Accumulators in XSLT and XSpec

Developing, Debugging, Testing

Amanda Galtman

August 2023

Key Takeaways

Accumulators are not that opaque

Ad hoc visibility:

- Debugger, logging
  - A few DOs and DON'Ts
- Generate report of values

Automated testing:

XSpec designs give you options

About Me

- Learned about XML and automated testing at MathWorks (MATLAB)
- Open to consulting opportunities
  - XSpec, XSLT, XQuery, Schematron, and other technologies
- Help maintain XSpec infrastructure
- Community contributor to NIST XML-based tools
- *New:* Blog about XSpec
  - https://medium.com/@xspectacles

Background

The Basics

- Accumulator associates data with a tree
- Declaration at top level of stylesheet

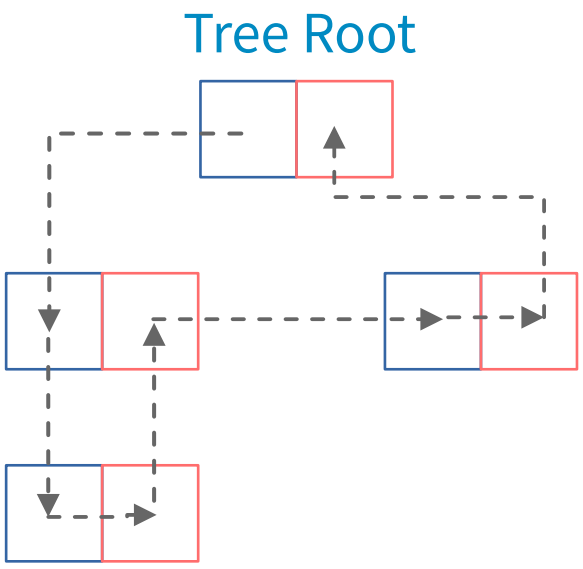
```
<xsl:stylesheet>
   <xsl:accumulator>
      <xsl:accumulator-rule> (one or more)
   </xsl:accumulator>
   …
</xsl:stylesheet>
```

Example 1: HTML Heading Level

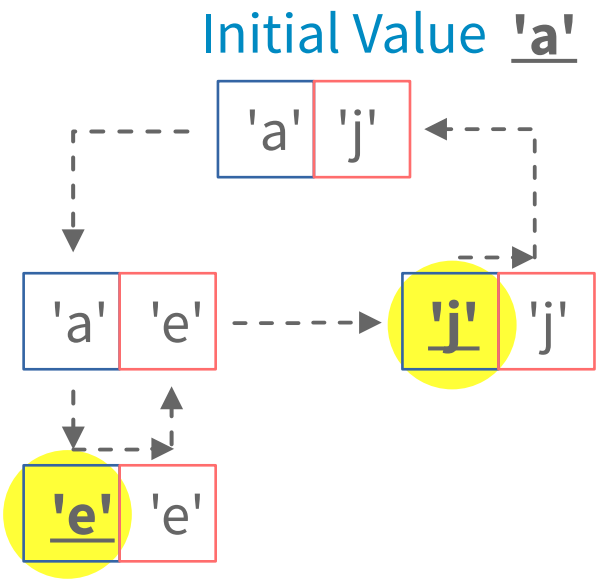
"What level heading is this paragraph under?"

```
<xsl:accumulator name="heading-depth"
as="xs:integer" initial-value="0">
   <xsl:accumulator-rule match="h1" select="1"/>
   <xsl:accumulator-rule match="h2" select="2"/>
   <xsl:accumulator-rule match="h3" select="3"/>
</xsl:accumulator>
```

