BUG 27273 – Review of Accumulators

		Comment	Response
1	18.2	n 18.2, for "An accumulator defines a value that is computed progressively ..." perhaps read "An accumulator defines a series of values that is computed progressively".	Done.
2		We did not find any rule that actively makes it impossible to apply an accumulator to a tree rooted in an attribute or namespace node; MSM wondered whether the language might not be simpler to describe if we didn't forbid it. MK eventually said there probably isn't any explicit rule.	Stated that accumulators can apply to trees rooted at any kind of node, but in the case of attribute and namespace nodes, there is no way to obtain the accumulator value.
		ABr did find a place saying that attribute and namespace nodes are not visited in the tree traversal for accumulators.	The place is 18.2.3. 18.2.3 "The traversal of a tree contains two traversal events for each node in the tree, other than attribute and namespace nodes"
		MSM suggested that "Accumulators can apply to trees rooted at any kind of node, other than attribute and namespace nodes." be revised to say simply "Accumulators can apply to trees rooted at any kind of node." It would leave the point of the note untouched, and avoid raising the question we spent time on.	Done.
4	18.2.2	- what is "tree-walking order" (18.2.2)	
		Possible solution: replace "traversing a tree in tree-walking order" with "traversing a tree, as follows."	Done.
5	18.2.1	For The accumulator-after function, however, is restricted by virtue of the streamability rules to appear after any instruction that reads the descendants of the node in question. MSM proposes The accumulator-after function, however, is not guaranteed streamable if it appears after any instruction that reads the descendants of the node in question.	
		MK was a little unhappy with the awkwardness of the new phrasing, but agreed to consider a change here.	Rephrased the para to talk of guaranteed streamability.
6		In 18.2.1, the following paragraph raises the same issue: When streaming, the applies-to pattern must be motionless. Note that a pattern of the form document-node(element(fpml)) is classified as motionless even though it involves a small amount of look-ahead. MSM thought we might want to define "When streaming" as shorthand for something involving "guaranteed streamability", to avoid having to rephrase all of the passages that use "when streaming" in this way.	Rephrased: "For an accumulator to be <termref def="dt-guaranteed-streamable"/>, …"
7	18.2.2	In para 3 ("On each node visit") it's worth mentioning applies-to.	Done.
8		- does accumulator-after('x') succeed if 'x' has no phase="end"?	Added a note to say yes, it does succeed.

		We discussed this and concluded that the answer entailed by the current text is "yes". (There may be a little confusion caused by the references to an accumulator having "two values" for a node; if the value doesn't change, some readers will think that only one value is involved, not two values which are the same.) Note that the text says that if there is no matching rule (no phase="end"), the value does not change; this does not mean that the post-descent value is necessarily the same as the pre-descent value, because the accumulator's value may have changed while visiting the descendants.	
9	18.2.3	In step 2a, applies-to should be mentioned.	Done.
10		In step 4 of the last list, we discussed briefly whether the focus for the evaluation of the initial value should be the global context item; the wording here aligns with that of 18.2.1 ("The initial value of the accumulator"), so we concluded that the text does reflect the decision we made when we discussed this question.	No change.
11		- the argument for accumulator-before does not need to be motionless (it does not make sense, it returns a property of the current node, consider a consuming variable in the same scope, such limits only apply to accumulator-after)	No change.
12		- accumulator-before could be given an extra argument that points to a node (as with fn:node-name, fn:nilled etc) Allowing a node argument would indeed make the function more convenient in non-streaming contexts, but would be just tricky enough that the WG did not want to undertake it now.	No change.
13	18.2.8	the first example uses a non-streamable pattern (positional predicate) >	Fixed.
14		third example uses function fn:accumulator >	Fixed.
15	19.8.8.1	>- I don't get the necessity of item 4 in 19.8.8.1, wouldn't it then also >apply to non-streaming cases as an error scenario? There is no problem - no action.	No change.
16		>- Also about that point: two rules match "foo", one phase="end", one >phase="start": always free-ranging? There is no problem - no action.	No change.
17		>- Item 6 in that list, would it apply to xsl:param/@select (part of >declaration)? MSMQ: I think we do need a change of wording here. Otherwise "contained in" is being used in a way that excludes some parent/child relations in XML.	Note also that literal result elements are not technically instructions, although we include them in section 19.8.4 as if they were... 1. change rule 6 "If the function call is not contained in an instruction," to "If no enclosing node of the function call is part of a sequence constructor". Define "enclosing node" as a generalization of "enclosing element" which is defined in 5.4.1. 2. In note 6, explain that the parts of a sequence constructor are defined in 5.8. This therefore includes attributes of instructions and LREs, and text value templates. Also explain that it does NOT include xsl:param or xsl:sort elements.

