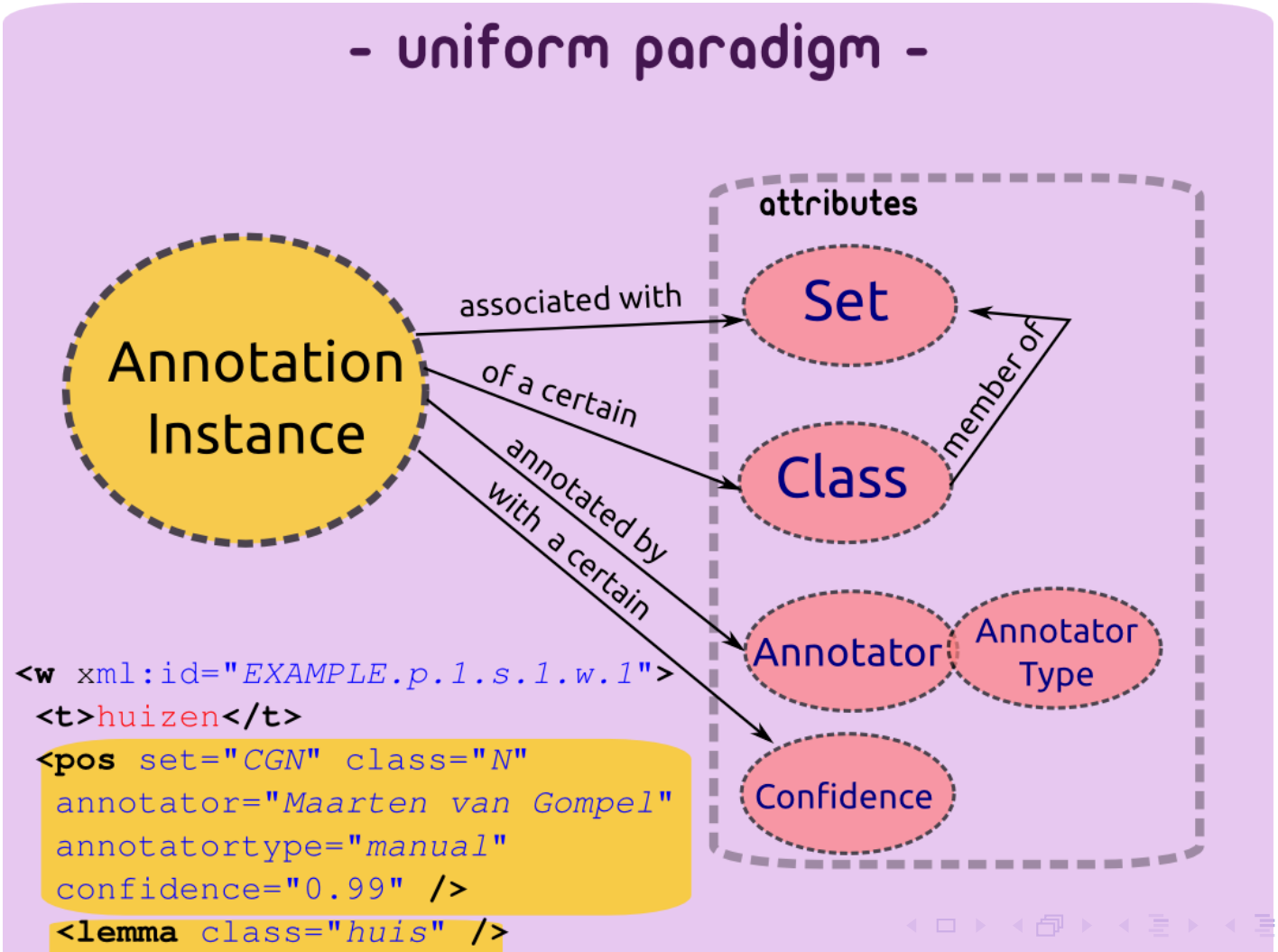
- **Structure Annotation** - Elements denoting document structure
  - E.g: Divisions, Header, Paragraphs, Sentences, Lists, Figures, Gaps, Quote
- **Token Annotation** - Linguistic Annotations pertaining to a single token (inline annotation)
  - E.g: Part of Speech Annotation, Lemma Annotation, Lexical Semantic Sense Annotation
- **Span Annotation** - Linguistic Annotations spanning over multiple tokens (standoff annotation)
  - E.g: Syntactic Parses, Dependency Relation, Entities/Multi-word Units

Paradigm: Common Attributes

FoLiA Attributes common to the paradigm.

- **ID** - A unique ID for the element
- **Set** - Identifier of a particular (tag)set
- **Class** - One class from the specified set
- **Annotator** - An open identifier for the user/system who provided the annotation
- **Annotator type** - “auto” or ”manual”
- **Confidence** - A confidence value between one and zero.
- **N** - A sequential number (for numbered divisions/sections, list items, etc)

Paradigm: Schematic



Token Annotation

Token annotation occurs within the scope of a word/token (*w*) element.

