Just-In-Time-Trees
Recognizing Markup on Demand

University of Kentucky

Patrick Durusau

8 October 2003

Overview

- Why overlapping hierarchies?
- Prior approaches
- Bottom-Up-Virtual-Hierarchies
- Recognizing Markup
- Just-In-Time-Trees
- Future Research

Why overlapping hierarchies?

- Different interpretations of a text
- Structures that do not “nest” properly
- Complex textual traditions with multiple witnesses and variants
- Recording physical layout of text and other analysis
- Versioning

Overlapping Example

Matthew 3:8	Bear fruit that befits repentance,
Matthew 3:9	and do not presume to say to yourselves, ‘We have Abraham as our father’; for I tell you, God is able from these stones to raise up children of Abraham.

Matthew 3:8-9 First Choice

```
<verse id=“Matt.3.8”>
```

Bear fruit that befits repentance,

```
</verse>
<verse=“Matt.3.9”>
```

and do not presume to say to yourselves, ‘We have Abraham as our father’; for I tell you, God is able from these stones to raise up children of Abraham.

```
</verse>
```

Matthew 3:8-9: Second Choice

<sentence>

Bear fruit that befits repentance, and do not presume to say to yourselves, ‘We have Abraham as our father’; for I tell you, God is able from these stones to raise up children of Abraham

</sentence>

Matthew 3:8-9 Verboten!

```
<verse id=“Matt.3.8”>
      <sentence>
      Bear fruit that befits repentance,
</verse>
<verse=“Matt.3.9”>
      and do not presume to say to yourselves, ‘We have
   Abraham as our father’; for I tell you, God is able from
   these stones to raise up children of Abraham.
</verse>
      </sentence>
```

Other Examples:

- Open a textbook or journal
  - Paragraph crosses page boundary
  - Quote crosses a paragraph
  - Footnote crosses a page boundary
  - Highlighting begins in one sentence and ends in another
- All of these require overlapping markup

Prior Approaches

- CONCUR (cf Sema Group)
- Fragmentation (virtual joins)
- Milestones
- Standoff Markup
- Non-SGML/XML markup (Tex-Mecs, LMNL)
- Bottom-Up-Virtual-Hierarchies

Concur

```
<(vh)verse id="Matt.3.8">
     <(sh)sentence>
     Bear fruit that befits repentance,
</(vh)verse>
<(vh)verse="Matt.3.9">
     and do not presume to say to yourselves, 'We have
  Abraham as our father'; for I tell you, God is able from
  these stones to raise up children of Abraham.
</(vh)verse>
     </(sh)sentence>
```

Fragmentation

```
<verse id="Matt.3.8">
     <sentence id="Matt.pt1">
     Bear fruit that befits repentance, </sentence>
</verse>
<verse="Matt.3.9"> <sentence id="Matt.pt2">
        and do not presume to say to yourselves, 'We have
   Abraham as our father'; for I tell you, God is able from
   these stones to raise up children of Abraham.
   </sentence>
</verse>
```

Elsewhere <join targets="Matt.pt1 Matt.pt2" result="sentence" />

Milestones

```
<verse id="Matt.3.8">
      <ss id="s1"/>
      Bear fruit that befits repentance,
</verse>
<verse="Matt.3.9">
      and do not presume to say to yourselves, 'We have
   Abraham as our father'; for I tell you, God is able from
   these stones to raise up children of Abraham.
</verse>
      <se corresp="s1"/>
```

Standoff Markup

```
<linkPoint verse id="Matt.3.8">
    <linkPoint sentence>
```

Bear fruit that befits repentance,

```
 <linkPoint /verse>
 <linkPoint verse="Matt.3.9">
```

and do not presume to say to yourselves, 'We have Abraham as our father'; for I tell you, God is able from these stones to raise up children of Abraham.

```
 < linkPoint /verse>
    <linkPoint /sentence>
```

Non-SGML/XML Syntaxes

- Tex-MECS: Wittengenstein Project
- LMNL: Tennison and Piez
- Both develop non-SGML/XML syntaxes that currently lack processor support.
- LMNL syntax is based on core range algrebra, which allows layering of ranges of text, one upon another.

