Saxon - Bug #4640

Reference to a variable as an XPath step

2020-07-11 19:03 - Ken Holman

Status:	Closed	**Start date:**	2020-07-11
Priority:	Normal	**Due date:**	
Assignee:	Michael Kay	**% Done:**	100%
Category:	XPath conformance	**Estimated time:**	0:00 hour
Sprint/Milestone:		**Spent time:**	0:00 hour
Legacy ID:		**Fixed in Maintenance Release:**	10.2, 9.9.1.8
Applies to branch:	10, 9.9, trunk	**Platforms:**	
Fix Committed on Branch:	10, 9.9, trunk		

Description

In Saxon9HE 9.8.0.15 this expression works fine using the first XPath step as an "if" statement:

```
  <xsl:copy-of
      select="self::table-wrap[ancestor::fig]/$c:f/c:figTableCaptionFonts/@*"/>
```

In Saxon9HE 9.9.1.7 the expression adds nothing to the result tree.

The $c:f is a tunnel parameter that is initialized with a global variable that is a massage of another global variable:

```
<xsl:variable name="c:CENcoverFormatting" as="document-node()">
  <xsl:document>
    <xsl:for-each select="($c:mainFormattingDef)/*">
      <xsl:copy>
        <xsl:attribute name="{name(.)}"/>
        <xsl:copy-of select="@*"/>
      </xsl:copy>
    </xsl:for-each>
  </xsl:document>
</xsl:variable>

...

  <xsl:next-match>
    <xsl:with-param name="c:f" tunnel="yes" select="$c:CENcoverFormatting"/>
  </xsl:next-match>

...

<block x:h="div*{name(.)}-caption" x:ignoreXHTML=""
       text-align="center">
  <xsl:copy-of
select="self::table-wrap[ancestor::fig]/$c:f/c:figTableCaptionFonts/@*"/>
```

Unfortunately, when I isolate the expression in a sample XSLT file, the expression works in both versions of Saxon. There is a 21-stylesheet-fragment tree acting on an XML file of 4300 lines where the problem came to light in the 17000 lines of output. I'm reluctant to package this all up in a submission.

Using 9.8.0.15 the output does have x:figTableCaptionFonts="" and font-size="9pt":

Using 9.9.1.7 the output doesn't have x:figTableCaptionFonts="" and font-size="9pt":

I'm guessing this is an optimization thingy, but I am but a user.

What would you have me put into my example to try and recreate the issue?

Thanks for your guidance!

. . . . . Ken

History

#1 - 2020-07-11 19:06 - Ken Holman

(trying again!)

Using 9.8.0.15 the output does have x:figTableCaptionFonts="" and font-size="9pt":

```
<block x:h="div*table-wrap-caption" x:ignoreXHTML="" text-align="center" x:figTableCaptionFonts="" font-size="
9pt" space-after=".5em" space-after.conditionality="retain" font-weight="bold">
```

Using 9.9.1.7 the output doesn't have x:figTableCaptionFonts="" and font-size="9pt":

```
<block x:h="div*table-wrap-caption" x:ignoreXHTML="" text-align="center" space-after=".5em"
space-after.conditionality="retain" font-weight="bold">
```

#2 - 2020-07-11 19:33 - Michael Kay

I'm afraid it's impossible to diagnose this kind of issue unless there's something I can run. A small repro is ideal, but if you can't produce that then a big complicated repro will do. Without a repro (especially for cases of "wrong results" rather than a crash) there's basically no way we can investigate it.

#3 - 2020-07-11 19:53 - Ken Holman

Thanks, Mike. I was just looking for guidance on what to do with my test stylesheet to try and trigger the problem.

Go ahead and close the ticket and I'll code around the issue ... I'm probably not doing this very much in my work ... it is just very handy to do and it was working before. It was a real surprise when our product stopped working correctly and it has taken some hours to narrow it down and try and recreate it in a short test.

. . . . . Ken

#4 - 2020-07-11 20:13 - Michael Kay

You could try and see if it's an optimization problem by running with optimizations disabled - use the -opt flag on the command line. To drill down, you can suppress optimizations selectively (function inlining and loop lifting are the most usual culprits). Running with the -explain option gives a trace of which optimizations are being performed.

#5 - 2020-07-11 20:25 - Michael Kay

I'm wondering a little bit what this code is trying to do. Using a variable reference on the RHS of "/" is legitimate, but it's not very usual.

```
self::table-wrap[ancestor::fig]/$c:f/c:figTableCaptionFonts/@*
```

The semantics here are that you get as many instances of $c:f as there are nodes in self::table-wrap[ancestor::fig], and the duplicate nodes then get eliminated, so you end up with either zero or one. The expression is equivalent, I think, to

```
if (exists(self::table-wrap[ancestor::fig]) then $c:f/c:figTableCaptionFonts/@* else ()
```

So you could try that rewrite and see if it works.

Also related is that the type of the variable isn't declared (as far as I can see from the information given; if Saxon doesn't know statically whether the expression on the RHS of "/" is going to yield nodes or atomic values, then it has too generate code that can handle either -- and it will often dismantle this code later in the optimization process as more information becomes available, e.g. during function inlining). So adding a variable declaration could make a difference.

