Visualization of concurrent markup
From trees to graphs, from 2D to 3D

Daniel Jettka
Maik Stührenberg

Bielefeld University

Introduction

Limits of XML-based markup languages apparent:

- traditionally belief in underlying tree model

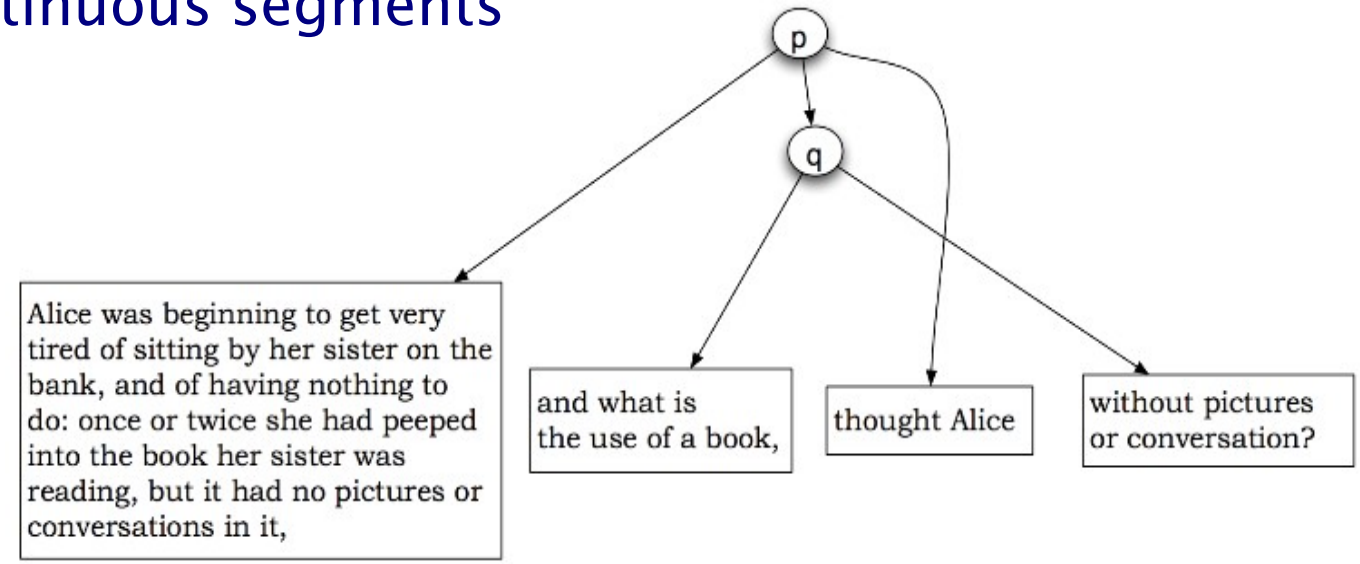
  → cannot cope with multiple and possibly overlapping structures
- other phenomena (e.g. discontinuous elements, repetitive structures) require advanced graph model (not only minor tree extensions)

Agenda:

- underlying data model of XML can be seen as a graph, not a tree
- possible to serialize graph-like structures including discontinuous elements with plain XML
- two XML-based representation formats as basis for visualization efforts
- 2D & 3D visualization methods for concurrent markup

A formal model of XML instances

- text as ordered hierarchy of content objects (OHCO thesis)
  - many people think markup languages use the formal model of a tree
- tree model: pure perspective on nesting of elements (and attributes)
  - often not possible to represent concurrent hierarchies ("overlap is multi-parentage")
  - crossing arcs not allowed → not possible to annotate discontinuous segments



(Sperberg-McQueen and Huitfeldt, 2008, GODDAG representing "Alice in Wonderland" example)

XML: underlying graph model

- hierarchical structure of element nesting implies tree model

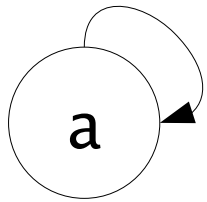
  → well-formed XML instances

- XML-inherit integrity constraints (DTD: ID, IDREF, IDREFS token type attributes; XSD: xs:ID, xs:IDREF, xs:IDREFS, xs:key, xs:keyref) allow for description of graph structures

  → valid XML instances

⇒ graphs (e.g. concurrent hierarchies + challenging structures) can be described by valid XML

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE a
[ <!ELEMENT a EMPTY>
  <!ATTLIST a id ID #IMPLIED idref IDREF #IMPLIED> ]>
<a id="a" idref="a"/>
```



Present non-XML solutions

- 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

→ not native XML (no use of present XML tools & specifications)

Present XML solutions

- XML-based formats
  - TEI milestones & fragments: relations between elements implicit (dominance relations may get lost)
  - NITE (serialization of Annotation Graphs) → focus on multimodal data/time axis; complex creation
  - XML serializations of LMNL: CLIX, ECLIX, xLMNL → no hierarchical relations between annotations
  - Feature Structures (ISO standard; can be used to serialize multiple annotations → resulting instances can get huge – difficult to render)
  - EARMARK → complex RDF format/difficult to create manually
  - XStandoff: meta annotation format → difficult to create manually (toolkit written in XSLT available)

⇒ xLMNL and XStandoff as starting points for visualization

Visualization of xLMNL & XStandoff

Example: “Drive my car”

primary data

```
Asked a girl what she wanted to be
She said baby, can't you see
I wanna be famous, a star on the screen
But you can do something in between
Baby you can drive my car
Yes I'm gonna be a star
Baby you can drive my car
And baby I love you
```

verse structure annotation

```
<?xml version="1.0" encoding="UTF-8"?>
<text xmlns="http://www.tei-c.org/ns/1.0">
 <body>
  <lg type="verse">
   <l>Asked a girl what she wanted to be</l>
   <l>She said baby, can't you see</l>
   <l>I wanna be famous, a star on the screen</l>
   <l>But you can do something in between</l>
  </lg>
  <lg type="chorus">
   <l>Baby you can drive my car</l>
   <l>Yes I'm gonna be a star</l>
   <l>Baby you can drive my car</l>
   <l>And baby I love you</l>
  </lg>
 </body>
</text>
```

direct discourse annotation

