

XML Schema and alternatives

Patryk Czarnik

XML and Applications 2015/2016

Lecture 4 - 21.03.2016

Some possibilities of XML Schema we have not learnt too much

- Deriving complex types by restriction
 - restriction of values set, not necessarily of structure
 - super type more general, subtype more specific
- Substitution groups
 - virtual elements that can be substituted with other ones
- Wildcards (**any** and **anyAttribute**)
 - allow to insert any tags or tags from given namespaces
- Nillable elements
 - marking elements as “nonexisting” with **xsi:nil** attribute
 - personally, I don't like this idea...
- Specifying actual type with **xsi:type** in documents

Some drawbacks of XML Schema

- Verbose syntax, specification hard to understand
- Limited power of expression
 - deterministic model required
 - no choice between:
 - text content and element content
 - attribute and element
 - content-aware model not available (in 1.0)
 - no value-aware constraints other than simple identity constraints (in 1.0)
- Extending types only by appending model fragments at the end of a sequence
 - no direct support for adding new elements to a choice
 - available with other techniques: element groups or substitution groups

Alternatives

- DTD – obviously
- RELAX NG (Regular Language for XML Next Generation)
 - by James Clark and Murata Makoto
 - OASIS (2001) and ISO (2003) standard
- Schematron
 - by Rick Jelliffe (1999), developed at Academia Sinica (Taiwan), ISO standard (2006)
- Examplotron
 - by Eric van der Vlist, project active in 2001-2003
- XML Schema 1.1
 - W3C Recommendation, 2001
 - borrows some ideas from the alternatives, mainly Schematron

Simple example in all standards

DTD

```
<!ELEMENT person (first-name+, last-name)>  
<!ELEMENT first-name (#PCDATA)>  
<!ELEMENT last-name (#PCDATA)>
```

XML Schema

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">  
  <xs:element name="person" minOccurs="0" maxOccurs="unbounded">  
    <xs:complexType>  
      <xs:sequence>  
        <xs:element name="first-name" type="xs:string"  
          maxOccurs="unbounded" />  
        <xs:element name="last-name" type="xs:string" />  
      </xs:sequence>  
    </xs:complexType>  
  </xs:element>  
</xs:schema>
```

Simple example in all standards

Relax NG – XML syntax

```
<grammar xmlns="http://relaxng.org/ns/structure/1.0">
  <start>
    <element name="person">
      <oneOrMore>
        <element name="first-name">
          <text/>
        </element>
      </oneOrMore>
      <element name="last-name">
        <text/>
      </element>
    </element>
  </start>
</grammar>
```

Relax NG – compact syntax

```
element person {
  element first-name { text }+
  element last-name { text }
}
```

Simple example in all standards

Schematron

```
<schema xmlns="http://purl.oclc.org/dsdl/schematron">
  <pattern>
    <title>Person rules</title>
    <rule context="/person">
      <assert test="count(first-name) >= 1">Person
        should contain one or more first names.</assert>
    </rule>
    <rule context="/person">
      <assert test="count(last-name) = 1">Person
        should contain exactly one last name.</assert>
    </rule>
    <rule context="/person">
      <assert test="not(*[local-name() != 'first-name'
        and local-name() != 'last-name'])">Person
        should contain no other elements.</assert>
    </rule>
  </pattern>
</schema>
```

Note: This is not the primary intended use of Schematron

Simple example in all standards

Examplotron :)

```
<person>
  <first-name>Adam</first-name>
  <first-name>Maria</first-name>
  <last-name>Abacki</last-name>
</person>
```

OK, there is something more in there...

```
<person>
  <first-name eg:occurs="+">Adam</first-name>
  <last-name>Abacki</last-name>
  <salary>10.00</salary>  <!-- guesses data types -->
  <birth-date>1999-01-01</birth-date>
</person>
```


Relax NG – basic ideas

- Clear theoretical basis: Tree automata with regular expressions specifying content in each node
 - The same model with appropriate restrictions is used by theoreticians to model DTD or XML Schema, but as a kind of “reverse engineering”
- Compact and readable XML syntax
 - Even more compact plain text syntax available
- Model components such as elements, attributes, or text nodes may be mixed together in definitions
- Modularisation available through **define** / **ref** mechanism
 - Equivalent to DTD parameter entities or XSD groups, but with some additional operations to enhance convenience
- No direct support for simple types, but referring to XML Schema types is possible.

Schematron – basic ideas

- Approach different from grammar-based DTD, XSD, and RelaxNG:
 - XPath expressions specify assertions that must hold for instance documents (and elements within them)
 - High power of expression
 - Less convenient (than grammar rules) to write structural definitions
- Official implementation:
 - translation of Schematron scheme to XSLT
 - XSLT evaluates expressions and report errors
- Other implementations available, also for Schematron rules embedded in XML Schema definitions
- XML Schema 1.1 covers most typical Schematron use cases. Probably XSD 1.1 will replace Schematron at all...^{10 / 21}

Non-deterministic model

- **Ambiguous** model:
 - It is not possible to state which particle of the model definition is matched, even if whole document is known.
 - example: $(A,A,A)^+|(A,A)^+$ for document $AAAAAA$
- **Non-deterministic** model
 - similar to (non-)LL(1) grammars, but extended to trees:
 - During one-pass parsing, it is not possible to determine the appropriate definition particle for a current element when only the start tag of the element is seen
 - Some models may be determined just by definition rearrangement, e.g.: $A, A?, A? \rightarrow A, (A, A?)?$
 - But some models may not;
notable example: $(A, B)^*, A?$
- XML Schema avoids non-deterministic models!

Model forbidden in XML Schema, valid in Relax NG