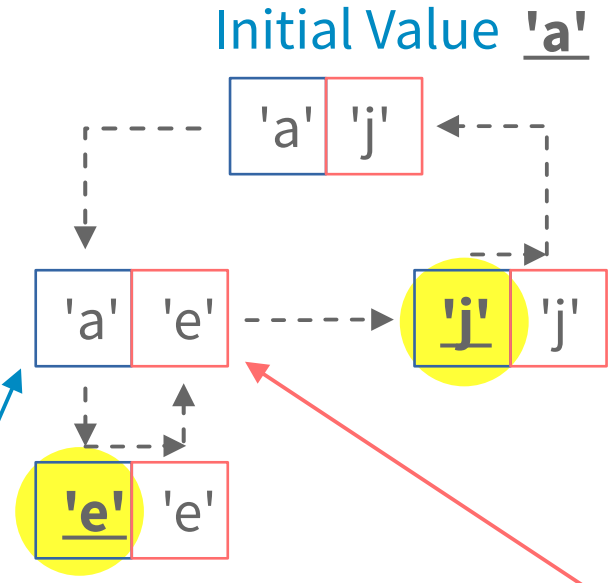
Tree Traversal



Data Associated with Tree



Data Retrieval

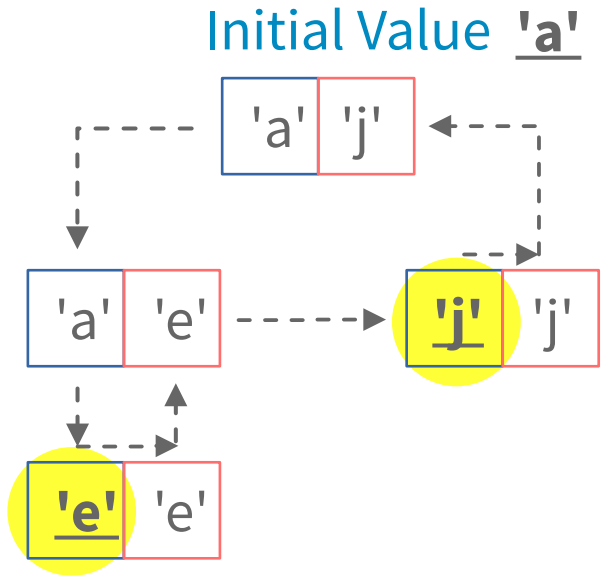


`accumulator-before(`*acc-name*`)`

`accumulator-after(`*acc-name*`)`

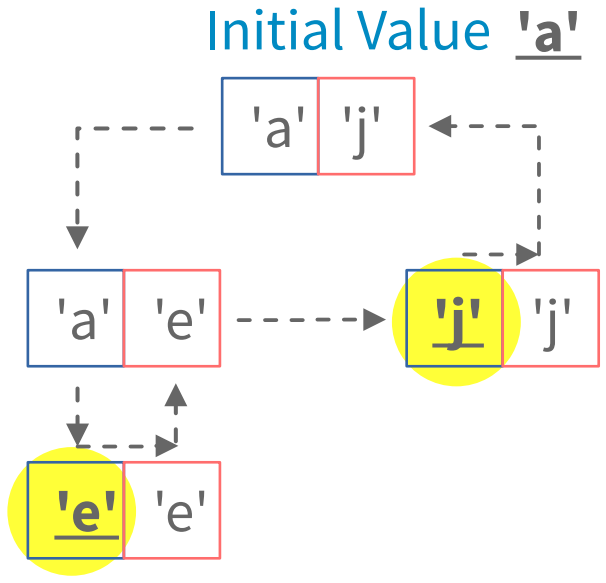
What's Interesting About This?

- Value persists until next match
  - Document-order traversal
- Accumulator rule can access prior value
  - **$value** variable
  - *Usage*: Progressive computation, e.g., running total, growing sequence



What's Interesting About This? (cont'd.)