```
<?xml version="1.0" encoding="UTF-8"?>
<text xmlns="http://www.tei-c.org/ns/1.0">
 <body>
  <p>Asked a girl what she wanted to be
    She said <q>baby, can't you see
    I wanna be famous, a star on the screen
    But you can do something in between</q></p>
  <p><q>Baby you can drive my car
    Yes I'm gonna be a star
    Baby you can drive my car
    And baby I love you</q></p>
 </body>
</text>
```

Starting points for visualization

two standoff formats as starting points:

xLMNL

```
<x:lmnl-document>
 <x:content>Asked a girl what she wanted to be
  She said baby, can't you see
  I wanna be famous, a star on the screen
  But you can do something in between
  Baby you can drive my car
  Yes I'm gonna be a star
  Baby you can drive my car
  And baby I love you</x:content>
 <x:range name="text" ID="text-1" start="0" end="235"/>
 <x:range name="text" ID="text-2" start="0" end="235"/>
 <x:range name="body" ID="body-1" start="0" end="235"/>
 <x:range name="body" ID="body-2" start="0" end="235"/>
 <x:range name="lg" ID="lg-1" start="0" end="140">
  <x:annotation name="type" role="start-annotation">
   <x:content>verse</x:content>
  </x:annotation>
 </x:range>
 <x:range name="p" ID="p-1" start="0" end="140"/>
 <x:range name="l" ID="l-1" start="0" end="34"/>
 <x:range name="l" ID="l-2" start="35" end="63"/>
 <x:range name="q" ID="q-1" start="44" end="139"/>
 <x:range name="l" ID="l-3" start="64" end="103"/>
 <x:range name="l" ID="l-4" start="104" end="139"/>
 <x:range name="lg" ID="lg-2" start="140" end="235">
  <x:annotation name="type" role="start-annotation">
   <x:content>chorus</x:content>
  </x:annotation>
 </x:range>
 <x:range name="p" ID="p-2" start="140" end="235"/>
 <x:range name="q" ID="q-2" start="140" end="235"/>
 <x:range name="l" ID="l-5" start="140" end="165"/>
 <x:range name="l" ID="l-6" start="166" end="189"/>
 <x:range name="l" ID="l-7" start="190" end="215"/>
 <x:range name="l" ID="l-8" start="216" end="235"/>
</x:lmnl-document>
```

XStandoff (simplified)

