A toolkit for multi-dimensional markup
The development of SGF to XStandoff

Maik Stührenberg
Daniel Jettka
Bielefeld University

Overview

- Multi-dimensional annotation
- SGF and XStandoff
- The XStandoff Toolkit

Multi-dimensional annotation

Why?

- Often the combination of multiple (linguistic) resources is necessary to solve more complex tasks
- Multiple annotation of the same primary data can be used to compare (linguistic) resources

Multi-dimensional annotation

A simple example:

```
A star shines brighter.
```

Multi-dimensional annotation

A simple example:

- Multiple annotations (in separate files, aka Twin Documents)...

```
<morphemes xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="morphemes.xsd">
  <m>A</m>
  <m>star</m>
  <m>shine</m>
  <m>s</m>
  <m>bright</m>
  <m>er</m>.
</morphemes>
```

```
<syllables xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="syllables.xsd">
 <s>A</s>
 <s>star</s>
 <s>shines</s>
 <s>brigh</s>
 <s>ter</s>.
</syllables>
```

Multi-dimensional annotation

A simple example:

- Multiple annotations (in separate files, aka Twin Documents)...
  ... results in primary data saved redundantly

```
<morphemes xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="morphemes.xsd">
  <m>A</m>
  <m>star</m>
  <m>shine</m>
  <m>s</m>
  <m>bright</m>
  <m>er</m>.
</morphemes>
```

```
<syllables xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="syllables.xsd">
 <s>A</s>
 <s>star</s>
 <s>shines</s>
 <s>brigh</s>
 <s>ter</s>.
</syllables>
```

Multi-dimensional annotation

A simple example:

- Multiple annotations (in separate files, aka Twin Documents)...
  ... results in primary data saved redundantly
- An integrated format would ease analyzing and exchanging data but...

```
<morphemes xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="morphemes.xsd">
  <m>A</m>
  <m>star</m>
  <m>shine</m>
  <m>s</m>
  <m>bright</m>
  <m>er</m>.
</morphemes>
```

```
<syllables xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="syllables.xsd">
 <s>A</s>
 <s>star</s>
 <s>shines</s>
 <s>brigh</s>
 <s>ter</s>.
</syllables>
```

Multi-dimensional annotation

A simple example:

- Multiple annotations (in separate files, aka Twin Documents)...
  ... results in primary data saved redundantly
- An integrated format would ease analyzing and exchanging data but...
  ...results in overlapping markup

```
<morphemes xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="morphemes.xsd">
  <m>A</m>
  <m>star</m>
  <m>shine</m>
  <m>s</m>
  <m>bright</m>
  <m>er</m>.
</morphemes>
```

```
<syllables xmlns:xsi="http://www.w3.org/2001/XMLSchema-instar
  xsi:noNamespaceSchemaLocation="syllables.xsd">
 <s>A</s>
 <s>star</s>
 <s>shines</s>
 <s>brigh</s>
 <s>ter</s>.
</syllables>
```

```
<syllables>
 <morphemes>
  <s><m>A</m></s>
  <s><m>star</m></s>
  <s><m>shine</m>
  <m>s</m></s>
  <s><m>brigh</s>
  <s>t</m>
  <m>er</m></s>.
 </morphemes>
</syllables>
```

Multi-dimensional annotation – Architectures

This problem exists only in inline XML. Architectures capable of storing multi-dimensional annotation developed in the last years:

- Prolog-based
  cf. C. M. Sperberg-McQueen et al. (2000); Witt et al. (2005)
- XML-related
  - GODDAG/TexMecs
  - LMNL
  - Multi-colored trees
  - XML delay nodes
  - XConcur
- Graph based formats
  - NITE
  - LAF/GrAF
  - Feature Structures
  - EARMARK

Multi-dimensional annotation – Architectures

All of these architectures have some trade-offs or disadvantages:

- Prolog-based
  - Conversion needed
  - quite complex and huge files
- XML-related
  - Multiple inline annotation can get very complex
  - Future development is often unknown, Validation languages are work in progress as well
  - Lack of support of the large number of tools that is available for XML-based solutions

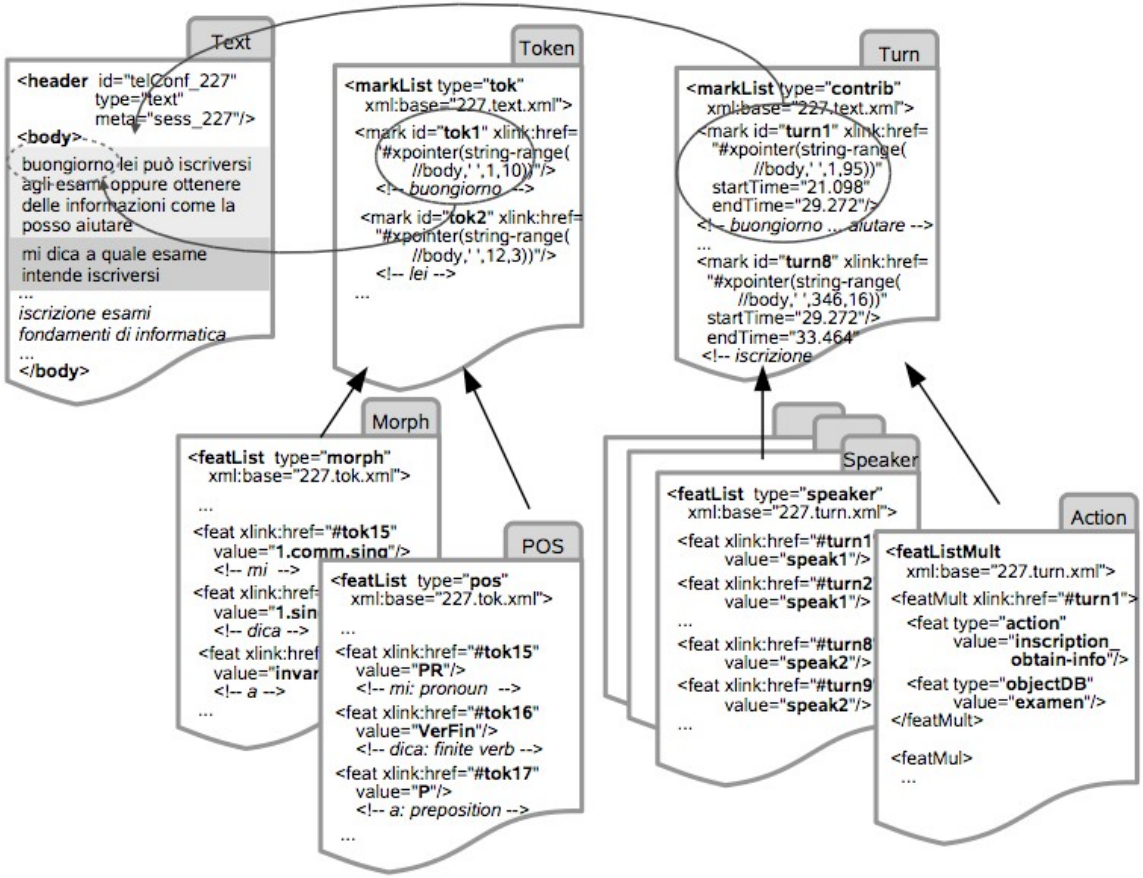
Multi-dimensional annotation – Graph-based architectures

In principle, graph-based annotation formats are quite nice for the task of dealing with multiple annotated data – and allow for staying in the XML context...

Multi-dimensional annotation – Graph-based architectures

...however, there are some disadvantages of the existing graph-based approaches:

- Often (even single) annotations are split into multiple files:
  - primary data file
  - markable/token file
    delimits text spans used in annotation
  - structure file
    stores relations between annotation elements
  - feature file
    stores the former annotation



- Graph-based architectures tend to provide interchange formats, not annotation formats

Figure taken from K. J. Rodríguez et al., 2007

Multi-dimensional annotation – Graph-based architectures

We prefer an architecture that...