- Rule matches are independent of retrieval sites
  - *Usage*: Rule-match one structure, retrieve based on another (possibly overlapping) structure, e.g.,
    - Count words in text nodes, retrieve at end of section
    - Record data by phrase, retrieve by line
- Two values for each node
  - Independent accumulator rules
  - *Usage*: Inverse operations, e.g.,
    - Push/pop data on stack
    - Increment/decrement depth
    - In/out of scope



Example 1: HTML Heading Level

"What level heading is this paragraph under?"

```
<xsl:accumulator name="heading-depth"
as="xs:integer" initial-value="0">
   <xsl:accumulator-rule match="h1" select="1"/>
   <xsl:accumulator-rule match="h2" select="2"/>
   <xsl:accumulator-rule match="h3" select="3"/>
</xsl:accumulator>
```

Example 2: Running Count of Headings

"How many headings have occurred so far?"

```
<xsl:accumulator name="heading-count"
as="xs:integer" initial-value="0">
   <xsl:accumulator-rule match="h1 | h2 | h3">
      <xsl:sequence select="$value + 1"/>
   </xsl:accumulator-rule>
</xsl:accumulator>
```

Use Cases

- Compute something progressively
  - Especially in stream processing
- Stack or queue
- Track state based on patterns
  - e.g., "In scope" or "Ready for release"

Not the Best Fit If...

- Computation misaligned with document-order traversal
- The data you need isn't fixed for each node+phase
- Attribute nodes require their own data
- Crossing XSLT boundary with transform()
  - But crossing tree boundary with doc() is OK

Debugging

Bug Prevention & Simple Fixes

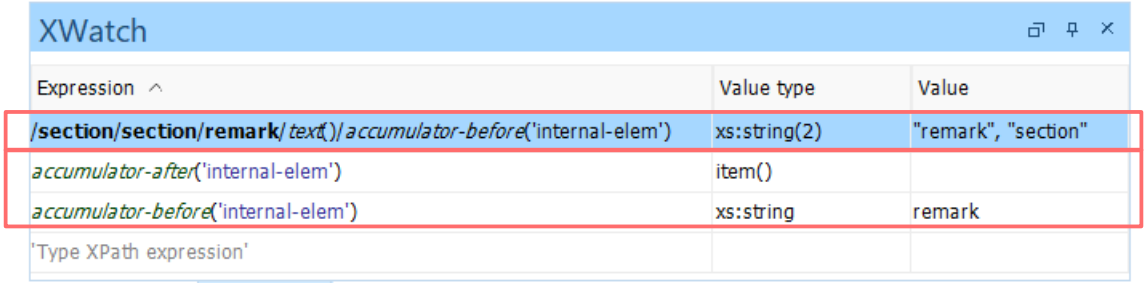
- Interactions among match patterns
  - Ambiguous rule matches
  - Nesting
- Resetting value
- `use-accumulators` attribute

Debugging Techniques

- XPath Watch (Oxygen debugger)
- Trace (Saxon)
- Report of accumulator values
- XSLT messages

XPath Watch (XWatch)

- Place breakpoint
  - Not in <xsl:accumulator>
- "XWatch" view
- Values at breakpoint's context node
  - accumulator-before(...)
  - accumulator-after(…)
- Values at arbitrary nodes
  - *//path/to/node*/accumulator-before(…)
  - *//path/to/node*/accumulator-after(...)



Trace in Saxon

- Requires Saxon PE or EE
- Set attribute `saxon:trace="yes"` in `<xsl:accumulator>`
  - Where `xmlns:saxon="http://saxon.sf.net/"`

```
internal-elem BEFORE /section/remark[1]: "remark"
internal-elem AFTER /section/remark[1]: ()
internal-elem BEFORE /section/section[1]: "section"
internal-elem BEFORE /section/section[1]/remark[1]:  ("remark", "section")
internal-elem AFTER /section/section[1]/remark[1]: "section"
internal-elem AFTER /section/section[1]: ()
```