```
<xsf:corpusData>
 <xsf:primaryData start="0" end="235">
  <xsf:primaryDataRef uri="../pd/drive_my_car.txt"/>
 </xsf:primaryData>
 <xsf:segmentation>
  <xsf:segment xml:id="seg1" start="0" end="235"/>
  <xsf:segment xml:id="seg2" start="0" end="140"/>
  <!-- ... -->
  <xsf:segment xml:id="seg11" start="190" end="215"/>
  <xsf:segment xml:id="seg12" start="216" end="235"/>
 </xsf:segmentation>
 <xsf:annotation>
  <xsf:level xml:id="drive_my_car_lines-level1">
   <xsf:layer xmlns="http://www.tei-c.org/ns/1.0">
    <text xsf:segment="seg1">
     <body xsf:segment="seg1">
      <lg xsf:segment="seg2" type="verse">
       <!-- ... -->
      </lg>
      <lg xsf:segment="seg8" type="chorus">
       <!-- ... -->
      </lg>
     </body>
    </text>
   </xsf:layer>
  </xsf:level>
  <xsf:level xml:id="drive_my_car_quotes-level1">
   <xsf:layer xmlns="http://www.tei-c.org/ns/1.0">
    <text xsf:segment="seg1">
     <body xsf:segment="seg1">
      <p xsf:segment="seg2">
       <q xsf:segment="seg5"/>
      </p>
      <p xsf:segment="seg8">
       <q xsf:segment="seg8"/>
      </p>
     </body>
    </text>
   </xsf:layer>
  </xsf:level>
 </xsf:annotation>
</xsf:corpusData>
```

2D visualization of concurrent markup

Provide solutions for:

(a) illustration of relationship between primary data and annotations

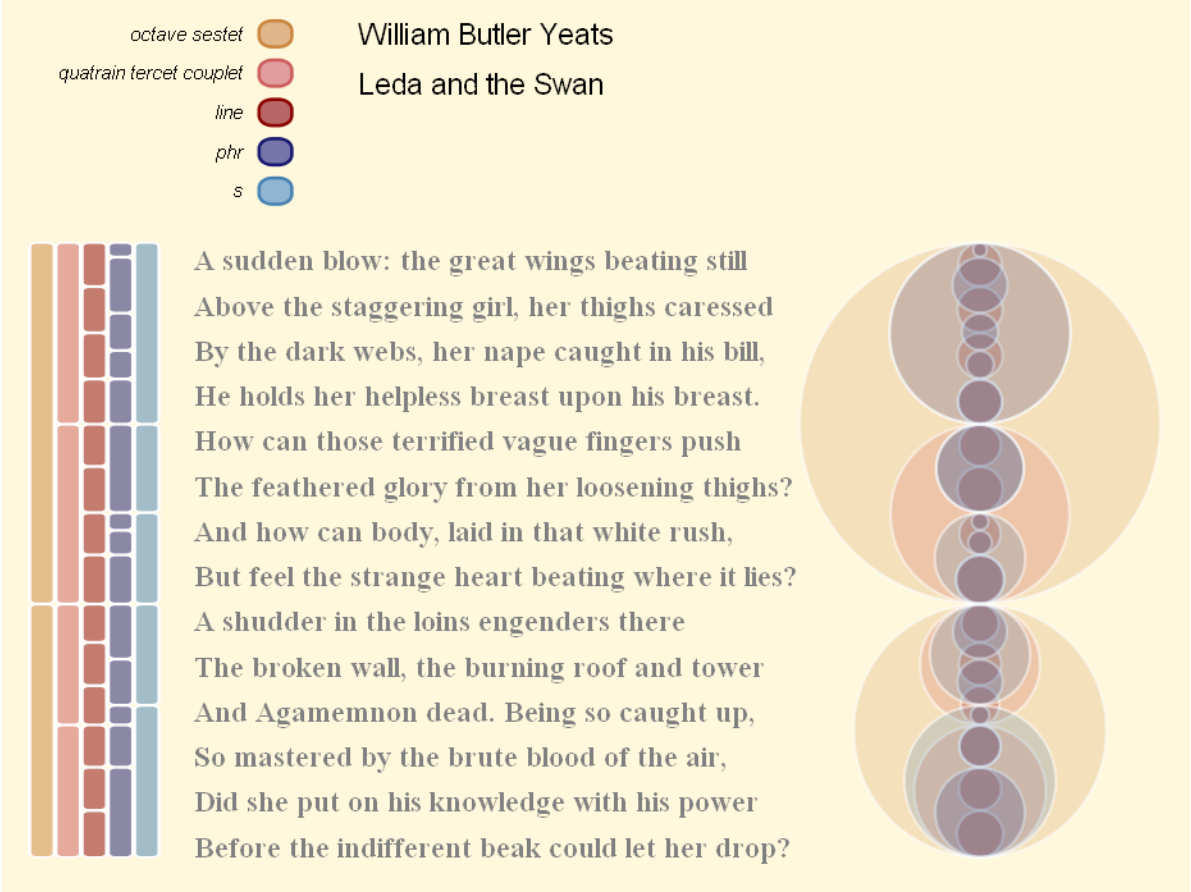