- ...is mostly tree-based (i.e. it supports the XML-inherent embedding)
- ...stays as close to inline annotation as possible
- ...supports validation
- ...eases analyzation of multi-dimensional markup

SGF and XStandoff

Back to our simple example:

```
<morphemes xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="morphemes.xsd">
  <m>A</m>
  <m>star</m>
  <m>shine</m>
  <m>s</m>
  <m>bright</m>
  <m>er</m>.
</morphemes>
```

```
<syllables xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="syllables.xsd">
 <s>A</s>
 <s>star</s>
 <s>shines</s>
 <s>brigh</s>
 <s>ter</s>.
</syllables>
```

SGF and XStandoff

Back to our simple example:

- First, we delete the redundancy of primary data by introducing offsets

```
  A     s  t  a  r     s  h  i  n  e  s     b  r  i  g  h  t  e  r  .
00|01|02|03|04|05|06|07|08|09|10|11|12|13|14|15|16|17|18|19|20|21|22|23
```

```
<morphemes xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="morphemes.xsd" start="0" end="23">
  <m start="0" end="1"/>
  <m start="2" end="6"/>
  <m start="7" end="12"/>
  <m start="12" end="13"/>
  <m start="14" end="20"/>
  <m start="20" end="22"/>
</morphemes>
```

```
<syllables xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="syllables.xsd" start="0" end="23">
 <s start="0" end="1"/>
 <s start="2" end="6"/>
 <s start="7" end="13"/>
 <s start="14" end="19"/>
 <s start="19" end="22"/>
</syllables>
```

SGF and XStandoff

Back to our simple example:

- Still some redundancy left

```
<morphemes xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="morphemes.xsd" start="0" end="23">
  <m start="0" end="1"/>
  <m start="2" end="6"/>
  <m start="7" end="12"/>
  <m start="12" end="13"/>
  <m start="14" end="20"/>
  <m start="20" end="22"/>
</morphemes>
```

```
<syllables xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="syllables.xsd" start="0" end="23">
 <s start="0" end="1"/>
 <s start="2" end="6"/>
 <s start="7" end="13"/>
 <s start="14" end="19"/>
 <s start="19" end="22"/>
</syllables>
```

SGF and XStandoff

Back to our simple example:

- Almost there...

```
<morphemes xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="morphemes.xsd" seg="seg1">
  <m seg="seg2"/>
  <m seg="seg3"/>
  <m seg="seg5"/>
  <m seg="seg6"/>
  <m seg="seg7"/>
  <m seg="seg10"/>
</morphemes>
```

```
<syllables xmlns:xsi="http://www.w3.org/2001
  xsi:noNamespaceSchemaLocation="syllables.x
 <s seg="seg2"/>
 <s seg="seg3"/>
 <s seg="seg4"/>
 <s seg="seg8"/>
 <s seg="seg9"/>
</syllables>
```

```
<seg id="seg1" start="0" end="23"/>
<seg id="seg2" start="0" end="1"/>
<seg id="seg3" start="2" end="6"/>
<seg id="seg4" start="7" end="13"/>
<seg id="seg5" start="7" end="12"/>
<seg id="seg6" start="12" end="13"/>
<seg id="seg7" start="14" end="20"/>
<seg id="seg8" start="14" end="19"/>
<seg id="seg9" start="19" end="22"/>
<seg id="seg10" start="20" end="22"/>
```

SGF and XStandoff

Final steps:

- Putting it all together in a single file
  - XML ID/IDREF for linking segments of primary data with the respective annotation(s)
  - XML namespaces for separation of annotation layers
- Reuse as much available standards and formats as possible – try not to reinvent the wheel
  - Existing metadata specifications
  - Global XML attributes such as xml:lang or xml:id

```
<seg id="seg1" start="0" end="23"/>
<seg id="seg2" start="0" end="1"/>
<seg id="seg3" start="2" end="6"/>
<seg id="seg4" start="7" end="13"/>
<seg id="seg5" start="7" end="12"/>
<seg id="seg6" start="12" end="13"/>
<seg id="seg7" start="14" end="20"/>
<seg id="seg8" start="14" end="19"/>
<seg id="seg9" start="19" end="22"/>
<seg id="seg10" start="20" end="22"/>
<m:morphemes seg="seg1">
  <m:m seg="seg2"/>
  <m:m seg="seg3"/>
  <m:m seg="seg5"/>
  <m:m seg="seg6"/>
  <m:m seg="seg7"/>
  <m:m seg="seg10"/>
</m:morphemes>
<s:syllables seg="seg1">
 <s:s seg="seg2"/>
 <s:s seg="seg3"/>
 <s:s seg="seg4"/>
 <s:s seg="seg8/>
 <s:s seg="seg9"/>
</s:syllables>
```

SGF and XStandoff – The resulting XStandoff instance

```
<xsf:corpusData  xmlns="http://www.xstandoff.net/2009/xstandoff/1.1"
  xmlns:xsf="http://www.xstandoff.net/2009/xstandoff/1.1"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsfVersion="1.1"
  xsi:schemaLocation="http://www.xstandoff.net/2009/xstandoff/1.1 xsf.xsd">
  <xsf:primaryData start="0" end="23" mime-type="text/plain" encoding="UTF-8">
    <primaryDataRef uri="sentence.txt"/>
  </xsf:primaryData>
  <xsf:segmentation>
    <xsf:segment xml:id="seg1" type="char" start="0" end="23"/>
    <xsf:segment xml:id="seg2" type="char" start="0" end="1"/>
    <!-- [..] -->
  </xsf:segmentation>
  <xsf:annotation>
    <xsf:level xml:id="morphemes">
      <xsf:layer xmlns:m="http://www.xstandoff.net/morphemes" priority="0"
        xsi:schemaLocation="http://www.xstandoff.net/morphemes morphemes.xsd">
        <m:morphemes xsf:segment="seg1">
          <m:m xsf:segment="seg2"/>
          <!-- […] -->
        </m:morphemes>
      </xsf:layer>
    </xsf:level>
    <xsf:level xml:id="syllables">
      <xsf:layer xmlns:s="http://www.xstandoff.net/syllables" priority="0"
        xsi:schemaLocation="http://www.xstandoff.net/syllables  syllables.xsd">
        <s:syllables xsf:segment="seg1">
          <s:s xsf:segment="seg2"/>
          <!-- [..] -->
        </s:syllables>
      </xsf:layer>
    </xsf:level>
  </xsf:annotation>
</xsf:corpusData>
```

SGF and XStandoff – The resulting XStandoff instance

```
<xsf:corpusData  xmlns="http://www.xstandoff.net/2009/xstandoff/1.1"
  xmlns:xsf="http://www.xstandoff.net/2009/xstandoff/1.1"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsfVersion="1.1"
  xsi:schemaLocation="http://www.xstandoff.net/2009/xstandoff/1.1 xsf.xsd">
  <xsf:primaryData start="0" end="23" mime-type="text/plain" encoding="UTF-8">
    <primaryDataRef uri="sentence.txt"/>
  </xsf:primaryData>
  <xsf:segmentation>
    <xsf:segment xml:id="seg1" type="char" start="0" end="23"/>
    <xsf:segment xml:id="seg2" type="char" start="0" end="1"/>
    <!-- [..] -->
  </xsf:segmentation>
  <xsf:annotation>
    <xsf:level xml:id="morphemes">
      <xsf:layer xmlns:m="http://www.xstandoff.net/morphemes" priority="0"
        xsi:schemaLocation="http://www.xstandoff.net/morphemes morphemes.xsd">
        <m:morphemes xsf:segment="seg1">
          <m:m xsf:segment="seg2"/>
          <!-- […] -->
        </m:morphemes>
      </xsf:layer>
    </xsf:level>
    <xsf:level xml:id="syllables">
      <xsf:layer xmlns:s="http://www.xstandoff.net/syllables" priority="0"
        xsi:schemaLocation="http://www.xstandoff.net/syllables  syllables.xsd">
        <s:syllables xsf:segment="seg1">
          <s:s xsf:segment="seg2"/>
          <!-- [..] -->
        </s:syllables>
      </xsf:layer>
    </xsf:level>
  </xsf:annotation>
</xsf:corpusData>
```

