Differentiating Elements

When we want to determine if 2 elements are different there are a couple of situations to consider

1. The two elements have different names.
2. The two elements have the same name but different namespaces
3. The two elements have the same name and namespace, but occur in different places in the schema (so by definition cannot both be global elements). These are differentiated by their line position in the schema, and are fuzzed separately.
4. The two elements have the same name and namespace, occur in the same place in the schema but two instances have different minOccurs or maxOccurs. These elements are considered of the same type and are fuzzed together, and hence appear once in the ElementNodeList. For the purposes of Occurrence fuzzing the unique values of minOccurs and maxOccurs that they have needs to be recoverable and hence they are stored separately in the ElementDB.

The same is true for attributes.

Selecting Nodes to Fuzz

Once the Element DB has been created we have a SortedDictionary of every element in the schema (indexed by XmlIdentifier with minOccurs and maxOccurs) and a list of XmlIdentifiers (which identify an element or attribute by a combination of its XMLQualifiedName and its type definition line position in the XSD).

Thus when we increment our index into our list of elements/attributes we will go through every element/attribute in the schema, even if two have the same name. We do not have any problems in outputting as the state chooses a unique index for the elements/attributes amongst all of them with that name. We get around the problem of elements and attributes have the same name by pre-pending an ‘@’ before the name of attributes.

The problem of elements being able to have the same name is unsolved elsewhere however, like specifying custom fuzzers using namespace and name, currently that is not supported. Similarily for attributes and elements.

## Fuzzing Types

Occurrence Fuzzing

Occurrence fuzzing is tough since we can get schema like

<element name=”node1”>

<complextype>

<sequence minOccurs=”10” maxOccurs=”20”>

<element name=”node2” />

<sequence minOccurs=”1” maxOccurs=”2”>

<element name=”node3” />

</sequence>

<choice>

<element name=”node4” />

<element name=”node5” />

</choice>

<element name=”node6” minOccurs=”4” maxOccurs=”12” />

</sequence>

</complextype>

</element>

As can be seen it each instance of each particle (element, sequence or choice) needs to be mapped from the XML to the schema, this is done in XMLToSchemaMapper, so that the Occurrence fuzzer has instance that it can repeat to satisfy all of the possible occurrence values.

The Occurrence fuzzer only attempts to repeat the particles an amount that is within the min and max range specified in the schema, and has been specified by the user in ComplexTypeFuzzerConfig.xml. Each particle is also treated separately, for instance ‘node6’ might be repeated anywhere from 4 to 12 times, and the rest of the nodes are left as they are. There is no fuzzing where one node is repeated x number of times and another node y number of times, except insomuch as already exists in the XML.