(b) illustration of relationship between annotations

→ visualization of potentially overlapping annotations and other tree-challenging phenomena

Creating SVG for xLMNL

Piez (2010):

- interactive SVG
- overlaps visualized



Problems:

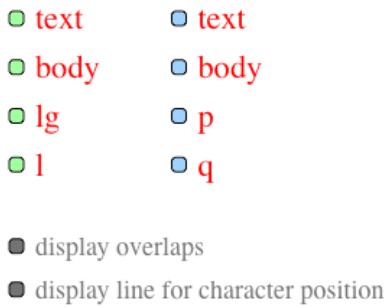
(1) create bar segments for small string ranges

(2) create circles for large string ranges

(3) less clear picture for less uniform annotations

Creating SVG for XStandoff

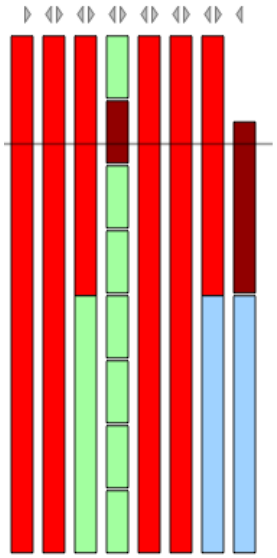
- XSLT stylesheet XSF2SVG.xsl – greatly inspired by Piez (2010)
- no circles; some additional features (e.g. switching of annotation levels & highlighting of classic overlaps)



Problems:

(1) other phenomena than overlaps & discontinuous structures, e.g. repetitive structures

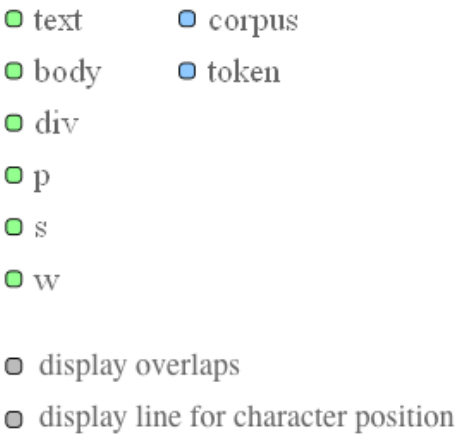
(2) create bar segments for short string ranges
→ compute chars per line

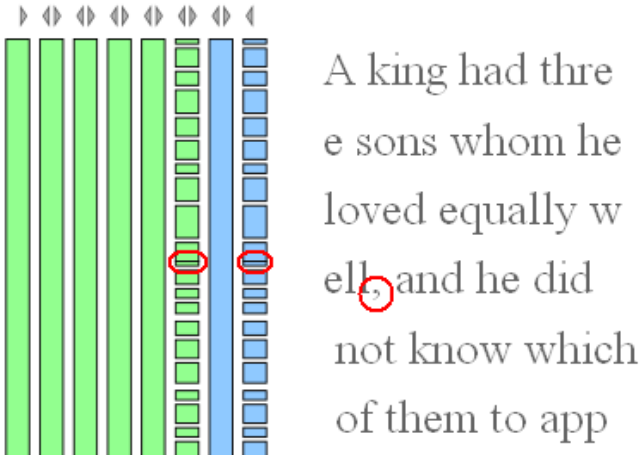


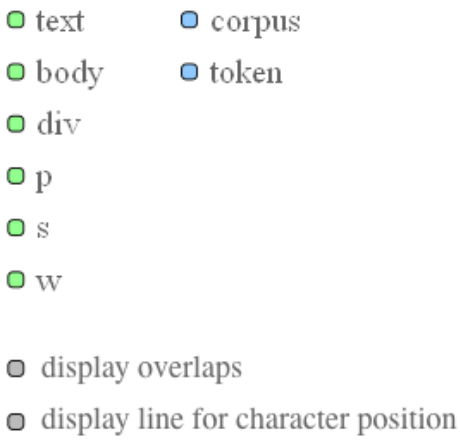
Asked a girl what she wanted to be
She said baby, can't you see
I wanna be famous, a star on the screen
But you can do something in between
Baby you can drive my car
Yes I'm gonna be a star
Baby you can drive my car
And baby I love you

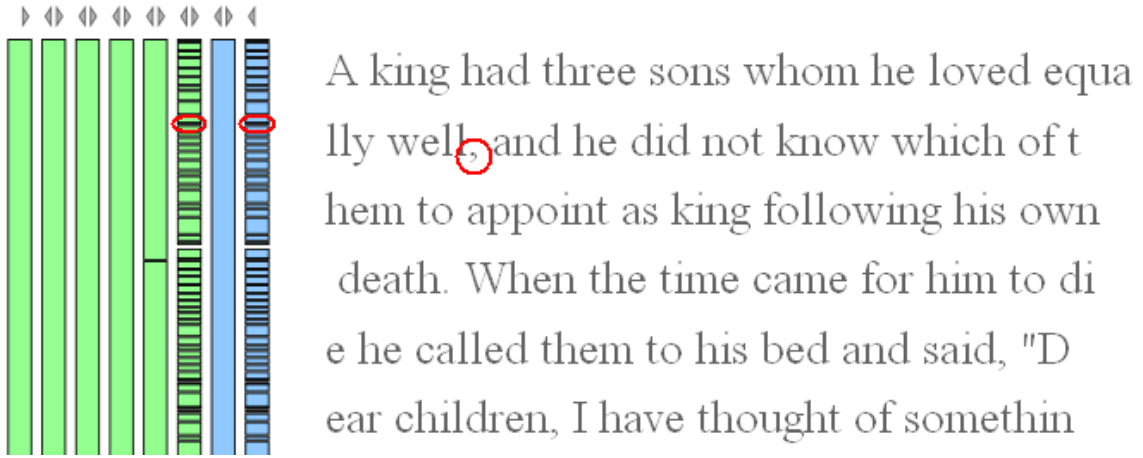
Creating SVG for XStandoff

Computation of $max-line-length to avoid segment bar problem (also adjustable manually: $max-line-length as xs:integer)