SGF and XStandoff's base layer

SGF and XStandoff – The resulting XStandoff instance

```
<xsf:corpusData  xmlns="http://www.xstandoff.net/2009/xstandoff/1.1"
  xmlns:xsf="http://www.xstandoff.net/2009/xstandoff/1.1"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsfVersion="1.1"
  xsi:schemaLocation="http://www.xstandoff.net/2009/xstandoff/1.1 xsf.xsd">
  <xsf:primaryData start="0" end="23" mime-type="text/plain" encoding="UTF-8">
    <primaryDataRef uri="sentence.txt"/>
  </xsf:primaryData>
  <xsf:segmentation>
    <xsf:segment xml:id="seg1" type="char" start="0" end="23"/>
    <xsf:segment xml:id="seg2" type="char" start="0" end="1"/>
    <!-- [..] -->
  </xsf:segmentation>
  <xsf:annotation>
    <xsf:level xml:id="morphemes">
      <xsf:layer xmlns:m="http://www.xstandoff.net/morphemes" priority="0"
        xsi:schemaLocation="http://www.xstandoff.net/morphemes morphemes.xsd">
        <m:morphemes xsf:segment="seg1">
          <m:m xsf:segment="seg2"/>
          <!-- […] -->
        </m:morphemes>
      </xsf:layer>
    </xsf:level>
    <xsf:level xml:id="syllables">
      <xsf:layer xmlns:s="http://www.xstando
        xsi:schemaLocation="http://www.xstandoff.net/syllables  syllables.xsd">
        <s:syllables xsf:segment="seg1">
          <s:s xsf:segment="seg2"/>
          <!-- [..] -->
        </s:syllables>
      </xsf:layer>
    </xsf:level>
  </xsf:annotation>
</xsf:corpusData>
```

Primary data can be stored internally or externally – multiple primary data files are supported (e.g. for multi-modal annotation)

SGF and XStandoff – The resulting XStandoff instance

```
<xsf:corpusData  xmlns="http://www.xstandoff.net/2009/xstandoff/1.1"
  xmlns:xsf="http://www.xstandoff.net/2009/xstandoff/1.1"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsfVersion="1.1"
  xsi:schemaLocation="http://www.xstandoff.net/2009/xstandoff/1.1 xsf.xsd">
  <xsf:primaryData start="0" end="23" mime-type="text/plain" encoding="UTF-8">
    <primaryDataRef uri="sentence.txt"/>
  </xsf:primaryData>
  <xsf:segmentation>
    <xsf:segment xml:id="seg1" type="char" start="0" end="23"/>
    <xsf:segment xml:id="seg2" type="char" start="0" end="1"/>
    <!-- [..] -->
  </xsf:segmentation>
  <xsf:annotation>
    <xsf:level xml:id="morphemes">
      <xsf:layer xmlns:m="http://www.xstandoff.net/morphemes" priority="0"
        xsi:schemaLocation="http://www.xstandoff.net/morphemes morphemes.xsd">
        <m:morphemes xsf:segment="seg1">
          <m:m xsf:segment="seg2"/>
          <!-- […] -->
        </m:morphemes>
      </xsf:layer>
    </xsf:level>
    <xsf:level xml:id="syllables">
      <xsf:layer xmlns:s="http://www.xstandoff.net/syllables" priority="0"
        xsi:schemaLocation="http://www.xstandoff.net/syllables  syllables.xsd">
        <s:syllables xsf:segment="seg1">
          <s:s xsf:segment="seg2"/>
          <!-- [..] -->
        </s:syllables>
      </xsf:layer>
    </xsf:level>
  </xsf:annotation>
</xsf:corpusData>
```

Segmentation of the primary data according to annotation layer(s)

SGF and XStandoff – The resulting XStandoff instance

```
<xsf:corpusData  xmlns="http://www.xstandoff.net/2009/xstandoff/1.1"
  xmlns:xsf="http://www.xstandoff.net/2009/xstandoff/1.1"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsfVersion="1.1"
  xsi:schemaLocation="http://www.xstandoff.net/2009/xstandoff/1.1 xsf.xsd">
  <xsf:primaryData start="0" end="23" mime-type="text/plain" encoding="UTF-8">
    <primaryDataRef uri="sentence.txt"/>
  </xsf:primaryData>
  <xsf:segmentation>
    <xsf:segment xml:id="seg1" type="char" start="0" end="23"/>
    <xsf:segment xml:id="seg2" type="char" start="0" end="1"/>
    <!-- [..] -->
  </xsf:segmentation>
  <xsf:annotation>
    <xsf:level xml:id="morphemes">
      <xsf:layer xmlns:m="http://www.xstandoff.net/morphemes" priority="0"
        xsi:schemaLocation="http://www.xstandoff.net/morphemes morphemes.xsd">
        <m:morphemes xsf:segment="seg1">
          <m:m xsf:segment="seg2"/>
          <!-- […] -->
        </m:morphemes>
      </xsf:layer>
    </xsf:level>
    <xsf:level xml:id="syllables">
      <xsf:layer xmlns:s="http://www.xstandoff.net/syllables" priority="0"
        xsi:schemaLocation="http://www.xstandoff.net/syllables  syllables.xsd">
        <s:syllables xsf:segment="seg1">
          <s:s xsf:segment="seg2"/>
          <!-- [..] -->
        </s:syllables>
      </xsf:layer>
    </xsf:level>
  </xsf:annotation>
</xsf:corpusData>
```

The annotation element stores one or more annotation level(s)

SGF and XStandoff – The resulting XStandoff instance

```
<xsf:corpusData  xmlns="http://www.xstandoff.net/2009/xstandoff/1.1"
  xmlns:xsf="http://www.xstandoff.net/2009/xstandoff/1.1"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsfVersion="1.1"
  xsi:schemaLocation="http://www.xstandoff.net/2009/xstandoff/1.1 xsf.xsd">
  <xsf:primaryData start="0" end="23" mime-type="text/plain" encoding="UTF-8">
    <primaryDataRef uri="sentence.txt"/>
  </xsf:primaryData>
  <xsf:segmentation>
    <xsf:segment xml:id="seg1" type="char" start="0" end="23"/>
    <xsf:segment xml:id="seg2" type="char" start="0" end="1"/>
    <!-- [..] -->
  </xsf:segmentation>
  <xsf:annotation>
    <xsf:level xml:id="morphemes">
      <xsf:layer xmlns:m="http://www.xstandoff.net/morphemes" priority="0"
        xsi:schemaLocation="http://www.xstandoff.net/morphemes morphemes.xsd">
        <m:morphemes xsf:segment="seg1">
          <m:m xsf:segment="seg2"/>
          <!-- […] -->
        </m:morphemes>
      </xsf:layer>
    </xsf:level>
    <xsf:level xml:id="syllables">
      <xsf:layer xmlns:s="http://www.xstandoff.net/syllables" priority="0"
        xsi:schemaLocation="http://www.xstandoff.net/syllables  syllables.xsd">
        <s:syllables xsf:segment="seg1">
          <s:s xsf:segment="seg2"/>
          <!-- [..] -->
        </s:syllables>
      </xsf:layer>
    </xsf:level>
  </xsf:annotation>
</xsf:corpusData>
```