Example

PoS and Lemma Annotation:

```
<w xml:id="example.p.1.s.1.w.2">
    <t>boot</t>
    <pos set="brown" class="n"
     annotator="Maarten van Gompel" annotatortype="manual" />
    <lemma set="english-lemmas" class="boot" />
</w>
```

Token Annotation

All annotations need to be declared in the metadata:

- Default sets and annotator *may* be predefined at this level

Example

```
<metadata>
 <annotations>
  <token-annotation />
  <pos-annotation set="brown" annotator="Maarten van Gompel"
     annotatortype="manual"/>
  <lemma-annotation />
 </annotations>
</metadata>
```

Alternative Token Annotations

Encodes mutually exclusive alternative annotations. Any annotations that are not alternatives are considered "selected".

```
<w xml:id="example.p.1.s.1.w.2">
    <t>bank</t>
    <sense set="wordnet3.0" class="bank%1:17:01:"
     annotator="Maarten van Gompel" annotatortype="manual"
     confidence="0.8">
     sloping ground near water</sense>
    <alt xml:id="example.p.1.s.1.w.2.alt.1">
     <sense set="wordnet3.0" class="bank%1:14:01:"
      annotator="WSDsystem" annotatortype="auto"
      confidence="0.6">
      financial institution</sense>
    </alt>
</w>
```

Alternative Token Annotations

All token annotations grouped as one alternative are considered dependent. Multiple alternatives are always independent:

```
<w xml:id="example.p.1.s.1.w.2">
    <t>vlieg</t>
    <pos class="N" />
    <lemma class="vlieg" />
    <alt xml:id="example.p.1.s.1.w.2.alt.1">
        <pos class="V" />
        <lemma class="vliegen" />
    </alt>
</w>
```

Alternative Token Annotations

Annotations of the same type, but different sets need *not* be alternatives.

```
<w xml:id="example.p.1.s.1.w.2">
    <t>luid</t>
    <pos set="brown" class="jj" />
    <pos set="cgn" class="adj" />
</w>
```

There can be only one of the same set though, this is illegal and requires usage of alternatives instead:

```
<w xml:id="example.p.1.s.1.w.2">
    <t>luid</t>
    <pos set="cgn" class="adj" />
    <pos set="cgn" class="adv" />
</w>
```

Complex Token Annotations

Some annotation types are more complex, as they allow multiple classes, subclasses or classes with values. For example, Part-of-Speech annotation enriched with features:

```
<w xml:id="example.p.1.s.1.w.2">
    <t>boot</t>
    <pos set="cgn" class="N">
        <feat class="ntype" value="soort" />
        <feat class="number" value="ev" />
        <feat class="degree" value="basis" />
        <feat class="gender" value="zijd" />
        <feat class="case" value="stan" />
    </pos>
</w>
```

Span Annotation

- Token Annotation not sufficient, some annotations span over multiple tokens
- Spanning multiple tokens can produce nesting problems (e.g $A(BC)D$ and $AB(CD)$)
- **Solution:** Span Annotation using standoff notation
- **Applications:** Syntactic Parses, Chunking, Dependency Relations, Entities/Multi-Word Units
- **Layers:** Each type of span annotation is placed within an *annotation layer*, annotation layers are embedded within *sentences* (`s`))
- Same paradigm: Set, class, annotator, confidence

```
<s xml:id="example.p.1.s.1">
  <t>The Dalai Lama greeted him.</t>
  <w xml:id="example.p.1.s.1.w.1"><t>The</t></w>
  <w xml:id="example.p.1.s.1.w.2"><t>Dalai</t></w>
  <w xml:id="example.p.1.s.1.w.3"><t>Lama</t></w>
  <w xml:id="example.p.1.s.1.w.4"><t>greeted</t></w>
  <w xml:id="example.p.1.s.1.w.5"><t>him</t></w>
  <w xml:id="example.p.1.s.1.w.6"><t>.</t></w>
  <entities>
    <entity xml:id="example.p.1.s.1.entity.1" class="person">
        <wref xml:id="example.p.1.s.1.w.2" />
        <wref xml:id="example.p.1.s.1.w.3" />
    </entity>
  </entities>
</s>
```

```
<syntax>
<su xml:id="example.p.1.s.1.su.1" class="s">
  <su xml:id="example.p.1.s.1.su.1_1" class="np">
      <su xml:id="example.p.1.s.1.su.1_1_1" class="det">
         <wref xml:id="example.p.1.s.1.w.1" />
      </su>
      <su xml:id="example.p.1.s.1.su.1_1_2" class="pn">
         <wref xml:id="example.p.1.s.1.w.2" />
         <wref xml:id="example.p.1.s.1.w.3" />
      </su>
   </su>
 </su>
 <su xml:id="example.p.1.s.1.su.1_2" class="vp">
    <su xml:id="example.p.1.s.1.su.1_1_1" class="v">
       <wref xml:id="example.p.1.s.1.w.4" />
    </su>
    <su xml:id="example.p.1.s.1.su.1_1_2" class="pron">
      <wref xml:id="example.p.1.s.1.w.5" />
    </su>
 </su>
```

Span Annotation: Alternatives

Q: How to encode alternatives?