Another problem: additional line breaks

Investigation of 3D visualization method

- native browser support for 3D graphics
  - HTML5's <canvas> allows for rendering of APIs like WebGL (current builds of Firefox 5 and Chrome 12)
- 3D graphics in XML: X3DOM + serialization format X3D
- possible to implement transformation scenarios for XML-based representation formats without leaving the XML context

- native browser support of XSLT 2.0 would make the framework even more straightforward (same for SVG approach)

  → very promising alternative: Saxon-CE (not tested yet)

Basic principle

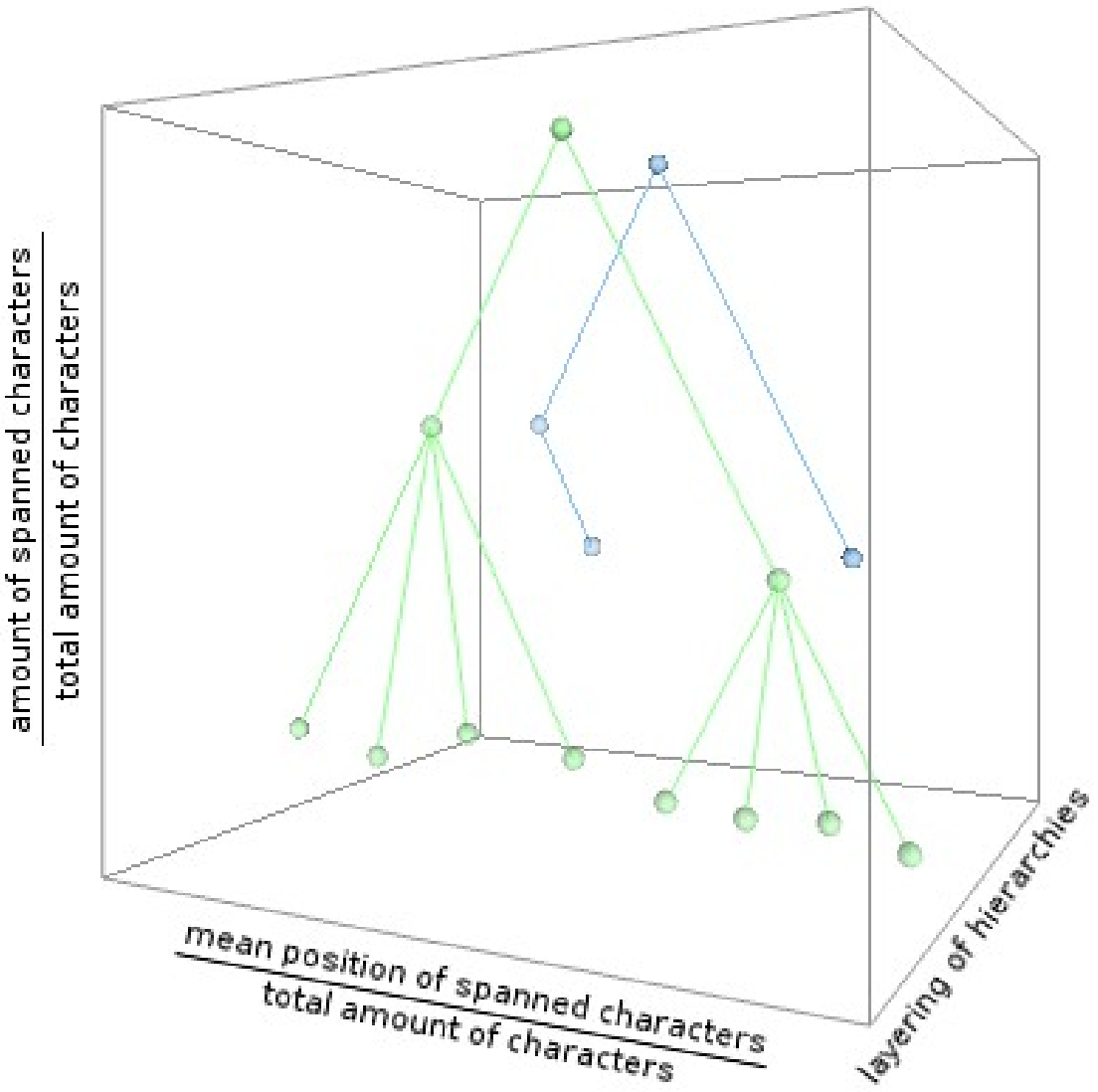
- new perspectives: possible to order concurrent hierarchies (trees, extended trees, graphs) along the z-axis of 3D space
- construction of comparable layers
  - normalization of illustrations
  - horizontal normalization:
    - horizontal position of nodes representing annotations
    - positioning nodes on the basis of their spanned character string
  - vertical normalization:
    - divide amount of spanned characters by the total amount of characters in PD to determine vertical position of nodes

→ possible to allow for different realizations of layer visualizations (containment vs. dominance perspective)

Positioning of annotation layers

Visualization of two XStandoff layers (ordering along z-axis)

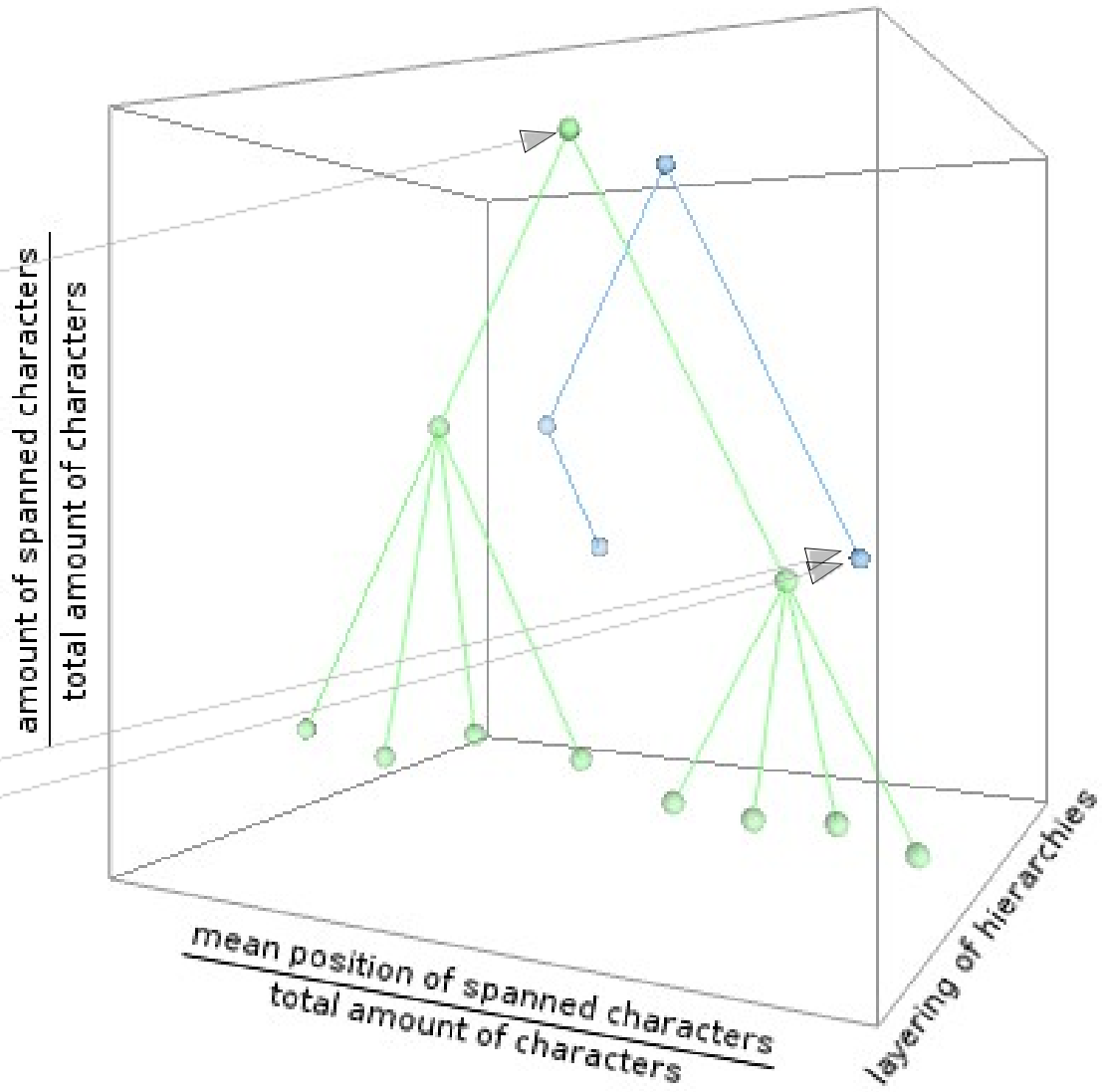