Each annotation level may contain one or more annotation layer(s)

SGF and XStandoff – The resulting XStandoff instance

```
<xsf:corpusData  xmlns="http://www.xstandoff.net/2009/xstandoff/1.1"
  xmlns:xsf="http://www.xstandoff.net/2009/xstandoff/1.1"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsfVersion="1.1"
  xsi:schemaLocation="http://www.xstandoff.net/2009/xstandoff/1.1 xsf.xsd">
  <xsf:primaryData start="0" end="23" mime-type="text/plain" encoding="UTF-8">
    <primaryDataRef uri="sentence.txt"/>
  </xsf:primaryData>
  <xsf:segmentation>
    <xsf:segment xml:id="seg1" type="char" start="0" end="23"/>
    <xsf:segment xml:id="seg2" type="char" start="0" end="1"/>
    <!-- [..] -->
  </xsf:segmentation>
  <xsf:annotation>
    <xsf:level xml:id="morphemes">
      <xsf:layer xmlns:m="http://www.xstandoff.net/morphemes" priority="0"
        xsi:schemaLocation="http://www.xstandoff.net/morphemes morphemes.xsd">
        <m:morphemes xsf:segment="seg1">
          <m:m xsf:segment="seg2"/>
          <!-- […] -->
        </m:morphemes>
      </xsf:layer>
    </xsf:level>
    <xsf:level xml:id="syllables">
      <xsf:layer xmlns:s="http://www.xstandoff.net/syllables" priority="0"
        xsi:schemaLocation="http://www.xstandoff.net/syllables  syllables.xsd">
        <s:syllables xsf:segment="seg1">
          <s:s xsf:segment="seg2"/>
          <!-- [..] -->
        </s:syllables>
      </xsf:layer>
    </xsf:level>
  </xsf:annotation>
</xsf:corpusData>
```

The converted representation of the original morpheme annotation layer

SGF and XStandoff

SGF...

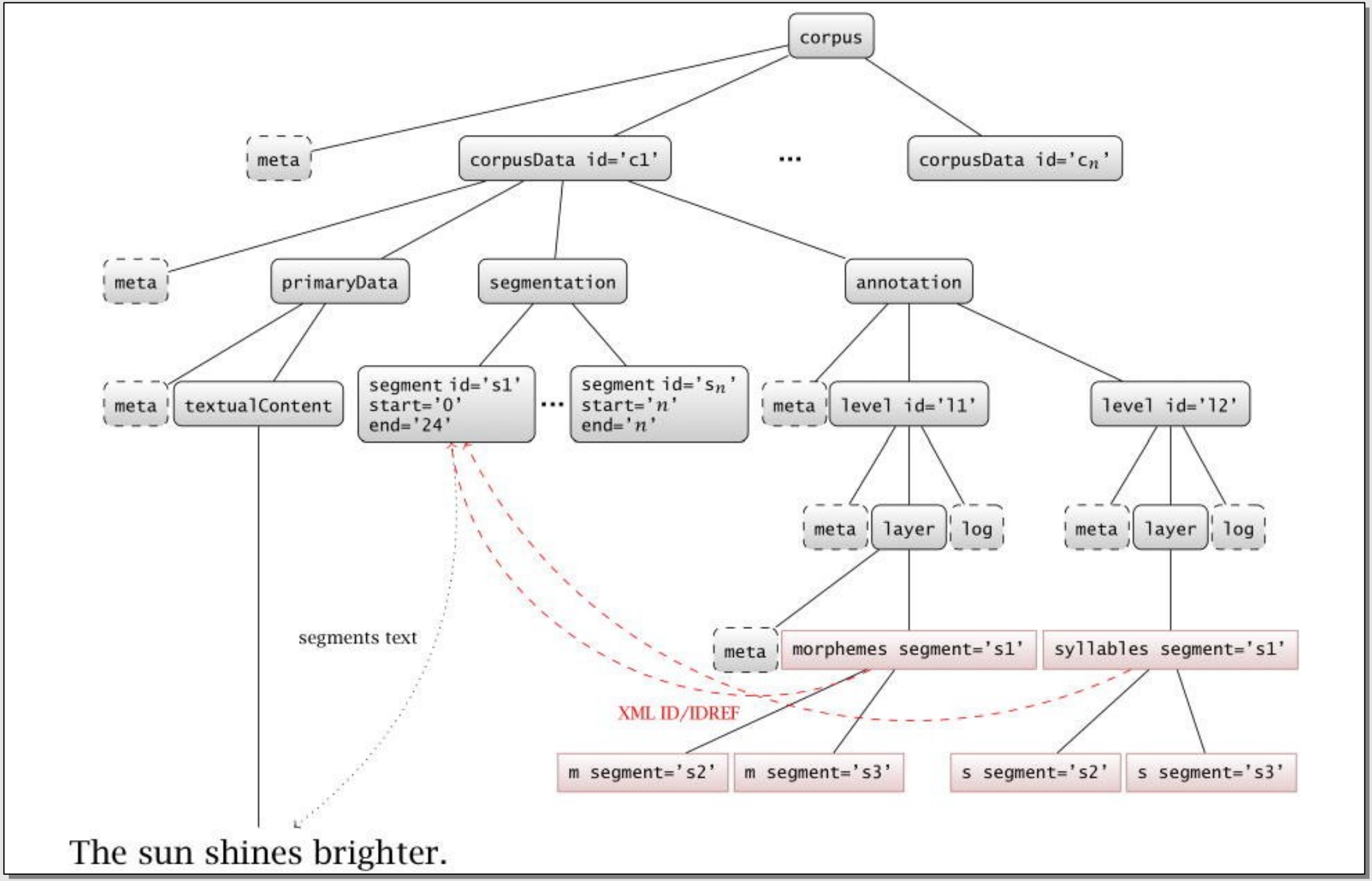
- ...was developed between 2006 and 2008 in the Sekimo project at Bielefeld University and presented at Balisage 2008
- ...is a logical descendant of the Prolog fact base format by Witt et al., 2005 – which itself is an multi layer markup extension of the format introduced by Sperberg-McQueen et al., 2000
- ...follows the Annotation Graph paradigm, i.e. annotations are aligned via character positions (if textual primary data is used) or positions in time
- ...stores multiple annotations in a single XML file, a native XML database, a relational database or a hybrid database
- ...uses standard (i.e. unmodified) XPath and XQuery for analyzing multi-dimensional annotations

SGF and XStandoff

XStandoff...

- ...is the successor of SGF developed between 2008 and 2009
- ...is a refinement in terms of naming conventions (camelCase for all element and attribute names)
- ...uses a better structured XML schema file (including named types)
- ...supports a new all namespace and an inline representation
- ...is accompanied by the respective XStandoff Toolkit

SGF and XStandoff



SGF and XStandoff

Some more words before we go on any further...

- Both, SGF and XStandoff are meta-markup languages
- The formal model has changed slightly from multi-rooted trees (SGF) to GODDAG-like structures (including discontinuous segments)
- Both, SGF and XStandoff rely heavily on XML Schema and XML Namespaces and support intra-layer and cross-layer validation
- The next slides will show mainly advancements that were made during the development of XStandoff, i.e., most of the following features are available in XStandoff only.

