WLE Rules in XML Format for Representing Label Generation Rulesets and in Regular Expressions

The document “Representing Label Generation Rulesets in XML” [XML-LGR] contains a series of match operators that function essentially like regular expression operators. Because of the representation in XML, there are some details in representation.

This document describes the equivalents as well as the differences. The goal is to make it easier for users familiar with regular expressions to write LGR definitions using the [XML-LGR] specification, or for implementers to implement an evaluator for the [XML-LGR] format using an existing regular expression library.

Equivalents between Regular Expression and XML-LGR syntax elements

The following table lists the equivalents.

Regex Syntax	XML Equivalent	Notes
Quantifiers		
*****	`count="0+"`	zero or more
+	`count="1+"`	one or more
?	`count="0:1"`	one or more
{n,m}	`count="n:m"`	from *n* to *m* occurrences
Wildcards		
.	`<any/>`	match any character
Literals		
a	`<char cp="0061"/>`	match literally
Alternation		

(a|b)	`<choice>`
	` <char cp="0061"/>`
	` <char cp="0062"/>`
	`</choice>`
		match alternates literally
		Note: because in XML-LGR there is no explicit divider, each nested element in a choice is a single alternative; must treat (aa|b) as

		if the regex was ((aa)|b) using explicit grouping for “aa”
Grouping		
(ab)	<rule> <char cp="0061"/> <char cp="0062"/> </rule>	use an anonymous *rule* element to group; elements nested inside a rule are evaluated in order; useful for attaching quantifiers to the entire group e.g. (ab)+ to match “abababab”;
Positional		
^	<start/>	use *start* element to match the start of the label
$	<end/>	use *end* element to match the end of the label
Character Classes		
[ab]	<class> <char cp="0061"/> <char cp="0062"/> </class> *or* <class>0061 0062</class>	match any of the code points in the set, members listed individually
[a-z0-9]	<class> <char first-cp="0061" last-cp="007A"/> <char first-cp="0030" last-cp="0039"/> </class> *or* <class>0061-007A 0030- 0039</class>	match any of the code points in the set, members given by ranges (inclusive) Note: ranges and explicitly listed code points can be intermixed
[\p{sc=Latn}]	<class property="sc:Latn"/>	all code points having the specified property.

Set operators		
[^ab]	<complement> <class>0061 0062</class> </complement>	negate the set
[[ab]&&[cd]]	<intersection> <class>0061 0062</class> <class>0063 0064</class> </intersection>	intersection of *exactly* two classes
[[ab][cd]]	<union> <class>0061 0062</class> <class>0063 0064</class> </union>	union of two or more classes
[[ab]&&[^cd]] *or* **[[ab]-[cd]]**	<difference> <class>0061 0062</class> <class>0063 0064</class> </difference>	difference – subtract the second set from the first.
[[ab][cd]] - [[ab]&&[cd]]	<symmetric-difference> <class>0061 0062</class> <class>0063 0064</class> </symmetric-difference>	no native regex operator for it, but can be represented using *union - intersection*
Anchor and Look-ahead / Look-behind		
N/A	<look-behind>, <look-ahead> and <anchor> elements	no direct equivalents, see discussion of differences

Differences Between Regex and XML-LGR Specification

1. [XML-LGR] uses greedy matching with yielding. If a greedy match of an earlier part of a pattern prevents a match for a tail, the greedy match will give up already matched code points if that allows the whole pattern to match.
2. Some regex library functions perform whole string matching against the pattern. LGR-XML rules behave more like a “search” or “find” operator, in that, by default, they match any interior substring. In order to achieve a whole string match, use the <start/> and <end/> elements.
3. The <anchor/> element corresponds to a specific code point or sequence at a specific location. It is otherwise equivalent to a literal (a or aa) that can be set at the time a label

is matched against a rule. An <anchor> must follow and/or precede a look-behind/ahead.

4. While regex libraries support look-ahead and look-behind assertion syntax using (?= ...) and (?< ...) operators, they do not map directly to [XML-LGR] because the position of the anchor is significant, and a match is only valid if it contains the particular instance of the code point or sequence in the label (there could be multiple instances).
5. Classes and rules can be defined external to a pattern and “invoked” by reference. Essentially they function like a subroutines.
6. The Unicode properties recommended for LGRs is a subset.
7. In a regex character class, some characters(like ‘-‘ or ‘^’) have to be escaped or listed in special positions. These restrictions do not apply in [XML-LGR]. Instead, it is recommended to list members and ranges in code point order.