```
<xsf:corpusData>
 <xsf:primaryData start="0" end="235">
  <xsf:primaryDataRef uri="../pd/drive_my_car.txt"/>
 </xsf:primaryData>
 <xsf:segmentation>
  <xsf:segment xml:id="seg1" start="0" end="235"/>
  <xsf:segment xml:id="seg2" start="0" end="140"/>
  <!-- ... -->
  <xsf:segment xml:id="seg11" start="190" end="215"/>
  <xsf:segment xml:id="seg12" start="216" end="235"/>
 </xsf:segmentation>
 <xsf:annotation>
  <xsf:level xml:id="drive_my_car_lines-level1">
   <xsf:layer xmlns="http://www.tei-c.org/ns/1.0”>
    <text xsf:segment="seg1">
     <body xsf:segment="seg1">
      <lg xsf:segment="seg2" type="verse">
       <!-- ... -->
      </lg>
      <lg xsf:segment="seg8" type="chorus">
       <!-- ... -->
      </lg>
     </body>
    </text>
   </xsf:layer>
  </xsf:level>
  <xsf:level xml:id="drive_my_car_quotes-level1">
   <xsf:layer xmlns="http://www.tei-c.org/ns/1.0">
    <text xsf:segment="seg1">
     <body xsf:segment="seg1">
      <p xsf:segment="seg2">
       <q xsf:segment="seg5"/>
      </p>
      <p xsf:segment="seg8">
       <q xsf:segment="seg8"/>
      </p>
     </body>
    </text>
   </xsf:layer>
  </xsf:level>
 </xsf:annotation>
</xsf:corpusData>
```



Normalized positioning of nodes in 3D space

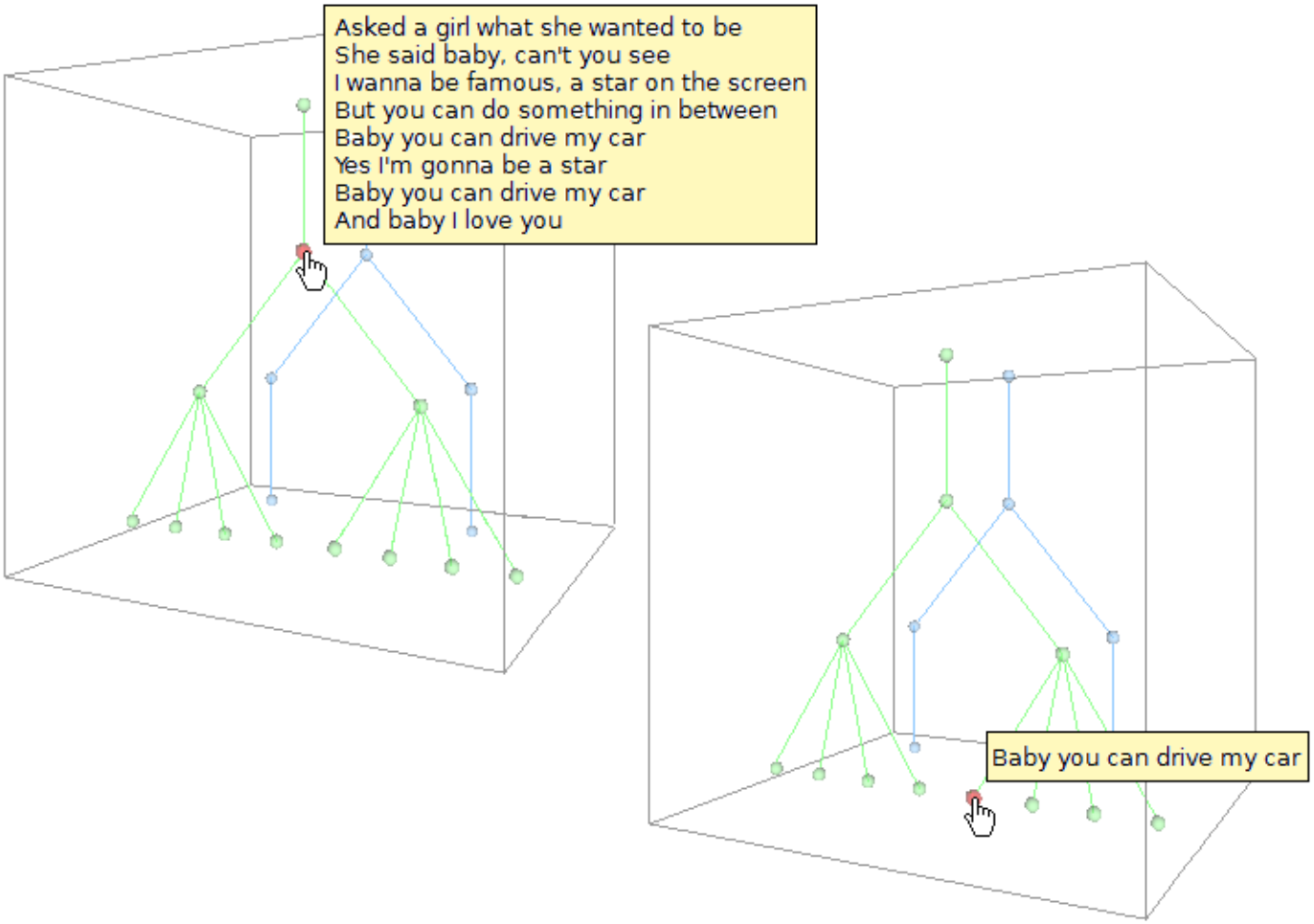
Visualization of two XStandoff layers (due to normalization n:1 relation between elements and nodes)