XStandoff – Disjoint elements

Sometimes one wants to annotate disjoint elements

```
<p>
Alice was beginning to get very tired of sitting by her sister on the bank, and of
having nothing to do: once or twice she had peeped into the book her sister was
reading, but it had no pictures or conversations in it, <q>and what is the use of a
book,</q> thought Alice <q>without pictures or conversation?</q>
</p>
```

- Quoting C. M. Sperberg-McQueen and C. Huitfeldt at last year's Balisage, the inline annotation "does not capture the intuitive sense the reader may have that the reported thoughts of Alice form a single continuous unitary utterance"
- There are different options to cope with this:
  - TEI's part attribute (with the I(nitial), M(iddle), and F(inal) value
  - TEI's next and previous attribute
  - TEI's join element
  - Using TexMecs or other XML-related specifications

XStandoff – Disjoint elements

The XStandoff way is similar to the mentioned TEI join element:

```
<xsf:corpusData xmlns="http://www.xstandoff.net/2009/xstandoff/1.1"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsf="http://www.xstandoff.net/2009/xstandoff/1.1"
 xsi:schemaLocation="http://www.xstandoff.net/2009/xstandoff/1.1 xsf.xsd"
 sgfVersion="1.1" xml:id="alice">
  <xsf:primaryData start="0" end="302">
   <primaryDataRef uri="alice.txt" encoding="UTF-8"/>
  </xsf:primaryData>
  <xsf:segmentation>
    <xsf:segment xml:id="seg1" type="char" start="0" end="302"/>
    <xsf:segment xml:id="seg2" type="char" start="218" end="250"/>
    <xsf:segment xml:id="seg3" type="char" start="266" end="302"/>
    <xsf:segment xml:id="seg4" type="seg" segments="seg2 seg3" mode="disjoint"/>
  </xsf:segmentation>
  <xsf:annotation>
    <xsf:level xml:id="alice-log">
      <xsf:layer xmlns:log="http://www.xstandoff.net/alice/log"
       xsi:schemaLocation="http://www.xstandoff.net/alice/log alice-log.xsd">
        <log:text xsf:segment="seg1">
          <log:p xsf:segment="seg1">
            <log:q xsf:segment="seg4"/>
          </log:p>
        </log:text>
      </xsf:layer>
    </xsf:level>
  </xsf:annotation>
</xsf:corpusData>
```

XStandoff – Disjoint elements

The XStandoff way is similar to the mentioned TEI join element:

```
<xsf:corpusData xmlns="http://www.xstandoff.net/2009/xstandoff/1.1"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsf="http://www.xstandoff.net/2009/xstandoff/1.1"
 xsi:schemaLocation="http://www.xstandoff.net/2009/xstandoff/1.1 xsf.xsd"
 sgfVersion="1.1" xml:id="
  <xsf:primaryData start="
   <primaryDataRef uri="al
  </xsf:primaryData>
  <xsf:segmentation>
    <xsf:segment xml:id="seg1" type="char" start="0" end="302"/>
    <xsf:segment xml:id="seg2" type="char" start="218" end="250"/>
    <xsf:segment xml:id="seg3" type="char" start="266" end="302"/>
    <xsf:segment xml:id="seg4" type="seg" segments="seg2 seg3" mode="disjoint"/>
  </xsf:segmentation>
  <xsf:annotation>
    <xsf:level xml:id="alice-log">
      <xsf:layer xmlns:log="http://www.xstandoff.net/alice/log"
       xsi:schemaLocation="http://www.xstandoff.net/alice/log alice-log.xsd">
        <log:text xsf:segment="seg1">
          <log:p xsf:segment="seg1">
            <log:q xsf:segment="seg4"/>
          </log:p>
        </log:text>
      </xsf:layer>
    </xsf:level>
  </xsf:annotation>
</xsf:corpusData>
```

Firstly, two segments are established referring to the spans of each part of the quote

XStandoff – Disjoint elements

The XStandoff way is similar to the mentioned TEI join element:

```
<xsf:corpusData xmlns="http://www.xstandoff.net/2009/xstandoff/1.1"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsf="http://www.xstandoff.net/2009/xstandoff/1.1"
 xsi:schemaLocation="http://www.xstandoff.net/2009/xstandoff/1.1 xsf.xsd"
 sgfVersion="1.1" xml:id="
  <xsf:primaryData start="
   <primaryDataRef uri="al
  </xsf:primaryData>
  <xsf:segmentation>
    <xsf:segment xml:id="s
    <xsf:segment xml:id="s
    <xsf:segment xml:id="seg3" type="char" start="266" end="302"/>
    <xsf:segment xml:id="seg4" type="seg" segments="seg2 seg3" mode="disjoint"/>
  </xsf:segmentation>
  <xsf:annotation>
    <xsf:level xml:id="alice-log">
      <xsf:layer xmlns:log="http://www.xstandoff.net/alice/log"
       xsi:schemaLocation="http://www.xstandoff.net/alice/log alice-log.xsd">
        <log:text xsf:segment="seg1">
          <log:p xsf:segment="seg1">
            <log:q xsf:segment="seg4"/>
          </log:p>
        </log:text>
      </xsf:layer>
    </xsf:level>
  </xsf:annotation>
</xsf:corpusData>
```

Next, a new segment is established by referring to the already available segments.
The disjoint value of the mode attribute signals this segment as discontinuous span

XStandoff – Disjoint elements

The XStandoff way is similar to the mentioned TEI join element:

```
<xsf:corpusData xmlns="http://www.xstandoff.net/2009/xstandoff/1.1"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsf="http://www.xstandoff.net/2009/xstandoff/1.1"
 xsi:schemaLocation="http://www.xstandoff.net/2009/xstandoff/1.1 xsf.xsd"
 sgfVersion="1.1" xml:id="
  <xsf:primaryData start="
   <primaryDataRef uri="al
  </xsf:primaryData>
  <xsf:segmentation>
    <xsf:segment xml:id="seg1" type="char" start="0" end="302"/>
    <xsf:segment xml:id="seg2" type="char" start="218" end="250"/>
    <xsf:segment xml:id="seg3" type="char" start="266" end="302"/>
    <xsf:segment xml:id="seg4" type="seg" segments="seg2 seg3" mode="disjoint"/>
  </xsf:segmentation>
  <xsf:annotation>
    <xsf:level xml:id="alice-log">
      <xsf:layer xmlns:log="http://www.xstandoff.net/alice/log"
       xsi:schemaLocation="http://www.xstandoff.net/alice/log alice-log.xsd">
        <log:text xsf:segment="seg1">
          <log:p xsf:segment="seg1">
            <log:q xsf:segment="seg4"/>
          </log:p>
        </log:text>
      </xsf:layer>
    </xsf:level>
  </xsf:annotation>
</xsf:corpusData>
```

Finally, we use the new segment in the annotation layer.

XStandoff – Containment and dominance

The same example can be used for demonstrating the support for differentiating between containment and dominance in XStandoff:

```
<p>
Alice was beginning to get very tired of sitting by her sister on the bank, and of
having nothing to do: once or twice she had peeped into the book her sister was
reading, but it had no pictures or conversations in it, <q>and what is the use of a
book,</q> thought Alice <q>without pictures or conversation?</q>
</p>
```

The q element is contained in p. However, there are three possible versions of the dominance relation between p and q:

- p does not dominate q
- p dominates the q fragments
- p dominates the discontinuous entire q

For a discussion of containment and dominance cf. C. M. Sperberg-McQueen and C. Huitfeldt, "Markup Discontinued: Discontinuity in TexMecs, Goddag structures, and rabbit/duck grammars." Presented at Balisage: The Markup Conference 2008, Montréal, Canada, August 12-15, 2008.

