Restoring the Primacy of PCDATA

Patrick Durusau
(Society of Biblical Literature)

Matthew Brook O'Donnell
(Publishing Dimensions & OpenText.org)
Previously on jitts.org...

It was observed that:

#1. Markup is METADATA about PCDATA

#2. The recognition of what is and is not MARKUP—and from that the assertion of 'structure'—takes place during PROCESSING and not encoding (JITTs)
Corollary to Obs. #1

Markup is METADATA about PCDATA

- PCDATA should be primary and its metadata (the markup) secondary
- XML in its syntax, and particularly in the common processing methods, inverts this relationship—the metadata 'contains' or becomes the parent of the data itself.
Explorations from Obs. #2

Markup is asserted during PROCESSING

- Multiple trees can be recognized in encoding and a particular tree selected during processing (JITTs paradigm)

- It is not necessary to assert a tree at all!...

Set Theory offers alternative model
A Property of Simple Set Theory

- Objects that compose sets are treated as members of each set as they occur and we apply set operations to those members (and not the sets themselves).

- Enumerating a set always gives the atomic objects.
A Property of Simple Set Theory

A = \{a,b,c,d,e,f\}
B = \{d,e\}
C = \{f\}

A = \{a,b,c,B,C\}

Durusau & O'Donnell
Restoring the Primacy of PCDATA
(j5)

XML Europe, May 7, 2003
**Compare DOM representation**

```xml
<A>
  a b c
  <B>
    d e
  </B>
  <C>
    f
  </C>
</A>
```

```
A
  /\    /
 /   \  /   \\
B   C  a b c
     /\   |
    /  \ /  \\
   d e f
```
In Set Theory...

- Set members may be members of multiple sets (ie. They are not constrained to be a member of only one set)

- Sets are created by *operations*:
  - union
  - intersection
  - complement
In Markup Theory...

- SGML/XML has promoted a 'hierarchical container' view of markup

- Questions as to how markup reflects the nature of 'text' have focused on:
  a) the hierarchical nature of text
  b) the container-like nature of its components

- OHCO thesis (Renear et al. 1995)
In Markup Theory...

<b>Overlap</b> <i>is</i> a <i>problem</i>!

or

<overlap>happens</overlap>
Using Set Theory...

- Hierarchy and containment are dynamic and optional
- Overlap is supported
- Set members can 'belong' to multiple sets that are unrelated (in terms of parent-child and sibling relationships)
An Example...

The assertion is that **PCDATA** should be **primary**.

*Naturally:*

- `<p>`The assertion is that `<b>PCDATA</b> `<i>`should be`</i>` `<u>`primary`</u>`.</p>`

*In XML:*

- `<p>`The assertion is that `<b>PCDATA</b> `<i>`<b>should be</b>` `<u>`primary`</u>`.</p>`
The assertion is that PCDATA should be primary.

5 PCDATA nodes on 3 levels, extra b node required to avoid overlap.

PCDATA on 1 level, metadata overlaid.
Some observations...

- **Under tree model:**
  - PCDATA is governed by metadata providing convenient method of recording membership
  - Metadata has a single structure
  - Range of membership limited to containing ancestors

- **Under set model:**
  - PCDATA is independent of metadata, membership must be recorded independently
  - Metadata may have multiple structures
  - No restrictions on membership
**BUVH: Back around again...**

- **Bottom-Up Virtual Hierarchies (BUVH) – Durusau & O’Donnell (2001)**
  - From initial observation of markup as metadata re. PCDATA
  - PCDATA atoms (tokenization of PCDATA)
  - Full XPath membership of each PCDATA atom in each hierarchy marked on every atom

```xml
<w id="w3"
   sn:clauses="/clauses/clause[1][@id='c1']/c[1]/atom()[1]"
   tx:text="/text/para[1][@id='p1']/atom()[3]"
   pg:pages="/pages/page[1][@id='p1']/line[2][@id='l2']/atom()[1]"
   vr:variants="/variants/app[1][@id='tv1']/rdg[1][@wit='A'] [@val='texs']"
   >text</w>
```
BUVH: Back around again...

- Designed for multiple independent hierarchies (OHCO)
  - e.g. page/line v. section/sentence analysis of text
- Highly verbose (redundant)
- Querying depends on string functions
- In most cases overlap is LOCAL and therefore not best treated as separate hierarchy
- BUT... fundamental insight of attaching memberships at PCDATA atom level is sound
1. Input document (not well-formed)

```
<p>The assertion is that <b>PCDATA</b> <i>should be</i> primary.</p>
```
Processing Example...

