Model of Sequence Comparison for XACML

September 12, 2002

Author: Michiharu Kudo

This is a proposal of a model for sequence comparison.

1. Basic Comparison on Sequences

There are a couple of patterns representing typical comparisons on two sequences. Sequence means an ordered list of some primitive types (string data type etc.). The length of the sequence is either zero, one or more. Zero sequence is called an empty sequence. In the following examples, I assume that the length of the first sequence is *m* and the length of the second sequence is *n*. Figure 1 shows four patterns of comparison on two sequences.

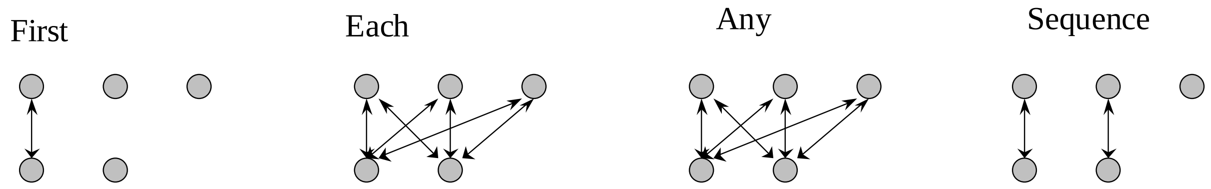


Figure 1 Patterns of comparison on two sequences

A pattern called “First” means that it compares the first element of the first sequence with the first element of the second sequence. The semantics how to compare two values are determined by other description (e.g. string-equal). Actual semantics is determined by combining “First” comparison on sequences with e.g. “string-equal” rule. The result is determined by one-time comparison. The comparison should raise error if either one of the sequences is an empty sequence.

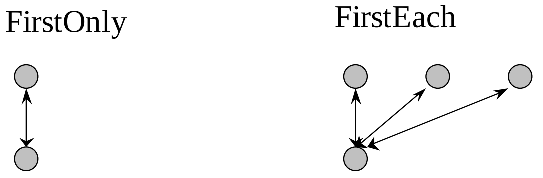
A pattern called “Each” means that it compares each element of the first sequence with the *first* element of the second sequence. If none of the comparisons becomes true, then the result of this comparison becomes false. If one or more comparisons become true, then it goes to the next round. It compares each element of the first sequence with the *second* element of the second sequence. The same comparison is applied and repeated. Therefore the result is determined by at most *m* times *n* comparisons.

A pattern called “Any” means that it compares each element of the first sequence with each element of the second sequence. If none of the comparisons becomes true, then the result of this comparison becomes false. Other wise, the result becomes true. This means *existential* comparison and *at least one match* comparison. Therefore the result is determined by at most *m* times *n* comparisons.

A pattern called “Sequence” means that it compares an element of the first sequence and an element of the second sequence at *i*’s position (from 1 to *m*). Therefore, if the length of the first sequence and the length of the second sequence are different, the result of this comparison becomes false. If the lengths are the same, if comparisons on every element become true, then the result of this comparison becomes true. Therefore the result is determined by *m* (=*n*) comparisons.

2. Derived Comparisons on Sequences

Besides the basic four patterns of the comparisons, there are a couple of derived patterns. Figure 2 shows two of them. The difference from the basic pattern is when error is returned with regard to the length of the sequence.



FirstOnly

If the length of either sequence is more than one, an error is returned.

FirstEach

If the length of the second sequence is more than one, an error is returned.

Figure 2 Derived patterns of comparison on two sequences

A pattern called “FirstOnly” has the same semantics with a pattern “First” but it requires the length of each sequence must be one. If the length is not one, it returns an error.

A pattern called “FirstEach” has the same semantics with a pattern “Each” but it requires the length of the second sequence must be one. If the length of the second sequence is not one, it returns an error.

3. Example XACML Policy Specification

3.1 Comparison in Target Element

Match function specification in Target element allows *AttributeDesignator or AttributeSelector for the first argument (produces a sequence) and allows only AttributeValue for the second argument (one element). This means that the semantics of the Match function is “First”, “FirstOnly”, or “FirstEach”. From the discussion so far, “FirstEach” seems the most appropriate pattern for Match functions. The policy

```
<Target>
  <Subjects>
    <Subject MatchId=”function:string-equal” ComparisonBase=”FirstEach”>
      <AttributeSelector RequestContextPath=”/a/b”>
      <AttributeValue>abc</AttributeValue>
    </Subject>
  </Subjects>
</Target>
```

If the schema assumes “FirstEach” as a default value for the ComparisonBase attribute, then it can be omitted. The resultant policy fragment has no change from the current one.

```
<Target>
  <Subjects>
    <Subject MatchId=”function:string-equal” >
      <AttributeSelector RequestContextPath=”/a/b”>
      <AttributeValue>abc</AttributeValue>
    </Subject>
  </Subjects>
</Target>
```

The above semantics corresponds to the following condition specification:

```
<Condition>
  <Apply FunctionId=”function:string-equal” ComparisonBase=”FirstEach”>
    <AttributeSelector RequestContextPath=”/a/b”>
    <AttributeValue>abc</AttributeValue>
  </Apply>
</Condition>
```

3.2 Comparison in Condition Element

Users can use any comparison patterns in condition function if they wish.

Example 1)

```
<Condition>
  <Apply FunctionId=”function:string-equal” ComparisonBase=”First”>
    <AttributeSelector RequestContextPath=”/a/b”>
    <AttributeSelector RequestContextPath=”/c/d”>
  </Apply>
</Condition>
```

Example 2)

```
<Condition>
  <Apply FunctionId=”function:string-equal” ComparisonBase=”Any”>
    <AttributeSelector RequestContextPath=”/a/b”>
    <AttributeSelector RequestContextPath=”/c/d”>
  </Apply>
</Condition>
```

Example 3)

```
<Condition>
  <Apply FunctionId=”function:string-equal” ComparisonBase=”Sequence”>
    <AttributeSelector RequestContextPath=”/a/b”>
    <AttributeSelector RequestContextPath=”/c/d”>
  </Apply>
</Condition>
```

4. Relationship with Existing Models

4.1 XACML Function Specification

In Function draft 0.8 uses the following syntax for comparing a sequence (produced by AttributeSelector) with a constant value:

```
<Condition>
  <Apply FunctionId="function:string-member-of">
    <AttributeValue>bb1</AttributeValue>
    <AttributeSelector RequestContextPath="/a/b/c/text()">
  </Apply>
</Condition>
```

Using the proposed comparison model, the above syntax becomes:

```
<Condition>
  <Apply FunctionId="function:string-equal" ComparisonBase=”FirstEach”>
    <AttributeSelector RequestContextPath="/a/b/c/text()">
    <AttributeValue>bb1</AttributeValue>
  </Apply>
</Condition>
```

We can specify string-match, integer-equal, integer-greater-than, etc. in the same manner.

```
<Condition>
  <Apply FunctionId="function:integer-equal" ComparisonBase=”FirstEach”>
    <AttributeSelector RequestContextPath="/a/b/c/text()">
    <AttributeValue>100</AttributeValue>
  </Apply>
</Condition>
```

4.2 XPath Data Model

In XPath 2.0, there are several types of comparisons: a *value comparison* (e.g. eq), a *general comparison* (e.g. =), and a *sequence comparison* (e.g. sequence-equal). The value comparison compares two primitive types, which corresponds to “FirstOnly” pattern in this proposal. The general comparison compares two sequences, which corresponds to “Any” pattern in this proposal. The sequence comparison compares two sequences, which corresponds to “Sequence” pattern in this proposal.