		3. In rule 8, change "If no containing instruction has a preceding sibling instruction whose sweep is consuming" to "If no enclosing node N of the function call has a preceding-sibling node P such that N and P are part of the same sequence constructor and the sweep of P is consuming, ...
	Examples: <xsl:template expand-text="yes">{accumulator-after('test')}</xsl:template> <xsl:variable select="accumulator-after('tes')" /> --> as instruction, currently allowed (?) <xsl:sort select="accumulator-after('bla')" /> Maybe we should say "contained in a sequence constructor" instead, but that does not cover xsl:sequence in xsl:fork... ACTION-2014-11-06-002: Mike Kay to look at 19.8.8.1 item 6 in view of Abel's comments and examples identified at telcon 2014-11-06.	
	> 18.2.4 fn:accumulator-before - last telcon ended here	
18	The last sentence is a bit "opaque" in the view of some reviewers... Perhaps for "irrespecive of the nesting of sections" read "in the order in which sections are encountered in a depth-first traversal of the tree" (or something shorter and clearer)	Deleted the phrase "irrespective of the nesting of sections".
19	> 18.2.5 fn:accumulator-after	
	"Rules" section lacks the "green" from 18.2.4.	Fixed.
19a	Add a note saying that it is not necessary to have an accumulator rule for the "after" in order to use fn:accumulator-after	Done.
20	> 18.2.6 Visibility and Overriding of Accumulators	
	In the first paragraph: stylesheet -> package.	Done.
21	> 18.2.7 Streamability of Accumulators	
	"motionless" - add a link to the classification section?	Done.
22	MSMQ: pecifying streamable="yes" on an xsl:accumulator element declares an intent that the accumulator should be guaranteed streamable according to these criteria. I think that's not quite right. I think it declares an intent that the accumulator should be usable in a streaming context. If I'm using Saxon 42, which has solved all sorts of additional problems, I may have a streamable accumulator that is not guaranteed streamable. I think my concern would be addressed (and the paragraph would still work) if for "guaranteed streamable according to these criteria" we say "streamable" It might be possible to delete the sentence entirely -- the next sentence does the real work here. It is a valid comment and Mike Kay will review.	Changed "declares an intent that it should be guaranteed strreamable" to "declares an intent that it should be streamable, either because it is guaranteed streamable, or because it takes advantage of streamability extensions offered by a particular processor". Made this change not just for accumulators but for the other "declared streamable" constructs where the same form of words appears.
23	Abel: @initial-value is marked required, but the text does not reflect this (18.2.1), also, I don't think its semantics requires it to be required. Decided to leave @initial-value mandatory.	Done.
	> 18.2.8 Examples of Accumulators	

24	First example: match attribute has a positional predicate, which is not allowed for streaming. Easiest fix is to drop the "[1]" (thus being the last title)	Done (duplicate)
25	Example 3 (Output Hierarchic Section Numbers) Has fn:accumulator (not -before or -after).	Done (duplicate)
26	Example 4: Suggested to enhance the example to show how the accumulator is actually used.	Done.