Bottom-Up Virtual Hierarchies

- Observations:
  - Membership of PCDATA in a hierarchy
  - Membership in multiple hierarchies
- Question: How to represent in standard XML?
- Atomic PCDATA (word division)
- Base file with XML Membership XPath expression for each hieraracy

Bottom-Up Virtual Hierarchies II

- Sound verbose?

```
<w id="w4"
    sn:clauses="/clauses/clause[1][@id='c1']/a[1]/*[1]"
    tx:text="/text/para[1][@id='p1']/*[4]"
    pg:pages="/pages/page[2][@id='p2']/line[1][@id='l1']/*[1]"
>in</w>

<w id="w5"
    sn:clauses="/clauses/clause[1][@id='c1']/a[1]/*[2]"
    tx:text="/text/para[1][@id='p1']/*[5]"
    pg:pages="/pages/page[2][@id='p2']/line[1][@id='l1']/*[2]"
    vr:variants="/variants/app[2][@id='tv2']/rdg[1][@wit='C'][@val='an']"
>a</w>
```

Bottom-Up Virtual Hierarchies III

- Represents all possible hierarchies
- Allows querying across hierarchies
- But:
  - Fixed (like traditional markup)
  - Fragile (like standoff markup)
  - Non-standard syntax
  - Requires pre-parsing of data
  - Verbose

Lessons of BUVH

- Markup is metadata about PCDATA
  - Membership of PCDATA in hierarchies
  - PCDATA should be primary, markup secondary
- Markup is asserted/recognized during processing
  - Not fixed at time of entry, but upon demand from the processor

Recognizing Markup

- What composes a markup tree?
  - Elements/PCDATA
- How to declare a markup tree?
  - DTD or schema
- But, what is a markup tree really?
  - A language, based on a meta-language
- And how are languages defined?

Defining A Language

- Standard Language Definition
  - A language L is a set of strings over an alphabet
  - SGML/XML parsers require:
    - Language predefined as <, </
    - Tokens must nest into a tree
    - Only defining tokens, not the language
    - Markup vs. PCDATA distinction fixed
  - Results in monolingual parsers

Multilingual Parsers

- What is needed?
  - Definition of lexical level
  - Definition of parsing level
    - (separately)
- In XML
  - Lexical and parsing defined together
  - DTD/Schema defines a particular lexical vocabulary
  - Parsing is predefined
  - XML parser != JITTs parser

Building a JITTs Parser

- JITTs parser requires
  - Definition of lexical level
  - Definition of parsing rules
    - (separately)
- SGML/XML documents
  - No changes required
  - Enhanced use of existing documents

Building a JITTs Parser II

- Don’t build from scratch!
  - Island Grammars!
- Specific productions match constructs of interest, “islands”
- General productions match the “water” around the island
- Separates the lexical from parsing

JITTs Advantages I

- Tree based Access : SAX-like speed
  - DOM-Lite (less memory footprint)
  - Recognize the tree as far or as shallow as desired
  - Allows a tree based interface to the document, while preserving lower level markup
  - When container retrieved, lower level markup recognized for presentation

JITTs Advantages II

- Partial validation
  - Recognize only markup of interest
  - Useful for partial validation of offshore data entry or markup
  - Avoids validation of entire file for proofing of particular errors

Reasons for JITTs

- Single tree view of texts

  Vs.
- Multi-tree view of texts
- Dom-Lite
- Unlimited by current parser models
- Consider your Texts
  - Simple tree or Complex tree
- Question is: Which do you prefer?
- Better Question: Which fits your texts?

JITTs Pitfalls

- Watch out for trees!
  - Naive top-down parse may be confused by recursive elements
  - Ex: text/div/p/q
  - Will become confused at:
  - <text><div><p>….<q>Then it is agree, <q>all debts are paid in full</q> by the signing of this document.</q>….</p></div></text>
  - Problem with tree based syntax.

Future Research

- But what of descriptive markup?
- Is it limited by the tree model?
- Where elements share a common start or end point?
- Where elements share both a common start and end point?
- Traditional syntax requires container relationship

Conclusion

- JITTs parsing offers advantages over current SGML/XML parsers
- Frees descriptive markup from its tree ancestry
- Frees document authors from crude work arounds to make their texts match an imaginary model

Island Grammar References

- Generating Robust Parsers using Island Grammars, Moonen, http://www.cwi.nl/~leon/papers/wcre2001/wcre20
- Lightweight Impact Analysis using Island Grammars, Moonen, http://citeseer.nj.nec.com/moonen02lightweight.ht
- Disambiguation Filters for Scannerless Generalized LR Parsers, Visser, www.cs.uu.nl/people/visser/ftp/BSVV02.pdf

Support for Research

This research has been conducted in collaboration with Matthew Brook O'Donnell and supported by the Society of Biblical Literature http://www.sbl-site.org and OpenText.org http://www.opentext.org .