Saxon - Bug #4054

Spurious "ambiguous match" warning when a reloaded stylesheet contains a template rule with an overlapping union pattern

2018-11-28 18:53 - Michael Kay

Status:	Closed	**Start date:**	2018-11-28
Priority:	Normal	**Due date:**	
Assignee:	Michael Kay	**% Done:**	100%
Category:	XSLT conformance	**Estimated time:**	0:00 hour
Sprint/Milestone:		**Spent time:**	0:00 hour
Legacy ID:		**Fixed in Maintenance Release:**	9.9.1.2
Applies to branch:	9.9, trunk	**Platforms:**	
Fix Committed on Branch:	9.9, trunk		

Description

When a source stylesheet contains a match pattern such as `match="xxx | /*", and some element matches both parts of the union pattern, Saxon normally takes care to avoid issuing an "ambiguous match" warning. However, when the stylesheet is reloaded from a SEF export file, the two parts of the rule have been split into separate rules and it's not clear there's enough information available to avoid the warning.

History

#1 - 2018-11-28 19:07 - Michael Kay

The code (SImpleMode.java line 859) is avoiding the warning if two Rules have the same action part and the same sequence number.

After reloading, the template rule is split into two with different (but cloned) action parts.

Perhaps testing on the sequence number alone would be adequate? Or the line/column/moduleId of the rule's location?

We could consider commoning up the action part to avoid the duplication.

#2 - 2018-11-28 19:09 - Michael Kay

On the subject of this warning, it would be nice if we could only report it once, rather than once every time the same pair of rules is ambiguously matched.

#3 - 2019-02-27 13:15 - Michael Kay

I created a test case next-match-039 which uses a match pattern match="one[XXX] | one[YYY]" where the same input element matches both predicates. So both branches of the pattern match, and both have the same priority. The body of the template uses xsl:next-match, but the rule is only firing once.

The spec says: *As explained in 6.4 Conflict Resolution for Template Rules, a template rule with no priority attribute, whose match pattern contains multiple alternatives separated by |, is treated equivalently to a set of template rules, one for each alternative. This means that where the same item matches more than one alternative, it is possible for an xsl:next-match instruction to cause the current template rule to be invoked recursively.*

The reference in this text is incorrect: it should point to §6.5 (clause 2), which states:

If the top-level pattern is a UnionExprP consisting of multiple alternatives separated by | or union, then the template rule is treated equivalently to a set of template rules, one for each alternative. These template rules are adjacent to each other in declaration order, and the declaration order within this set of template rules (which affects the result of xsl:next-match if the alternatives have the same default priority) is the order of alternatives in the UnionExprP.

So I think we should definitely be invoking the template rule twice. The reason we aren't is that both rules (branches) have the same value for the sequence property, and the reason for this is so that we can suppress the warning when we get an ambiguous match on two branches of the union pattern.

The rules for warning-on-multiple-match can be read as requiring a warning in this case, or at any rate permitting it. However, warnings are always discretionary and it seems unfriendly to produce a warning in this case; certainly many users would regard it as a regression.

#4 - 2019-02-28 10:28 - Michael Kay

I think that this problem -- especially the original problem of reconstructing things correctly on import -- comes down to a question of what data we are maintaining for a rule. The data we currently maintain is as follows:

- **precedence** - meaning is straightforward
- **priority** - the user-specified or defaulted priority; initially an absent priority is indicated by NaN, but by the time the SEF file is output, we can't distinguish whether the priority was system-allocated or user-allocated
- **sequence** - to simplify slightly, this represents the position of the template rule in declaration order
- **rank** - this is technically redundant, it represents the position of the rule in a sequence of rules sorted first by precedence and then by priority and then by sequence.

The tricky bit is when we split a rule governed by a union pattern into multiple rules. There are two cases to consider: (a) where we are doing the split because the spec says we must, which only happens when there is no user-defined priority attribute, and (b) when we do the split voluntarily, because it can lead to faster matching.

In case (a) we allocate different sequence numbers to the two (or more) sub-rules, so they are treated for nearly all purposes as if they were separate rules to start with. However, we don't want to output errors or warnings if a node matches more than one branch of a union pattern, so we need to recognize when this is the case. We do this by asking whether the two rules have the same action body (tested using "=="). This isn't a very safe test: (i) it doesn't work after export/import, and (ii) it doesn't work if one of the bodies is optimized differently from the other (which is possible because different type information is available).

In case (b) we allocate the same sequence number to the two rules. This means that xsl:next-match knows that it should only invoke the rule once.

I think we can get by with a fairly small tweak to this apparatus. In case (a), we record an extra item of information, the "family". The family of a rule is the sequence number of the first rule in a set of rules that resulted from splitting a union pattern. If two rules are in the same family, then we don't output the error/warning message when both rules are matched.

#5 - 2019-02-28 12:09 - Michael Kay

Now working (9.9 branch, need to retrofit to the development branch).

The main change is that Rule acquires an additional property, partNumber, which is used to distinguish the several rules created for matching the parts of a single UnionPattern. If the split was done for spec reasons (because there is no user-allocated priority) then the parts have different part numbers; if it was done for optimisation reasons, then they all have the same part number. The part number is exported in a new SEF attribute if it is non-zero. Rules created from a single source xsl:template always have the same sequence number, and ambiguities are never reported for two rules if their sequence number is the same.

#6 - 2019-02-28 12:38 - Michael Kay

- Category changed from Diagnostics to XSLT conformance

- Status changed from New to Resolved

- Applies to branch trunk added

- Fix Committed on Branch 9.9, trunk added

#7 - 2019-03-12 18:57 - O'Neil Delpratt

- Status changed from Resolved to Closed

- % Done changed from 0 to 100

- Fixed in Maintenance Release 9.9.1.2 added

Bug issue fixed in the Saxon 9.9.1.2 maintenance release.