Saxon - Bug #4399

functx:non-distinct-values says cannot compare xs:integer to xs:integer, in 9.8 EE only

2019-11-27 18:32 - Priscilla Walmsley

Status:	Closed	**Start date:**	2019-11-27
Priority:	Normal	**Due date:**	
Assignee:	Michael Kay	**% Done:**	0%
Category:	XSLT conformance	**Estimated time:**	0:00 hour
Sprint/Milestone:		**Spent time:**	0:00 hour
Legacy ID:		**Fixed in Maintenance Release:**	9.9.1.6
Applies to branch:	9.8, 9.9, trunk	**Platforms:**	
Fix Committed on Branch:	9.9, trunk		

Description

When the attached XSLT runs in EE, it results in the error:

Engine name: Saxon-EE 9.8.0.12 Description: Cannot compare xs:integer to xs:integer

but it works in HE and PE 9.8.0.12

History

#1 - 2019-11-28 01:14 - Michael Kay

- Category set to XSLT conformance

- Status changed from New to In Progress

- Assignee set to Michael Kay

- Priority changed from Low to Normal

Problem reproduced.

I'm afraid the comical error message is a bit of an old chestnut. When we're comparing A and B, and get a ClassCastException, we report "cannot compare A to B, because that's how we detect the error in the normal course of events. Of course if the ClassCastException is caused by some internal bug then the message is nonsense.

#2 - 2019-11-28 01:26 - Michael Kay

Stack trace of the exception is:

```
java.lang.ClassCastException: net.sf.saxon.value.Int64Value cannot be cast to net.sf.saxon.value.UntypedAtomic
Value
 at net.sf.saxon.expr.sort.UntypedNumericComparer.compareAtomicValues(UntypedNumericComparer.java:183)
 at net.sf.saxon.expr.sort.UntypedNumericComparer.comparesEqual(UntypedNumericComparer.java:223)
 at net.sf.saxon.expr.ValueComparison.compare(ValueComparison.java:742)
 at net.sf.saxon.expr.ValueComparison.effectiveBooleanValue(ValueComparison.java:703)
```

The UntypedNumericComparer is supposed to handle comparisons of xs:untypedAtomic to numerics (it tries to do this "smartly", returning false for cases like "123" = 5 without actually doing a string-to-integer conversion).

The question here is, why has an UntypedNumericConverter been allocated? Something must have gone wrong with the type inferencing.