1. Input document (not well-formed)
2. JITTs filter
   • Recognise all markup
   • Output milestones

The assertion is that PCDATA should be primary.
1. Input document (not well-formed)
2. JITTs filter
   - Recognise all markup
   - Output milestones

```xml
<?xml version="1.0"?>
<root>
  <p_start/>
  The assertion is that
  <b_start/>PCDATA
  <i_start/>should be
  <b_end/>
  <u_start/>primary
  <i_end/>
  <u_end/>
  .
  <p_end/>
</root>
```
The assertion is that **PCDATA** should be *primary*.

```xml
<?xml version="1.0"?>
<root>
  <p_start/>
The assertion is that
  <b_start/>PCDATA
    <i_start/>should be
  <b_end/>
  <u_start/>primary
    <i_end/>
  <u_end/>
. <p_end/>
</root>
```
The assertion is that PCDATA should be primary.
The assertion is that PCDATA should be primary.
The assertion is that PCDATA should be primary.
The assertion is that PCDATA should be primary.
The assertion is that **PCDATA** should **be** **primary**.

**Events:**
- character(‘The assertion is that’)

**Active sets:**
- p
The assertion is that PCDATA should be primary.
The assertion is that PCDATA should be primary.

Events:

character('The assertion is that')

Active sets:

p
The assertion is that PCDATA should be primary.

### Events:

- character('The assertion is that')

### Active sets:

- p
The assertion is that PCDATA should be primary.
The assertion is that **PCDATA** should be **primary**.
The assertion is that \textbf{PCDATA} should be \textit{primary}.

\textbf{Events:}

\begin{itemize}
  \item character(‘PCDATA’)
\end{itemize}

\textbf{Active sets:}

\begin{itemize}
  \item p, b
\end{itemize}
The assertion is that **PCDATA** should be **primary**.

**Active sets:**

p, b

**Events:**

- character(‘PCDATA’)

<?xml version="1.0"?>
<root>
  <p_start/>The assertion is that
    <b_start/>PCDATA
      <i_start/>should be
       <b_end/>
      <u_start/>primary
       <i_end/>
      <u_end/>
    .<p_end/>
</root>
The assertion is that PCDATA should be primary.

**Active sets:**

- p
- b
- i

**Events:**

- startElement('i_start')
The assertion is that PCDATA should be primary.
The assertion is that **PCDATA should be** primary.

Active sets: p, b, i

Events:

character('should be')

```xml
<?xml version="1.0"?>
<root>
  <p_start/>
The assertion is that
  <b_start/>PCDATA
  <i_start/>should be
  <b_end/>
  <u_start/>primary
    <i_end/>
  <u_end/>
  .<p_end/>
</root>
```
The assertion is that **PCDATA** should **be** primary.

```xml
<?xml version="1.0"?>
<root>
  <p_start/>The assertion is that
    <b_start/>PCDATA
    <i_start/>should
    <b_end/>
    <i_end/>primary
    <u_end/>
  <u_end/>
  .<p_end/>
</root>
```

Active sets:

```
p, b, i
```

Events:

```
character('should be')
```
The assertion is that **PCDATA** should be primary.
The assertion is that PCDATA should be primary.

Events:
startElement('b_end')

Active sets:
p, i
The assertion is that PCDATA should be primary.
The assertion is that **PCDATA** should be primary.
The assertion is that **PCDATA** should be **primary**.
The assertion is that PCDATA should be primary.
The assertion is that `<b>PCDATA</b>` should be `<u>primary</u>`.

**Active sets:**
- `p`, `i`, `u`
The assertion is that `PCDATA` should be primary.
The assertion is that PCDATA should be primary.
The assertion is that **PCDATA** should be **primary**.
The assertion is that PCDATA should be primary.
The assertion is that **PCDATA** should be primary.
The assertion is that **PCDATA** should be **primary**.
The assertion is that **PCDATA** should be **primary**.
Set-based queries...

- Simple XPath expressions for presence of attributes

```
<?xml version="1.0"?>
<root>
  <w p="1">The</w>
  <w p="1">assertion</w>
  <w p="1">is</w>
  <w p="1">that</w>
  <w p="1" b="1">PCDATA</w>
  <w p="1" b="1" i="1">should</w>
  <w p="1" b="1" i="1">be</w>
  <w p="1" i="1" u="1">primary</w>
</root>
```
Set-based queries...