XStandoff – Containment and dominance

Version 1: p does not dominate q

```
<xsf:corpusData xmlns="http://www.xstandoff.net/2009/xstandoff/1.1"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsf="http://www.xstandoff.net/2009/xstandoff/1.1"
 xsi:schemaLocation="http
  <xsf:primaryData start=
   <primaryDataRef uri="a
  </xsf:primaryData>
  <xsf:segmentation>
    <xsf:segment xml:id="seg1" type="char" start="0" end="302"/>
    <xsf:segment xml:id="seg2" type="char" start="218" end="250"/>
    <xsf:segment xml:id="seg3" type="char" start="266" end="302"/>
    <xsf:segment xml:id="seg4" type="seg" segments="seg2 seg3" mode="disjoint"/>
  </xsf:segmentation>
  <xsf:annotation>
    <xsf:level xml:id="alice-log">
      <xsf:layer xmlns:log="http://www.xstandoff.net/alice/log"
       xsi:schemaLocation="http://www.xstandoff.net/alice/log alice-log.xsd">
        <log:text xsf:segment="seg1">
          <log:p xsf:segment="seg1"/>
          <log:q xsf:segment="seg4"/>
        </log:text>
      </xsf:layer>
    </xsf:level>
  </xsf:annotation>
</xsf:corpusData>
```

Note that containment is still present by the start and end positions of the respective segments

XStandoff – Containment and dominance

Version 2: p dominates the q fragments but not the entire q:

```
<xsf:corpusData xmlns="http://www.xstandoff.net/2009/xstandoff/1.1"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsf="http://www.xstandoff.net/2009/xstandoff/1.1"
 xsi:schemaLocation="http://www.xstandoff.net/2009/xstandoff/1.1 xsf.xsd">
  <xsf:primaryData start="0" end="302">
   <primaryDataRef uri="alice.txt" encoding="UTF-8"/>
  </xsf:primaryData>
  <xsf:segmentation>
    <xsf:segment xml:id="seg1" type="char" start="0" end="302"/>
    <xsf:segment xml:id="seg2" type="char" start="218" end="250"/>
    <xsf:segment xml:id="seg3" type="char" start="266" end="302"/>
    <xsf:segment xml:id="seg4" type="seg" segments="seg2 seg3" mode="disjoint"/>
  </xsf:segmentation>
  <xsf:annotation>
    <xsf:level xml:id="alice-log">
      <xsf:layer xmlns:log="http://www.xstandoff.net/alice/log"
       xsi:schemaLocation="http://www.xstandoff.net/alice/log alice-log.xsd">
        <log:text xsf:segment="seg1">
          <log:p xsf:segment="seg1">
            <log:q xsf:segment="seg2"/>
            <log:q xsf:segment="seg3"/>
          </log:p>
          <log:q xsf:segment="seg4"/>
        </log:text>
      </xsf:layer>
    </xsf:level>
  </xsf:annotation>
</xsf:corpusData>
```

XStandoff – Containment and dominance

Version 3: p dominates the entire discontinuous q (already seen):

```
<xsf:corpusData xmlns="http://www.xstandoff.net/2009/xstandoff/1.1"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsf="http://www.xstandoff.net/2009/xstandoff/1.1"
 xsi:schemaLocation="http://www.xstandoff.net/2009/xstandoff/1.1 xsf.xsd">
  <xsf:primaryData start="0" end="302">
   <primaryDataRef uri="alice.txt" encoding="UTF-8"/>
  </xsf:primaryData>
  <xsf:segmentation>
    <xsf:segment xml:id="seg1" type="char" start="0" end="302"/>
    <xsf:segment xml:id="seg2" type="char" start="218" end="250"/>
    <xsf:segment xml:id="seg3" type="char" start="266" end="302"/>
    <xsf:segment xml:id="seg4" type="seg" segments="seg2 seg3" mode="disjoint"/>
  </xsf:segmentation>
  <xsf:annotation>
    <xsf:level xml:id="alice-log">
      <xsf:layer xmlns:log="http://www.xstandoff.net/alice/log"
       xsi:schemaLocation="http://www.xstandoff.net/alice/log alice-log.xsd">
        <log:text xsf:segment="seg1">
          <log:p xsf:segment="seg1">
            <log:q xsf:segment="seg4"/>
          </log:p>
        </log:text>
      </xsf:layer>
    </xsf:level>
  </xsf:annotation>
</xsf:corpusData>
```

XStandoff – Alignment

Alignment can be used in different locations in XStandoff:

- Different serializations of a single annotation level are subsumed underneath the respective level element, e.g.:
  - The output of different syntactic parsers
  - TEI's logical document structure vs. DocBook's logical document structure
  - Annotations done by different annotators (the xsf:creator attribute shall be used in this case)
- Different corpusData entries can be related to each other via the newly introduced xsf:alt attribute – which is available at the segmentation attribute as well for annotating alternative segmentation

XStandoff – Inline representation

A new inline representation format was established as well:

- The root element is xsf:inline
- A generic xsf:milestone element (similar to the TEI's) is used for overlapping markup
- Note that the inline representation lacks validation support and was created for demonstration purposes only

```
<xsf:inline  xmlns:xsf="http://www.xstandoff.net/2009/xstandoff/1.1"
             xmlns="http://www.xstandoff.net/2009/xstandoff/1.1"
             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             xmlns:silbe="http://www.xstandoff.net/buergschaft/silbe"
             xmlns:wort="http://www.xstandoff.net/buergschaft/wort"
             xmlns:vers="http://www.xstandoff.net/buergschaft/vers">
   <silbe:text xsf:segment="seg1">
      <silbe:body xsf:segment="seg1">
         <wort:text xsf:segment="seg1">
            <wort:body xsf:segment="seg1">
               <vers:text a="b" xsf:segment="seg1">
                  <vers:body xsf:segment="seg1">
                  <!-- [...] -->
```

XStandoff – The all namespace

XStandoff provides two XML Namespaces:

- The target namespace
  http://www.xstandoff.net/2009/xstandoff/1.1/
- The new all namespace which has been introduced during the development of the inline representation
  http://www.xstandoff.net/2009/all

The all namespace allows for subsuming elements with the same local name ranging over the same text span in a single element

XStandoff – The all namespace

The inline example without and with all namespace

```
<xsf:inline xmlns:xsf="http://www.xstandoff.net/2009/xstandoff/1.1"
            xmlns="http://www.xstandoff.net/2009/xstandoff/1.1"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xmlns:silbe="http://www.xstandoff.net/buergschaft/silbe"
            xmlns:wort="http://www.xstandoff.net/buergschaft/wort"
            xmlns:vers="http://www.xstandoff.net/buergschaft/vers">
   <silbe:text xsf:segment="seg1">
      <silbe:body xsf:segment="seg1">
         <wort:text xsf:segment="seg1">
            <wort:body xsf:segment="seg1">
               <vers:text a="b" xsf:segment="seg1">
                  <vers:body xsf:segment="seg1">
                  <!-- [...] -->
```

```
<xsf:inline xmlns:xsf="http://www.xstandoff.net/2009/xstandoff/1.1"
            xmlns="http://www.xstandoff.net/2009/xstandoff/1.1"
            xmlns:all="http://www.xstandoff.net/2009/all"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xmlns:silbe="http://www.xstandoff.net/buergschaft/silbe"
            xmlns:wort="http://www.xstandoff.net/buergschaft/wort"
            xmlns:vers="http://www.xstandoff.net/buergschaft/vers">
   <all:text xsf:segment="seg1" vers:a="b">
      <all:body xsf:segment="seg1">
      <!-- [...] -->
```

XStandoff – The all namespace

The inline example without and with all namespace

```
<xsf:inline xmlns:xsf="http://www.xstandoff.net/2009/xstandoff/1.1"
            xmlns="http://www.xstandoff.net/2009/xstandoff/1.1"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xmlns:silbe="
            xmlns:wort="h
            xmlns:vers="h
   <silbe:text xsf:segmen
      <silbe:body xsf:seg
         <wort:text xsf:s
            <wort:body xs
               <vers:text
                  <vers:b
                  <!-- [...] -->
```

Note that the a attribute derived from the vers namespace stays intact.
The all layer makes no statement about the semantic value of an element – only about its range and local name!

```
<xsf:inline xmlns:xsf="http://www.xstandoff.net/2009/xstandoff/1.1"
            xmlns="http://www.xstandoff.net/2009/xstandoff/1.1"
            xmlns:all="http://www.xstandoff.net/2009/all"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xmlns:silbe="http://www.xstandoff.net/buergschaft/silbe"
            xmlns:wort="http://www.xstandoff.net/buergschaft/wort"
            xmlns:vers="http://www.xstandoff.net/buergschaft/vers">
   <all:text xsf:segment="seg1" vers:a="b">
      <all:body xsf:segment="seg1">
      <!-- [...] -->
```

So...
...where do all these XStandoff files come from?