```
<list>
  <odd/>
  <even/>
  <odd/>
  <even/>
  <odd/>
  ...
</list>
```

```
<element name="list">
  <oneOrMore>
    <element name="odd"/>
    <element name="even"/>
  </oneOrMore>
  <optional>
    <element name="odd"/>
  </optional>
</element>
```

Choice between text and element

- We'd like to allow both formats:

```
<phone>1234567</phone>  
<phone><cc>48</cc><ac>22</ac><main>123456</main><int>1313</int></phone>
```

- No possibility in XML Schema (other than mixed content)
- No problem in RelaxNG:

```
<element name="phone">  
  <choice>  
    <text/>  
    <group>  
      <optional><element name="cc"/></optional>  
      <optional><element name="ac"/></optional>  
      <element name="main"/>  
      <optional><element name="int"/></optional>  
    </group>  
  </choice>  
</element>
```

Choice between attribute and element

- Similarly as before, we want to allow both

```
<section title="Introduction">  
  ...
```

and

```
<section>  
  <title>What does <q>Be or not to be</q> mean in fact?</title>  
  ...
```

Relax NG solution:

```
<element name="section">  
  <choice>  
    <attribute name="title"/>  
    <element name="title">  
      <ref name="text-model"/>  
    </element>  
  </choice>  
  <ref name="text-model"/>  
</element>
```

Value-aware constraints

- Relations and constraints other than equality

```
<order>
  <order-date>2014-01-20</order-date>
  <delivery-date>2014-01-25</delivery-date>
  <item>
    ...
    <value>199.90</value>
  </item>
  <item>
    ...
    <value>20</value>
  </item>
  <value>219.90</value>
</order>
```

Value-aware constraints - Schematron solution

```
<pattern>
  <title>Order</title>
  <rule context="order">
    <assert test="delivery-date >= order-date">
      Delivery is not possible before order.</assert>
    </rule>
  <rule context="order">
    <assert test="value = sum(item/value)">
      Order value must be equal to the sum of item values.</assert>
    </rule>
  </pattern>
```


Value-aware constraints - XSD 1.1 solution

```
<xs:complexType name="Order">
  <xs:sequence>
    ...
    <xs:element name="order-date" type="xs:date"/>
    <xs:element name="delivery-date" type="xs:date"/>
    ...
    <xs:element name="item" type="OrderItem" maxOccurs="unbounded"/>
    <xs:element name="value" type="xs:decimal"/>
    ...
  </xs:sequence>
  <xs:assert test="delivery-date >= order-date">
  <xs:assert test="value = sum(item/value)">
</xs:complexType>
```

Content-aware model

- We want to choose one of models depending on the value of a field (attribute or element), e.g:

```
<order>
  ...
  <payment method="transfer">
    <due-date>2014-02-07</due-date>
  </payment>
</order>
```

```
<order>
  ...
  <payment method="card">
    <card-no>4400-1234-1234-1234</card-no>
    <holder>Adam Abacki</holder>
    <exp-date>2014-12-31</exp-date>
  </payment>
</order>
```

Content aware model - Schematron solution

```
<pattern>
  <title>Payment method</title>
  <rule context="payment">
    <assert test="
      @method = 'transfer' and due-date
      or @method = 'card' and card-no and holder and exp-date">
      Payment with a transfer or a card.
    </assert>
  </rule>
</pattern>
```

- It will work provided that type of payment will be defined generally and that all elements will have appropriate types
- But Schema 1.1 offers more natural solution...

Content aware model - XSD 1.1 solution

```
<xs:element name="payment" type="Payment">
  <xs:alternative test="@method = 'transfer'" type="TransferPayment" />
  <xs:alternative test="@method = 'card'" type="CardPayment" />
  <xs:alternative test="not(@method = ('transfer', 'card', 'cash'))"
    type="xs:error" />
</xs:element>

<xs:complexType name="Payment"/>

<xs:complexType name="TransferPayment">
  <xs:complexContent>
    <xs:extension base="Payment">
      <xs:sequence>
        <xs:element name="due-date" type="xs:date"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
  ...
</xs:complexType>

<xs:complexType name="CardPayment">
  <xs:complexContent>
    <xs:extension base="Payment">
      <xs:sequence>
        <xs:element name="card-no" type="CardNumber"/>
        <xs:element name="holder" type="xs:string"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
  ...
</xs:complexType>
```

So why do most people use XSD 1.0?

- Alternatives?
 - XML Schema is a W3C Recommendation
 - In this business it means even more than an ISO standard
 - Some popular technologies, with Web Services (WSDL) at the first place, use XML Schema
- XML Schema 1.1?
 - Still little support in generally available software, especially programming libraries