#3 - 2019-11-28 01:37 - Michael Kay

As often happens with artificial test cases involving literals as input, the optimizer has gone to town on this one:

```
OPT : At line 12 of file:/Users/mike/bugs/2019/4399-walmsley/test.xsl
OPT : Replaced general comparison by value comparison
OPT : Expression after rewrite: (count($Q{}seq[. = $val])) gt 1
OPT : At line 4 of file:/Users/mike/bugs/2019/4399-walmsley/test.xsl
OPT : Moved function functx:non-distinct-values inline:
OPT : Expression after rewrite: let $seq := (1, 2, 3, 3) return (for $val in distinct-values($seq) return ($va
l[(count($seq[. = $val])) gt 1]))
OPT : At line 12 of file:/Users/mike/bugs/2019/4399-walmsley/test.xsl
```

```
OPT : Pre-evaluated function call fn:distinct-values(...)
OPT : Expression after rewrite: (1, 2, 3)
OPT : At line 12 of file:/Users/mike/bugs/2019/4399-walmsley/test.xsl
OPT : Replaced general comparison by value comparison
OPT : Expression after rewrite: . eq $val
OPT : At line 4 of file:/Users/mike/bugs/2019/4399-walmsley/test.xsl
OPT : Inlined constant variable seq
OPT : Expression after rewrite: (1, 2, 3, 3)
OPT : At line 12 of file:/Users/mike/bugs/2019/4399-walmsley/test.xsl
OPT : Rewrite count()>=N as:
OPT : Expression after rewrite: exists(tail((1, 2, 3, 3)[. eq $val]))
OPT : At line 12 of file:/Users/mike/bugs/2019/4399-walmsley/test.xsl
OPT : Lifted (tail((1,...)[. eq $val])) above ($val[fn:exists(...)]) on line 12
OPT : Expression after rewrite: let $Q{http://saxon.sf.net/generated-variable}v0 := tail((1, 2, 3, 3)[. eq $va
l]) return ($val[exists($Q{http://saxon.sf.net/generated-variable}v0)])
OPT : At line 12 of file:/Users/mike/bugs/2019/4399-walmsley/test.xsl
OPT : Lifted (bytecode(fn:exists(...))) above ($val[$vv:v0]) on line 12
OPT : Expression after rewrite: let $Q{http://saxon.sf.net/generated-variable}v0 := bytecode(exists($Q{http://
saxon.sf.net/generated-variable}v0)) return ($val[$Q{http://saxon.sf.net/generated-variable}v0])
OPT : At line 12 of file:/Users/mike/bugs/2019/4399-walmsley/test.xsl
OPT : Created indexed filter expression:
OPT : Expression after rewrite: IndexedFilterExpression($seq, ., $val)
OPT : At line 12 of file:/Users/mike/bugs/2019/4399-walmsley/test.xsl
OPT : Rewrite count()>=N as:
OPT : Expression after rewrite: exists(tail(IndexedFilterExpression($seq, ., $val)))
OPT : At line 12 of file:/Users/mike/bugs/2019/4399-walmsley/test.xsl
OPT : Lifted (tail($seq[(:indexed:).=$val])) above ($val[fn:exists(...)]) on line 12
OPT : Expression after rewrite: let $Q{http://saxon.sf.net/generated-variable}v0 := tail(IndexedFilterExpressi
on($seq, ., $val)) return ($val[exists($Q{http://saxon.sf.net/generated-variable}v0)])
```

I'm mildly surprised that it hasn't done the whole computation at compile time, since there is no dependency on anything in the source document. The reason an "innocent" query like this can exercise unusual paths in the Saxon code is because real life queries can rarely be optimized quite so heavily.

#4 - 2019-11-28 09:21 - Michael Kay

A peculiarity, which I don't think I've ever seen before, is that it doesn't crash when run with the -explain option.

Without -explain, we seem to following a slightly different path in the optimizer. I suspect that it's simply an effect of the fact that the order in which functions/templates are processed by the optimizer is fairly unpredictable.

Looking more closely at the non-explain path in the debugger, it is actually oprimizing more ambitiously than the -explain case. In the IndexedFilterExpression($seq, ., $val), the variable $seq has has been replaced by its value (1,2,3,3), and the optimizer now tries to undo the indexing because it can see that it has become pointless. But the undoing is its undoing, to coin a phrase, because it doesn't allow for the different focus for ".". In particular, it wrongly decides that the context item type for "." is document-node(), and that it therefore needs to be atomized, and from this point everything falls apart.

#5 - 2019-11-28 10:26 - Michael Kay

- Status changed from In Progress to Resolved

- Applies to branch 9.9, trunk added

- Applies to branch deleted (9.8)

- Fix Committed on Branch 9.9, trunk added

Resolved with a patch to IndexedFilterExpression: on the path in optimize() where it decides to revert to an ordinary filter expression, it now correctly sets the context item type for the predicate of the new filter expression.

#6 - 2019-11-28 10:27 - Michael Kay

- Applies to branch 9.8 added

#7 - 2019-12-06 14:48 - O'Neil Delpratt

- Status changed from Resolved to Closed

- Fixed in Maintenance Release 9.9.1.6 added

Patch committed to the Saxon 9.9.1.6 maintenance release.

Files

saxonissues3.xsl	628 Bytes	2019-11-27	Priscilla Walmsley

dummyin.xml	66 Bytes	2019-11-27	Priscilla Walmsley