```
<xsf:corpusData>
 <xsf:primaryData start="0" end="235">
  <xsf:primaryDataRef uri="../pd/drive_my_car.txt"/>
 </xsf:primaryData>
 <xsf:segmentation>
  <xsf:segment xml:id="seg1" start="0" end="235"/>
  <xsf:segment xml:id="seg2" start="0" end="140"/>
  <!-- ... -->
  <xsf:segment xml:id="seg11" start="190" end="215"/>
  <xsf:segment xml:id="seg12" start="216" end="235"/>
 </xsf:segmentation>
 <xsf:annotation>
  <xsf:level xml:id="drive_my_car_lines-level1">
   <xsf:layer xmlns="http://www.tei-c.org/ns/1.0">
    <text xsf:segment="seg1">
     <body xsf:segment="seg1">
      <lg xsf:segment="seg2" type="verse">
       <!-- ... -->
      </lg>
      <lg xsf:segment="seg8" type="chorus">
       <!-- ... -->
      </lg>
     </body>
    </text>
   </xsf:layer>
  </xsf:level>
  <xsf:level xml:id="drive_my_car_quotes-level1">
   <xsf:layer xmlns="http://www.tei-c.org/ns/1.0">
    <text xsf:segment="seg1">
     <body xsf:segment="seg1">
      <p xsf:segment="seg2">
       <q xsf:segment="seg5"/>
      </p>
      <p xsf:segment="seg8">
       <q xsf:segment="seg8"/>
      </p>
     </body>
    </text>
   </xsf:layer>
  </xsf:level>
 </xsf:annotation>
</xsf:corpusData>
```

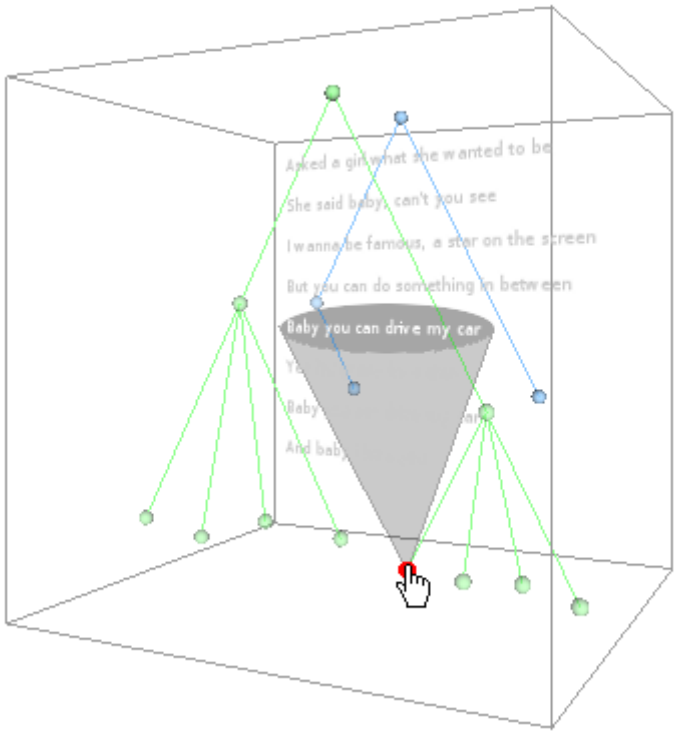


Relation between annotations and primary data

- different approaches imaginable
- could vary from rather simple illustrations to quite complex ones



(dominance perspective → 1:1 relation between elements and nodes)



(containment perspective → n:1 relation between elements and nodes)

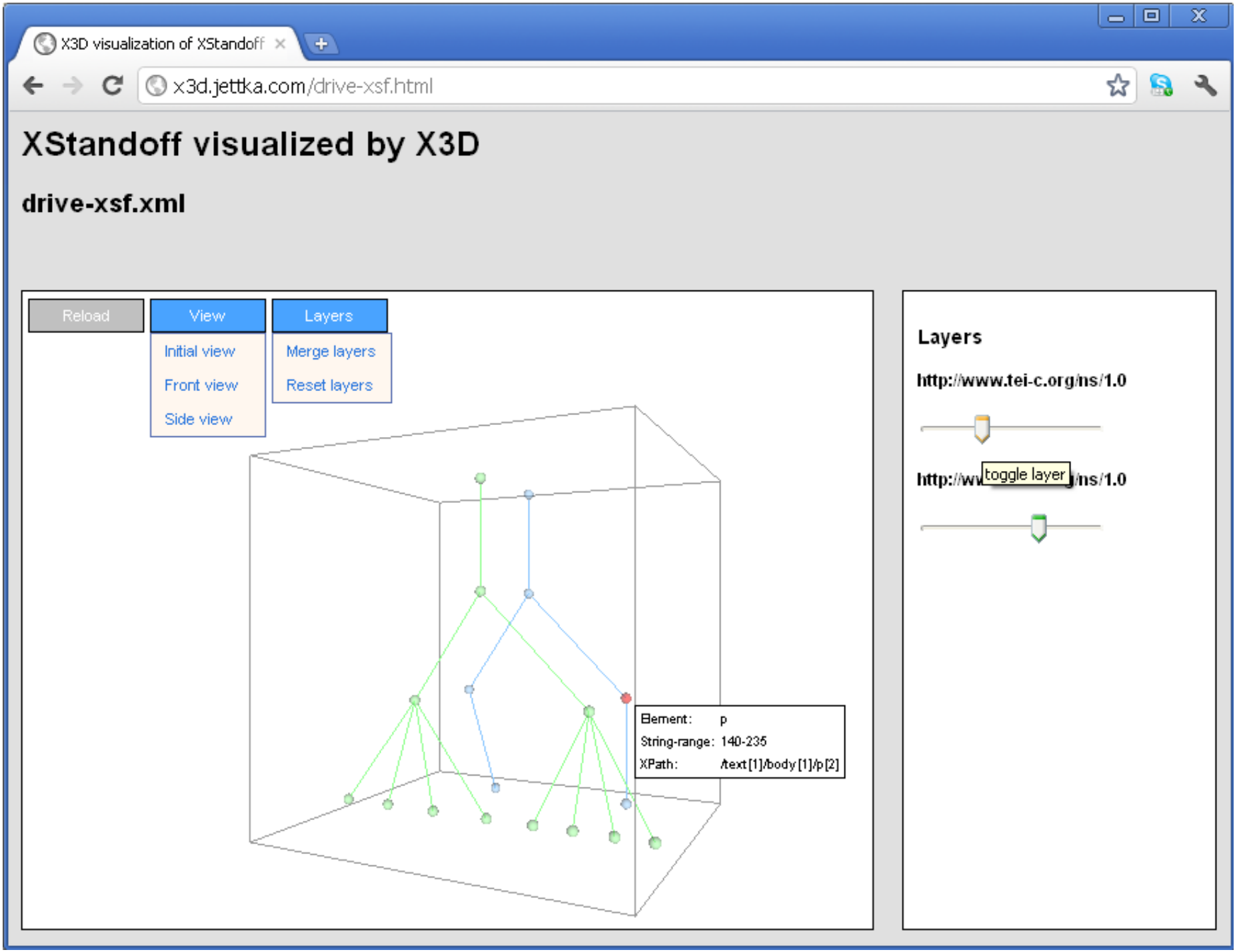
Core features of a 3D approach

- free user navigation
- draggable structures for layers (e.g. draggable as a whole along the z-axis)
- mouse-over effects for information on spanned primary data (textual content & positions), information on annotation, XPath, etc.
- highlighting of specific structures (distinct element relations, overlaps, discontinuous elements, virtual/repetitive structures)
- choice between displaying annotated or plain textual content for a node
- illustration of left and right context of focussed annotation elements and corresponding textual content (+ specification of the range of considered context)

3D prototype

- prototypic 3D visualization implemented

  → XSLT stylesheet XSF2X3D.xsl transforms XStandoff instances into X3D graphics
- other formats could be converted to XStandoff (e.g. xLMNL → only containment relations can be visualized)
- embedding of X3D into HTML5 → rendering of 3D visualizations in current Mozilla Firefox & Google Chrome versions
- some HTML5 components only available in Chrome (<input type=”range”>)



(a) Dominance relations (see above)

(b) Containment relations

Conclusion & future research