Report of Accumulator Values

- You provide
  - (a) XML document
  - (b) XSLT stylesheet
  - (c) Accumulator name
- Transform XML document (a) using XSLT tool
  - https://github.com/galtm/xslt-accumulator-tools
  - Tool combines your XSLT code with reporting code
- Result: HTML report of accumulator values across XML document

Sample Report of Accumulator Values

Values of internal-elem Accumulator for Document section-with-internal-content.xml

Document URI: .../xslt-accumulator-tools/sample-acc/sample-xml/section-with-internal-content.xml

Accumulator declaration URI: .../xslt-accumulator-tools/sample-acc/internal-elem.xsl

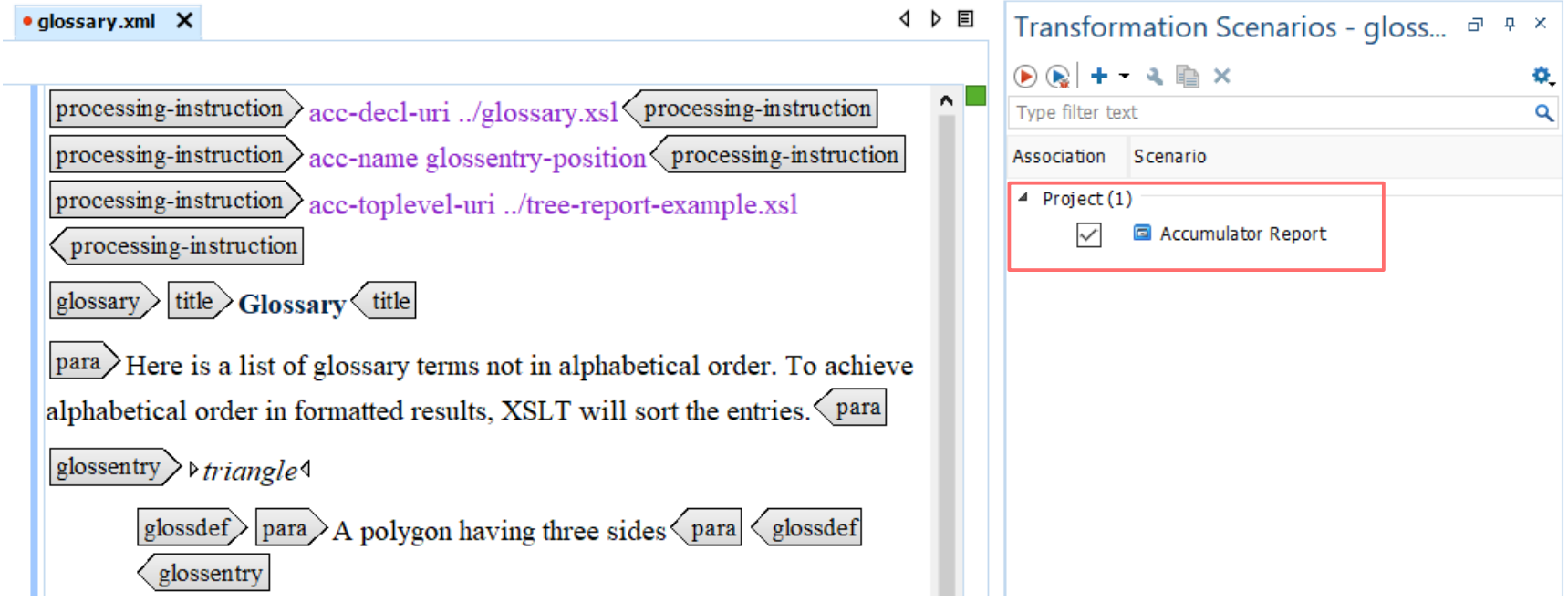
▼ Declaration

```
<xsl:accumulator xmlns:xs="http://www.w3.org/2001/XMLSchema"
                 xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                 name="internal-elem"
                 as="xs:string*"
                 initial-value="()">
   <xsl:accumulator-rule phase="start" match="remark | *[@condition='future']">
            <!-- Start: Push element name on head of stack -->
      <xsl:sequence select="(name(.), $value)"/>
   </xsl:accumulator-rule>
   <xsl:accumulator-rule phase="end" match="remark | *[@condition='future']">
            <!-- End: Pop first item from stack -->
      <xsl:sequence select="tail($value)"/>
   </xsl:accumulator-rule>
</xsl:accumulator>
```