- Simple XPath expressions for presence of attributes
  
  - All bold-italic:
    
    \[
    \text{primary}\left(\text{b} \cap \text{i}\right)
    \]

```xml
<?xml version="1.0"?>
<root>
  <w p="1">The</w>
  <w p="1">assertion</w>
  <w p="1">is</w>
  <w p="1">that</w>
  <w p="1" b="1">PCDATA</w>
  <w p="1" b="1" i="1">should</w>
  <w p="1" b="1" i="1">be</w>
  <w p="1" i="1" u="1">primary</w>
</root>
```
Set-based queries...

- Simple XPath expressions for presence of attributes
  - All bold-italic:
    \[//w[@b][@i] \quad b \cap i\]
  - non-italic bold:
    \[//w[@b][not (@i)]\]

<?xml version="1.0"?>
<root>
  <w p="1">The</w>
  <w p="1">assertion</w>
  <w p="1">is</w>
  <w p="1">that</w>
  <w p="1" b="1">PCDATA</w>
  <w p="1" b="1" i="1">should</w>
  <w p="1" b="1" i="1">be</w>
  <w p="1" i="1" u="1">primary</w>
</root>
A more complex example...

‘When Jesus saw their faith, he spoke to the paralyzed man...’ (Mk. 2.5)

Literal rendering of Greek word order:

‘and seeing Jesus their faith spoke to the paralytic’
A more complex example...

```xml
<clause>
  <conj>and</conj>
  <adjunct>
    <clause>
      <predicate>seeing</predicate>
      <subject>Jesus</subject>
      <complement>their faith</complement>
    </clause>
  </adjunct>
  <predicate>spoke</predicate>
  <complement>to the paralytic</complement>
</clause>
```
A more complex example...

```
and seeing Jesus their faith spoke to the paralytic
```
A more complex example...

```
and seeing Jesus their faith spoke to the paralytic
```

- Clause
  - Conjunction
    - Clause
      - Predicate
        - Subject
        - Complement
<clause>
  <conj>and</conj>
  <adjunct id="a1">
    <clause id="c1">
      <predicate>seeing</predicate>
    </clause>
  </adjunct>
  <subject>Jesus</subject>
  <adjunct ref="a1">
    <clause ref="c1">
      <complement>their faith</complement>
    </clause>
  </adjunct>
  <predicate>spoke</predicate>
  <complement>to the paralytic</complement>
</clause>
The Set Representation...

Jesus

subject

conjunction

and

adjunct

clause

subj. 2

compl. 2

seeing

their faith

predicate

spoke

to the paralytic

complement

Durusau & O'Donnell

Restoring the Primacy of PCDATA
The Set Representation...

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

<clause>
  <w conjunction="1">and</w>
  <w adjunct="1" clause="2" predicate="2">seeing</w>
  <w subject="1">Jesus</w>
  <w adjunct="1" clause="2" complement="2">their faith</w>
  <w predicate="1">spoke</w>
  <w complement="1">to</w>
  <w complement="1">the</w>
  <w complement="1">paralytic</w>
</clause>
```
Using Syntactical Containment...

- By default interpret markup at processing time as:
  - Elements = set boundaries
  - PCDATA = set members
  - Contained elements = subsets

```xml
<div>
  <p>Example for SET model.</p>
  <p>Hierarchical syntax interpreted as set definitions.</p>
</div>
```
Using Syntactical Containment...

- Membership of contained PCDATA and elements assumed unless exemption stated

<x>Non-contiguous
<y notSubsetOf="x">set memberships?</y>
No problem!
</x>
Using Syntactical Containment...

- Hierarchies are used to address places where set interpretation is imposed upon markup
  - don’t need to transform to a completely flat representation
  - overlap is usually local

- Attributes on atomized elements turned into metadata empty elements
Some use cases...

- Offshore conversion services often return ill-formed XML which requires interactive correction. Using a set-based interpretation well-formed XML created which can be queried to locate problems of overlap and unclosed elements.

```xml
<p>
  <b>Missing</b> closing
  <i>italic</i> tag
</p>
```

```xml
<root>
  <w p="1" b="1">Missing</w>
  <w p="1">closing</w>
  <w p="1" i="1">italic</w>
  <w p="1">tag</w>
  <opensets i="1"/>
</root>
```
Some use cases...

- Multiple and overlapping interpretations of text
  - e.g. Page/line and sentence/segment analysis
  - chapter/verse and linguistic clause analysis

- Discontinuous phenomena

- Reordering
Conclusions...

- Just as markup is a particular view of a text, the parsing of markup should represent only one particular view of the markup (i.e. NO canonical view of the markup)

- Parsing of markup should serve the user’s needs and not an abstract definition of acceptable parsing