- XML's formal model capable of representing graphs
- xLMNL & XStandoff can be converted into 2D visualizations
  - → adequate (though admittedly suboptimal) solution to overlapping structures
  - → not capable of illustrating enhanced graph-based phenomena like discontinuous elements or repetitive structures
- possible 3D renderings of concurrent markup
  - → prototypic realization demonstrated how adding of an additional dimension could in principle contribute to the appropriate visualization of concurrent markup
  - → current version will be made available under the GNU Lesser General Public License (LGPL v3) at http://www.xstandoff.net
  - → unresolved tasks (e.g. improved visualization of overlapping annotations, treatment of discontinuous and repetitive structures) to be tackled in future release

Thank you for your attention!

http://www.xstandoff.net

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

Bibliography

Cowan, J. (2010). MicroXML. Poster presented at XML Prague 2010.

Cowan, J. (2011). MicroXML. Editor's Draft 2011-06-30. http://www.ccil.org/~cowan/MicroXML.html

Kay, M. (2011). XSLT in the Browser. In Kosek, J. (ed). *XML Prague 2011 Conference Proceedings*, number 2011-519 in ITI Series, pp. 125-134, Prague, Czech Republic, Institute for Theoretical Computer Science.

Piez, W. (2010). Towards Hermeneutic Markup: An architectural outline. In *Digital Humanities 2010.* Conference Abstract, London.

Sperberg-McQueen, C. M. & Huitfeldt, C. (2008). Markup Discontinued Discontinuity in TexMecs, Goddag structures, and rabbit/duck grammars. In: *Proceedings of Balisage: The Markup Conference 2008. Balisage Series on Markup Technologies*, vol. 1 (2008).

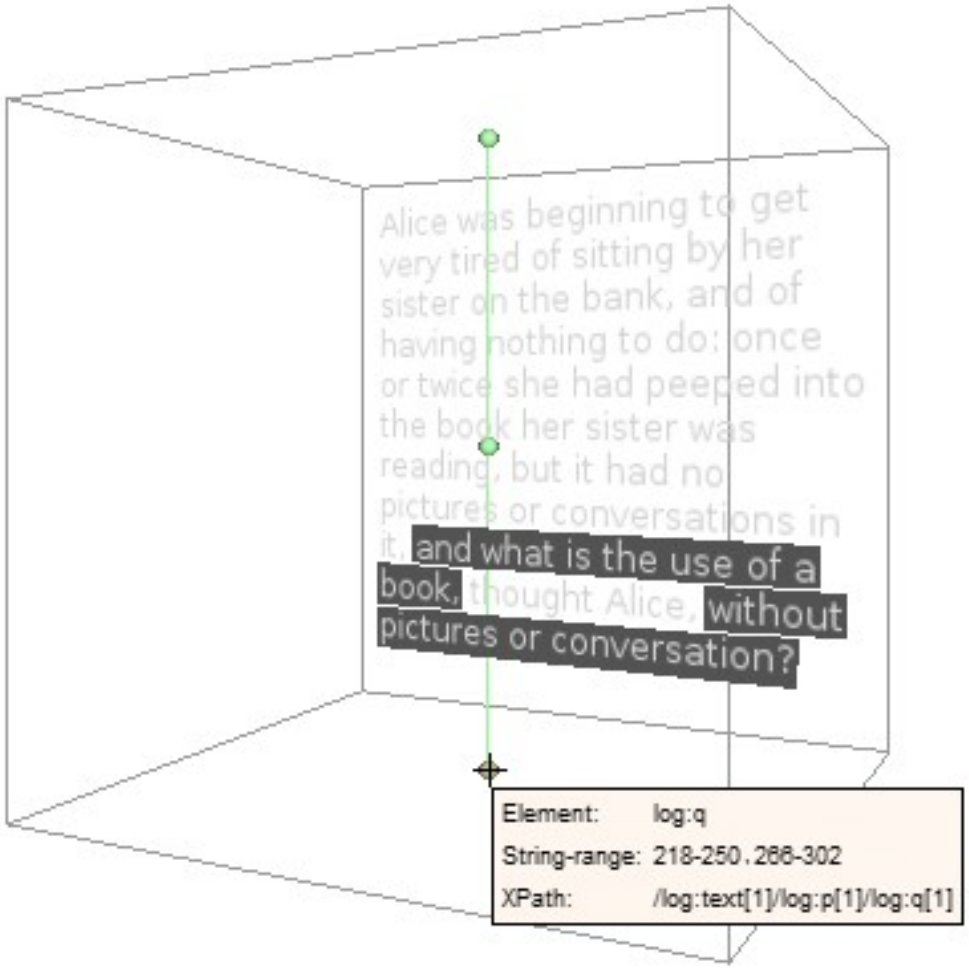
Witt, A., Goecke, D., Sasaki, F. & Lüngen, H. (2005). Unification of XML Documents with Concurrent Markup. In *Literary and Linguistic Computing*, 20(1): 103-116.

Some additional material

Discontinuous elements

Reconsidering the Alice example

```
<xsf:corpusData>
 <xsf:primaryData start="0" end="302">
  <xsf:textualContent>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, "and what is the use
of a book," thought Alice, "without pictures or
conversations?"</xsf:textualContent>
 </xsf:primaryData>
 <xsf:segmentation>
  <xsf:segment xml:id="seg1" start="0" end="302"/>
  <xsf:segment xml:id="seg2" start="218" end="250"/>
  <xsf:segment xml:id="seg3" 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"
     priority="0">
    <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>
```



Repeating structures

How to handle repeating structures?

ID/IDREF relationship
(cf. start of presentation)

Add label to indicate number
of repetitions