Node or Element Tag	Value, Changed or Document Start/End
Document node start	()
<section >	
<title >	
Sample Title	
</title>	
<remark >	"remark"
Ready for release	
</remark>	()
<para >	
Sample external content	
</para>	
<section condition="future">	"section"
<title >	
Sample Subsection Title	
</title>	
<remark >	"remark" "section"
For next year, maybe	
</remark>	"section"
<para >	
Additional content	
</para>	
</section>	()
</section>	
Document node end	()

Also Included…

- Instructions for reporting on a tree in XSLT variable (not XML file)
- Oxygen transformation scenario



XSLT Messages in Accumulator Itself

OK	**Not OK**
xsl:accumulator-**rule** / descendant::xsl:message	xsl:accumulator / xsl:message
xsl:accumulator-rule / element()	xsl:accumulator-rule / @select *(if rule contains a message)*
$value – but it returns prior value	accumulator-before() or -after() on the *same* accumulator

XSLT Messages in Templates/Functions

<xsl:message> in template/function can report accumulator values

- Cannot access $value
- Can call accumulator-before() or accumulator-after()
  - At *any* node that you can make the context node
  - Stream processing limits flexibility

Testing

Why Test Accumulators?

- Test-driven investigation of bug
  - You already have the XSpec code, so keep it
- Some accumulators are complicated
- Modularity

Design Patterns with XSpec

A) Test imports your XSLT

   i) Dedicated scenario to check accumulator

   ii) Integrate accumulator checking with other scenario

B) Test runs your XSLT externally

   i) Dedicated...

   ii) Integrate...

Dedicated Scenario, Imports Your XSLT

```
<x:description stylesheet="..." run-as="import" namespace declarations...>
   <x:scenario label="Values of internal-elem accumulator">
      <x:variable name="myv:tree"
         href="../sample-xml/section-with-internal-content.xml"
         select="/"/>
      <!-- x:call satisfies XSpec compiler -->
      <x:call function="true"/>
      <x:expect label="Start of subsection remark has 2 elements in stack"
         test="$myv:tree/sec/sec/remark/accumulator-before('internal-elem')"
         select="('remark', 'section')"/>
      <!-- more x:expect elements to check values at other nodes -->
   </x:scenario>
</x:description>
```

Integrated in Other Scenarios

Same <x:variable> and <x:expect> can go in existing scenarios.

- Near where you test code that uses accumulators
- If scenario organization is not aligned with XSLT code units
- Avoid "workaround" <x:call>