Nobody wants to annotate XStandoff instances manually, really!

The XStandoff Toolkit

The XStandoff Toolkit – a set of four XSLT 2.0 stylesheets – supports the generation and editing of XStandoff:

- Conversion of a single inline annotation to XStandoff
- Merging of XStandoff annotation layers
- Removing of single layers/levels from an XStandoff file
- Conversion of an XStandoff instance to an XStandoff inline annotation

The XStandoff Toolkit – Inline2XSF.xsl

Converting inline annotation to XStandoff layer

- Input file for transformation: XML instance containing inline annotation
- Obligatory stylesheet parameter: primary-data → path to file containing plain textual content for referencing
- Several additional optional parameters
- Use of Saxon extensions:
  - saxon:assign → decrease of processing effort (avoid unmanageable recursiveness)
  - saxon:eval, saxon:expression → use of XPath expressions in stylesheet parameters

The XStandoff Toolkit – Inline2XSF.xsl

Two steps of converting an inline annotation to XStandoff:

1. Build segments on the basis of occurring elements
    - Mapping by primary data file's character stream
    - Comparison of the content of the primary data file and the textual content of the input file (pd identity)
2. Return layers on the basis of occurring namespace
    - For every namespace the corresponding elements are released from the initial inline annotation and copied into the layer – hierarchies remain intact
    - Elements in the XStandoff layer are linked to the according segments by ID/IDREF binding

The XStandoff Toolkit – Inline2XSF.xsl

Additional parameters

- Stylesheet parameter virtual-root avoids problems with meta information in XML annotation (textual content not present in primary data).
  virtual-root="//*[local-name()='text']"
- In the same manner meta data can be copied into the resulting XStandoff instance.
  meta-root="//*[local-name()='teiHeader']"

```
<TEI>
  <teiHeader>
    <fileDesc>
      <titleStmt>
        <title type="main">Jede Menge blinkende Lichter</title>
      </titleStmt>
      <publicationStmt>
         <publisher>Leonardo</publisher>
      </publicationStmt>
    </fileDesc>
  </teiHeader>
  <text>
    <body>
      <p>Jede Menge blinkende Lichter</p>
```

The XStandoff Toolkit – Inline2XSF.xsl

Limitations:

- We tried different algorithms for checking pd identity – some were quite loose others are rigorous
- Converting mixed content elements is still sometimes awkward due to filtering indentation/virtual whitespace

The XStandoff Toolkit – Inline2XSF.xsl

Limitations:

- We tried different algorithms for checking pd identity – some were quite loose others are rigorous
- Converting mixed content elements is still sometimes awkward due to filtering indentation/virtual whitespace

```
Sample text morphemes
These are some morphemes.
```

```
<text>
  <title>Sample text</title>
  morphemes
  <morpheme>These</morpheme>
  <morpheme>are</morpheme>
  <morpheme>some</morpheme>
  <morpheme>morpheme</morpheme>
  <morpheme>s</morpheme>.
</text>
```

```
<text><title>Sample text</title>morphemes
<morpheme>These</morpheme><morpheme>are</morpheme><morpheme>some</morpheme>
<morpheme>morpheme</morpheme><morpheme>s</morpheme>.</text>
```

The XStandoff Toolkit – Inline2XSF.xsl

Limitations:

- We tried different algorithms for checking pd identity – some were quite loose others are rigorous
- Converting mixed content elements is still sometimes awkward due to filtering indentation/virtual whitespace

```
Sample text morphemes
These are some morphemes.
```

```
<text>
  <title>Sample text</title>
  morphemes
  <morpheme>These</morpheme>
  <morpheme>are</morpheme>
  <morpheme>some</morpheme>
  <morpheme>morpheme</morpheme>
  <morpheme>s</morpheme>.
```

```
<text>
  <title>Sampletext</title>
  morphemes
  <morpheme>These</morpheme>
  <morpheme>are</morpheme>
  <morpheme>some</morpheme>
  <morpheme>morpheme</morpheme>
  <morpheme>s</morpheme>.
</text>
```

```
<text><title>Sample text</title>morpheme s
<morpheme>These</morpheme> <morpheme>are</morpheme> <morpheme>some</morpheme>
<morpheme>morpheme</morpheme><morpheme>s</morpheme>.</text>
```

```
<morpheme>These</morpheme><morpheme>are</morpheme><morpheme>some</morpheme>
<morpheme>morpheme</morpheme><morpheme>s</morpheme>.</text>
```

Merging XStandoff instances – mergeXSF.xsl

Transforming two XStandoff instances into a single one

- First XStandoff file provided as the input file, second file's name to be included via the stylesheet parameter merge-with
- Main problem: adapt segments from XStandoff files
  - There will be segments spanning over the same string of the primary data, but having distinct IDs → combine
  - There will be segments with the same ID, but spanning over different character positions → generate new unique Ids
  - Reorganization of segment list making it necessary to update the segment references of the elements in layers
- Optional output of the all-layer

Merging XStandoff instances – mergeXSF.xsl

Limitations:

- Support only for merging two single XStandoff files at once
- Instantiating the all-layer:
  - Decide which element embeds each other
  - Strategies based on statistical measurements and priority attribute not always return desired result
  - Mixed strategy to be examined in future work

Deleting parts of an XStandoff instance – removeXSFcontent.xsl

Allows for the removal of selected elements of an XStandoff instance, e.g. a level or a layer

- Supply ID of element node to be removed during transformation call via the remove-ID stylesheet parameter
- During removal the list of segments is updated
- Removed content can be returned as a separate XStandoff instance (including both the segments referenced and the extracted content)

Deleting parts of an XStandoff instance – removeXSFcontent.xsl

```
<xsf:corpusData>
  <xsf:primaryData start="0" end="23">
    <xsf:textualContent>A star shines brighter.</xsf:textualContent>
  </xsf:primaryData>
  <xsf:segmentation>
    <xsf:segment xml:id="seg1" type="char" start="0" end="23"/>
    <xsf:segment xml:id="seg2" type="char" start="0" end="1"/>
    <!-- [...] -->
  </xsf:segmentation>
  <xsf:annotation>
    <xsl:level xml:id="log">
      <xsf:layer xmlns:l="http://www.xstandoff.net/log">
        <l:text xsf:segment="seg1">
          <l:p xsf:segment="seg1">
            <l:s xsf:segment="seg1"/>
          </l:p>
        </l:text>
      </xsf:layer>
    </xsf:level>
    <xsf:level xml:id="morphemes">
      <xsf:layer xmlns:m=" http://www.xstandoff.net/morphemes">
        <m:morphemes xsf:segment="seg1">
          <m:m xsf:segment="seg2"/>
          <m:m xsf:segment="seg3"/>
          <m:m xsf:segment="seg4"/>
          <m:m xsf:segment="seg5"/>
          <m:m xsf:segment="seg6"/>
          <m:m xsf:segment="seg7"/>
        </m:morphemes>
      </xsf:layer>
    </xsf:level>
  </xsf:annotation>
</xsf:corpusData>
```