A: Place the annotation layer within the `altlayers` element.

Example

```
<s xml:id="example.p.1.s.1">
    <syntax>...</syntax>
    <altlayers>
        <syntax>...</syntax>
    </altlayers>
</s>
```

Corrections

- Keeping track of corrections (spelling corrections, annotation corrections)
- Important principle: Identifiers **never** change.

Corrections as Token Annotation (1/3)

```
<w xml:id="example.p.1.s.1.w.1">
    <t>tree</t>
    <correction xml:id="TEST-000000001.p.1.s.1.w.1.c.1"
        class="spelling">
        <new>
            <t>tree</t>
        </new>
        <original>
            <t>treee</t>
        </original>
    </correction>
</w>
```

Corrections as Token Annotation (2/3)

```
<w xml:id="example.p.1.s.1.w.1">
    <t>tree</t>
    <pos class="n" />
    <correction xml:id="TEST-000000001.p.1.s.1.w.1.c.1">
        <new>
            <pos class="n" />
        </new>
        <original>
            <pos class="v" />
        </original>
    </correction>

</w>
```

Corrections as Token Annotation (3/3)

```
<correction xml:id="TEST-000000001.p.1.s.1.w.1.c.2" class=
  annotator="Jane␣Doe" annotatortype="manual" confidence="
    <new>
        <t>tree</t>
    </new>
    <original>
        <correction xml:id="TEST-000000001.p.1.s.1.w.1.c.1
            annotator="John␣Doe" annotatortype="manual" con
             <new>
                 <t>three</t>
             </new>
             <original>
                 <t>treee</t>
             </original>
        </correction>
    </original>
</correction>
```

Corrections of words themselves: Merges

```
<s xml:id="example.p.1.s.1">
 <correction xml:id="example.p.1.s.1.c.1" class="merge">
    <new>
        <w xml:id="example.p.1.s.1.w.1-2">
            <t>online</t>
        </w>
    </new>
    <original>
        <w xml:id="example.p.1.s.1.w.1">
            <t>on</t>
        </w>
        <w xml:id="example.p.1.s.1.w.2">
            <t>line</t>
        </w>
    </original>
 </correction>
</s>
```

Corrections of words themselves: Splits

```
<s xml:id="example.p.1.s.1">
 <correction xml:id="example.p.1.s.1.c.1" class="split">
    <new>
        <w xml:id="example.p.1.s.1.w.1_1">
            <t>on</t>
        </w>
        <w xml:id="example.p.1.s.1.w.1_2">
            <t>line</t>
        </w>
    </new>
    <original>
        <w xml:id="example.p.1.s.1.w.1">
            <t>online</t>
        </w>
    </original>
 </correction>
</s>
```

Working with FoLiA: Querying with XPath

- XPath query for all paragraphs: `/FoLiA/text//p`
- XPath query for all sentences:
  `/FoLiA/text//s[not(parent::s)]`
- XPath query for all words:
  `/FoLiA/text//w[not(ancestor::original)]`
- XPath query for all words with lemma X:
  `/FoLiA/text//w[not(ancestor::original)]/lemma[@class == "X"]`
- XPath query for the text of all words:
  `/FoLiA/text//w[not(ancestor::original)]/t/child::text()`

Working with FoLiA: Python Library in PyNLPl

```
import pynlpl.formats.folia as folia
from fictional import lemmatise

doc = folia.Document(file="/path/to/folia/document.xml")
for word in doc.words():
    print "Word: " , str(word)
    print "PoS: " , word.select(AnnotationType.POS).cls

    if not word.has(AnnotationType.LEMMA):
        word.append( Lemma(cls=lemmatise( str(word) ),
        annotator="lemmatiserX", annotatortype=AnnotatorType.AUTO) )

    print "Lemma: " , word.select(AnnotationType.LEMMA).cls

doc.save()
```

Loose ends

- No format defined yet for definition of sets
- Links with ISO Data Category Registry.
- Validation (RelaxNG + Schematron)
- Inclusion of more annotation types
- Metadata, incorporation of CMDI

Future

- libfolia: FoLiA $C++$ library
- FoLiA support in Frog
- Converters
- Visualisation

Conclusion

- **Uniformity:** generic framework with simple paradigm, XML based
- **Expressability:** Ability to encode many kinds of linguistic annotation, including structural annotation, alternatives, and corrections
- **Expandability:** easy to add new annotations with the same paradigm