External Transformation Challenges

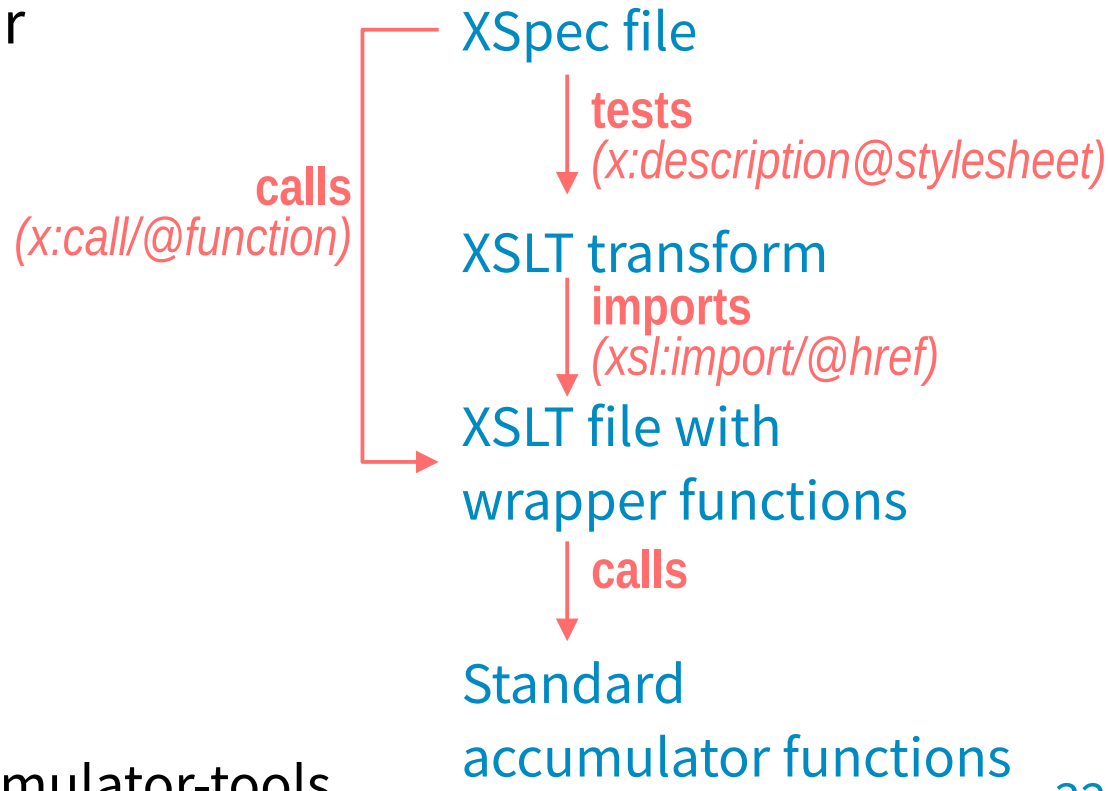
1) <x:expect> cannot access your accumulator directly
2) <x:call> cannot access accumulator-before() or accumulator-after()

Solution: <x:call> calls wrapper around accumulator function

https://github.com/galtm/xslt-accumulator-tools

External Transform Design Pattern

1) <x:expect> cannot access your accumulator directly
2) <x:call> cannot access accumulator-before() or accumulator-after()



Solution: <x:call> calls wrapper around accumulator function

https://github.com/galtm/xslt-accumulator-tools

External Transformation

```
<x:description stylesheet="..." run-as="external" namespace declarations...>
   <x:scenario label="Values of internal-elem accumulator">
      <x:variable name="myv:tree"
         href="../sample-xml/section-with-internal-content.xml"
         select="/"/>
         <x:scenario label="Start of subsection remark">
            <x:call function="at:accumulator-before">
                <x:param select="$myv:tree/sec/sec/remark"/>
                <x:param select="'internal-elem'"/>
            </x:call>
            <x:expect label="2 elements in stack" select="('remark', 'section')"/>
         </x:scenario>
      <!-- more x:scenario elements to check values at other nodes -->
   </x:scenario>
</x:description>
```

Comparison of Techniques

Approach	Software	Basic Procedure	Short-Term Use Case Related to Accumulators	Long-Term Benefit?
Oxygen debugger	Oxygen	Configure; run; analyze states	View selected values, as part of larger exploration	No
Trace	Saxon (PE/EE)	Configure; run; analyze log	View values for entire tree	No
Report	XSLT Accumulator Tools	Configure; run; analyze report	View values for entire tree	Maybe
XSLT messages		Insert code; run; analyze log	View selected values, or identify which rules are being matched	Maybe
XSpec tests	XSpec	Write tests; run; check results	View and verify selected values	Yes

```
/accumulator-after('takeaway')[1]
```

Accumulators are not that opaque

Thank You

Questions?