Deleting parts of an XStandoff instance – removeXSFcontent.xsl

```
<xsf:corpusData>
  <xsf:primaryData start="0" end="23">
    <xsf:textualContent>A star shines brighter.</xsf:textualContent>
  </xsf:primaryData>
  <xsf:segmentation>
    <xsf:segment xml:id="seg1" type="char" start="0" end="23"/>
    <xsf:segment xml:id="seg2" type="char" start="0" end="1"/>
    <!-- [...] -->
```

```
# saxon example.xml removeXSFcontent.xsl remove-ID=log return-removed=1
```

```
    <xsl:level xml:id="log">
      <xsf:layer xmlns:l="http://www.xstandoff.net/log">
        <l:text xsf:segment="seg1">
          <l:p xsf:segment="seg1">
            <l:s xsf:segment="seg1"/>
          </l:p>
        </l:text>
      </xsf:layer>
    </xsf:level>
    <xsf:level xml:id="morphemes">
      <xsf:layer xmlns:m=" http://www.xstandoff.net/morphemes">
        <m:morphemes xsf:segment="seg1">
          <m:m xsf:segment="seg2"/>
          <m:m xsf:segment="seg3"/>
          <m:m xsf:segment="seg4"/>
          <m:m xsf:segment="seg5"/>
          <m:m xsf:segment="seg6"/>
          <m:m xsf:segment="seg7"/>
        </m:morphemes>
      </xsf:layer>
    </xsf:level>
  </xsf:annotation>
</xsf:corpusData>
```

Deleting parts of an XStandoff instance – removeXSFcontent.xsl

```
<xsf:corpusData>
  <xsf:primaryData start="0" end="23">
    <xsf:textualContent>A star shines brighter.</xsf:textualContent>
  </xsf:primaryData>
  <xsf:segmentation>
    <xsf:segment xml:id="seg1" type="char" start="0" end="23"/>
    <xsf:segment xml:id="seg2" type="char" start="0" end="1"/>
    <!-- [...] -->
```

```
# saxon example.xml removeXSFcontent.xsl remove-ID=log return-removed=1
```

```
    <xsl:level xml:id="log">
      <xsf:layer xmlns:l="http://www.xstandoff.net/log">
        <l:text xsf:segment="seg1">
          <l:p xsf:segment="seg1">
            <l:s xsf:segment="seg1"/>
          </l:p>
        </l:text>
      </xsf:layer>
    </xsf:level>
    <xsf:level xml:id="morphemes">
      <xsf:layer xmlns:m=" http://www.xstandoff.net/morphemes">
        <m:morphemes xsf:segment="seg1">
          <m:m xsf:segment="seg2"/>
          <m:m xsf:segment="seg3"/>
          <m:m xsf:segment="seg4"/>
          <m:m xsf:segment="seg5"/>
          <m:m xsf:segment="seg6"/>
          <m:m xsf:segment="seg7"/>
        </m:morphemes>
      </xsf:layer>
    </xsf:level>
  </xsf:annotation>
</xsf:corpusData>
```

Deleting parts of an XStandoff instance – removeXSFcontent.xsl

```
<xsf:corpusData>
  <xsf:primaryData start="0" end="23">
    <xsf:textualContent>A star shines brighter.</xsf:textualContent>
  </xsf:primaryData>
  <xsf:segmentation>
    <xsf:segment xml:id="seg1" type="char" start="0" end="23"/>
    <xsf:segment xml:id="seg2" type="char" start="0" end="1"/>
    <!-- [...] -->
```

```
# saxon example.xml removeXSFcontent.xsl remove-ID=log return-removed=1
```

```
    <xsf:level xml:id="morphemes">
      <xsf:layer xmlns:m="http://www.xstandoff.net/morphemes">
        <m:morphemes xsf:segment="seg1">
          <m:m xsf:segment="seg2"/>
          <m:m xsf:segment="seg3"/>
          <m:m xsf:segment="seg4"/>
          <m:m xsf:segment="seg5"/>
          <m:m xsf:segment="seg6"/>
          <m:m xsf:segment="seg7"/>
        </m:morphemes>
      </xsf:layer>
    </xsf:level>
  </xsf:annotation>
</xsf:corpusData>
```

Deleting parts of an XStandoff instance – removeXSFcontent.xsl

```
<xsf:corpusData>
  <xsf:primaryData start="0" end="23">
    <xsf:textualContent>A star shines brighter.</xsf:textualContent>
  </xsf:primaryData>
  <xsf:segmentation>
    <xsf:segment xml:id="seg1" type="char" start="0" end="23"/>
    <xsf:segment xml:id="seg2" type="char" start="0" end="1"/>
    <!-- [...] -->
```

```
# saxon example.xml removeXSFcontent.xsl remove-ID=log return-removed=1
```

```
    <xsf:level xml:id="morphemes">
      <xsf:layer xmlns:m="http://www.xstandoff.net/morphemes">
        <m:morphemes xsf:segment="seg1">
          <m:m xsf:se
          <m:m xsf:se
          <m:m xsf:se
          <m:m xsf:se
          <m:m xsf:se
          <m:m xsf:se
        </m:morphemes
      </xsf:layer>
    </xsf:level>
  </xsf:annotation>
</xsf:corpusData>
```

```
<xsf:corpusData>
  <xsf:primaryData start="0" end="23">
    <xsf:textualContent>A star shines brighter.</xsf:textualContent>
  </xsf:primaryData>
  <xsf:segmentation>
    <xsf:segment xml:id="seg1" start="0" end="23"/>
  </xsf:segmentation>
  <xsf:annotation>
    <xsl:level xml:id="log">
      <xsf:layer xmlns:l="http://www.xstandoff.net/logical">
        <l:text xsf:segment="seg1">
          <l:p xsf:segment="seg1">
            <l:s xsf:segment="seg"/>
          </l:p>
        </l:text>
      </xsf:layer>
    </xsf:level>
  </xsf:annotation>
</xsf:corpusData>
```

Building inline XStandoff annotations: XSF2inline.xsl

Counterpart of inline2XSF.xsl – creates inline annotation on the basis of an XStandoff instance

- Covers the handling of overlapping markup by milestone elements
  - Detection of segments whose start and end position information constitute an overlap
  - Segments are split up into segments representing milestones being adequate basis to build an inline annotation
  - Linear list of segments is processed recursively by taking the currently 'outermost' segments (those who are not included in other segments' spans)
  - Copy respective elements into inline annotation
  - Recursive processing of segments which are embedded in the outermost segments

Building inline XStandoff annotations: XSF2inline.xsl

Necessary mechanism regarding the possibility of elements from different layers referencing the same segment

- Decision for the order in which elements should be nested into each other can be influenced by the optional stylesheet parameter sort-by (default value 'measure': statistical analysis on the basis of frequency measures)
- Inadequate for elements from different layers for which no definite statistical result for embedding can be achieved
- There could be elements whose boundaries always share the same character positions, but which can clearly semantically be assigned to a certain embedding
- Embedding can be based on priority attribute of the level (in SGF) or the layer (in XStandoff) element by specifying the value 'priority' for the parameter sort-by
- Underlies assumption that there can be a semantically grounded, definite decision for the embedding
- Future work will combine statistical and priority method

Analyzing XStandoff instances

An XQuery script is available as well as starting point for analyzing relations between elements derived from different layers:

- Virtual parentship (aka containment)
- Inclusion
- Equal start or end point
- Overlaps

Conclusion

To sum up:

- We've presented XStandoff – the successor of the Sekimo Generic Format
- XStandoff is XML-based, uses the Annotation Graph model and is a logical descendant to the Prolog fact base format
- XStandoff combines inline and standoff annotation and allows for validating multiple annotated data
- The XStandoff Toolkit contains stylesheets and scripts allowing for creating, editing and manipulating XStandoff instances
- Future work will concentrate on enhancing the XStandoff Toolkit
- Both, the XStandoff XML schema files and the XStandoff Toolkit are available online under the LGPL at `http://www.xstandoff.net`

Last but not least...

Thank your for your attention!

Visit us at `http://www.xstandoff.net`

`{maik.stuehrenberg|daniel.jettka}@uni-bielefeld.de`