#6 - 2020-07-11 21:48 - Ken Holman

Yes, in fact my first sentence of this ticket reads "this expression works fine using the first XPath step as an "if" statement" and the cardinality of the RHS reference to $c:f is going to be 0..1 since the LHS evaluates to 0..1 along the self:: axis. I use this often, so it is not unusual in my stylesheets.

And, yes, when I already had changed it to your suggested if(), it worked fine. But, as you say, my request is legitimate. And I do this a number of times in my code because it is a handy expression. I have to find all my uses of this approach and replace them.

But it turns out that the bug is not that the expression is optimized out, but that it is optimized in! When I quoted the examples above, I got them mixed up because I was presuming the problem to be that attributes should have been added when, in fact, they should not have been added. I realized this when I had replaced the expression.

It turns out there is no fig ancestor and so the LHS should evaluate to () and the RHS should be ignored, but the RHS is being processed. Consider this unchanged code but adding a diagnostic attribute:

```
<block x:h="div*{name(.)}-caption" x:ignoreXHTML=""
       text-align="center">
  <xsl:copy-of
select="self::table-wrap[ancestor::fig]/$c:f/c:figTableCaptionFonts/@*"/>
<xsl:attribute name="x:debug" select="
'DEBUG',c:xpath(.),exists(self::table-wrap[ancestor::fig]),self::table-wrap[ancestor::fig]/$c:f/c:figTableCapt
ionFonts/@*/name()"/>
```

... which produces this output:

```
<block x:h="div*table-wrap-caption" x:ignoreXHTML="" text-align="center" x:figTableCaptionFonts="" font-size="
9pt"
  x:debug="
DEBUG /standard/back[1]/app-group[1]/app[1]/sec[2]/sec[7]/sec[1]/list[1]/list-item[3]/p[2]/table-wrap[1] false
 x:figTableCaptionFonts x:figTableCaptionFonts font-size" space-after=".5em" space-after.conditionality="
retain" font-weight="bold">
```

You can see that font-size= is not part of the literal result element, and the debug attribute reveals the exists() evaluates to false() and "fig" is not in the XPath address, and the copy of the faulty XPath expression again incorrectly enumerates the attributes.

So this shows that the xsl:copy-of copied the variable's attributes to the result tree even though the exists() indicates that the LHS does not exist.

And it happens in EE as well.

But still I have not been able to reduce the issue to a short example.

#7 - 2020-07-11 23:08 - Michael Kay

I guess it's probably a loop-lifting problem - the expression on the RHS being moved out of the implicit loop defined by the "/" operator because it has no functional dependency on the value of the expression on the left.

#8 - 2020-07-24 19:29 - Michael Kay

I have reproduced this as XSLT3 test case select-0302.

#9 - 2020-07-24 19:40 - Michael Kay

Well, it's definitely an optimization rewrite, but I'm having trouble seeing where it's happening - it's not logged by -explain. It doesn't seem to be the LoopLifter, nor does it seem to be SlashExpression.promoteFocusIndependentSubexpressions.

#10 - 2020-07-24 19:59 - Michael Kay

It's happening in DocumentSorter.optimize() where it's trying to avoid sort operations. The offending code is at line 142:

```
                // docOrder(A/B) can be rewritten as docOrder(B) in the case where B returns nodes
                // and is independent of the focus. We already know it returns nodes otherwise we wouldn't be
here.
                if (!ExpressionTool.dependsOnFocus(rhs) &&
                        !rhs.hasSpecialProperty(StaticProperty.HAS_SIDE_EFFECTS) &&
                        rhs.hasSpecialProperty(StaticProperty.NO_NODES_NEWLY_CREATED)) {
                    setBaseExpression(slash.getRhsExpression());
                    return optimize(visitor, contextInfo);
                }
```

I think that in this situation we should probably be rewriting docOrder(A/B) as head(A) ! docOrder(B).

#11 - 2020-07-24 20:07 - Michael Kay

- Category set to XPath conformance

- Status changed from New to Resolved

- Assignee set to Michael Kay

- Priority changed from Low to Normal

- Applies to branch 10, trunk added

- Fix Committed on Branch 10, trunk added

Fixed as proposed.

#12 - 2020-08-26 10:23 - O'Neil Delpratt

- % Done changed from 0 to 100

- Fixed in Maintenance Release 10.2 added

Bug fix applied in the Saxon 10.2 maintenance release.

#13 - 2020-08-26 10:30 - O'Neil Delpratt

- Status changed from Resolved to Closed

#14 - 2020-10-16 13:15 - Michael Kay

- Applies to branch 9.9 added

- Fix Committed on Branch 9.9 added

Retrofitted the patch to the 9.9 branch.

#15 - 2020-10-16 13:15 - Michael Kay

- Status changed from Closed to Resolved

#16 - 2020-10-22 18:17 - O'Neil Delpratt

- Status changed from Resolved to Closed

- Fixed in Maintenance Release 9.9.1.8 added

Bug fix applied on the Saxon 9.9.1.8 maintenance release.