Relax NG

Relax New Generation

Relax NG was developed by merging two previous schema languages:

Relax	Regular Language for XML (Murata Makoto)
TREX	Tree Regular Expressions for XML (James Clark)

Three Kinds of Schema Languages

1. Constraints are expressed as rules, such as "the element named *phone* must have an attribute named *areaCode* and this attribute's content must follow this specific rule ...". This technique is used by Schematron.

2. Constraints are expressed by a description of each element and attribute, such as "we need an element named *phone*, and it has an attribute named *areaCode*, which looks like ...". This technique is used by DTDs and XML Schemas.

3. Constraints are expressed as patterns that define the permissible elements, attributes, and text nodes using regular expressions. This technique is used by Relax NG.

Key Features of Relax NG

- Simple like DTDs but expressive like XML Schemas.
- Allows two syntaxes: XML syntax and Compact syntax.
- Uses XML Schema data types.
- Permits user-defined data types.
- Supports namespaces.
- Allows modular definitions.
- Treats elements and attributes in a similar manner.
- Has a pattern-based grammar.

Basic Components of Compact Relax NG

An element pattern

element eleName { pattern }

An attribute pattern

attribute attName { pattern }

A text pattern

text

A sequence of patterns

pat_1, pat_2, pat_3, pat_4

A choice of patterns

pat_1 | pat_2 | pat_3 | pat_4

Cardinality of patterns

pat_1?, pat_2*, pat_3+

Grouping

(pat_1 | pat_2), pat_3

In these examples, Relax NG keywords are shown in italics.

Example: Translate phoneA.dtd into Relax NG

File: phoneA.dtd

```
<!ELEMENT phoneNumbers (title, entries)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT entries (entry*)>
<!ELEMENT entry (name, phone, city?)>
<!ELEMENT name (first, middle?, last)>
<!ATTLIST name gender (female | male) #IMPLIED>
<!ELEMENT first (#PCDATA)>
<!ELEMENT middle (#PCDATA)>
<!ELEMENT last (#PCDATA)>
<!ELEMENT phone (#PCDATA)>
<!ELEMENT city (#PCDATA)>
```

File: phoneA.rnc

```
element phoneNumbers
{
    element title { text },
    element entries
    {
        element entry
        {
            element name
            {
                attribute gender { text }?,
                element first { text },
                element middle { text }?,
                element last { text }
            },
            element phone { text },
            element city { text }?
        }*
    }
}
```

Validation

The *xmllint* utility will validate an XML document against a Relax NG schema, but it expects the definition to use the XML syntax for Relax NG.

Another utility program, developed by James Clark and called *Trang*, will translate compact Relax NG syntax into the corresponding XML syntax.

This program is provided in a Java archive file called *trang.jar*.

Usage

% **java -jar trang.jar phoneA.rnc phoneA.rng**

File: phoneA.rng

```
<?xml version="1.0" encoding="UTF-8"?>
<element name="phoneNumbers"
                xmlns="http://relaxng.org/ns/structure/1.0">
  <element name="title">
    <text/>
  </element>
  <element name="entries">
    <zeroOrMore>
      <element name="entry">
        <element name="name">
          <optional>
            <attribute name="gender"/>
          </optional>
          <element name="first">
            <text/>
          </element>
          <optional>
            <element name="middle">
              <text/>
            </element>
          </optional>
          <element name="last">
            <text/>
          </element>
        </element>
        <element name="phone">
          <text/>
        </element>
        <optional>
          <element name="city">
            <text/>
          </element>
        </optional>
      </element>
    </zeroOrMore>
  </element>
</element>
```

Validation with xmllint

% **xmllint --noout --relaxng phoneA.rng phoneA.xml**
phoneA.xml validates

Alternative Validation

James Clark has also provided a utility program, called *Jing*, that validates XML documents using Relax NG while accepting the compact syntax as well as the XML syntax.

Usage

% **java -jar jing.jar -c phoneA.rnc phoneA.xml**
%

% **java -jar jing.jar phoneA.rng phoneA.xml**
%

No output indicates that the XML document is valid with respect to the Relax NG schema.

Now alter *phoneA.xml* to create a document *phoneB.xml*.

- Interchange a *first* and *last* element.
- Duplicate a *phone* element.

% **java -jar jing.jar -c phoneA.rnc phoneB.xml**
/mnt/nfs/fileserv/fs3/slonnegr/xnotes/Relax/phoneB.xml:9:18: error: required elements missing
/mnt/nfs/fileserv/fs3/slonnegr/xnotes/Relax/phoneB.xml:10:19: error: element "first" not allowed in this context
/mnt/nfs/fileserv/fs3/slonnegr/xnotes/Relax/phoneB.xml:21:16: error: element "phone" not allowed in this context

Notes

- Relax NG provides no way to specify a particular schema in an XML document like DTD and XML Schemas do. Any validator needs to know both the XML document and the Relax NG file to carry out the validation.
- No precedence is defined between sequencing and choice. Parentheses must be used to avoid ambiguity.

Example: Using Element Combinations

Remember the DTD *elems.dtd*:

```
<!ELEMENT root (one+, (two | three)+,
                        four*, (five*, six)+, (one | two)?)>
<!ELEMENT one (EMPTY)>
<!ELEMENT two (EMPTY)>
<!ELEMENT three (EMPTY)>
<!ELEMENT four (EMPTY)>
<!ELEMENT five (EMPTY)>
<!ELEMENT six (EMPTY)>
```

In Relax NG we specify an empty element using the keyword *empty*.

element tag { *empty* }

If the element has an attribute, the *empty* keyword may omitted.

element image { *attribute* source { *text* } }

The empty pattern is required only when an element has no content and no attributes.

The compact Relax NG schema on the next page mimics the DTD shown above.

File: elems.rnc

```
element root
{
    element one { empty }+,
    (element two { empty } | element three { empty })+,
    element four { empty }*,
    (element five { empty }*, element six { empty })+,
    (element one { empty } | element two { empty })?
}
```

% **java -jar jing.jar -c elems.rnc e1.xml**
%

This Relax NG schema contains some redundancy.

The solution to this problem is to name some of the patterns to support reuse and reduce the depth of the element nesting.

oneDef = *element* one { *empty* }

Relax NG is designed so that element and attribute names cannot be confused with definition identifiers, which are always introduced on the left side of an equal sign.

one = *element* one { *empty* }

Since a Relax NG schema may define several different elements, we need a way to specify which will be the root of the corresponding XML document.

The keyword *start* indicates which element is the root.

The file on the next page shows a Relax NG schema in which each element is given a name.

File: elems2.rnc

```
start =  element root
         {
             one+,
             (two | three)+,
             four*,
             (five*, six)+,
             (one | two)?
         }
one = element one { empty }
two = element two { empty }
three = element three { empty }
four = element four { empty }
five = element five { empty }
six = element six { empty }
```

Using Choice (the | operation)

In Relax NG the choice operator can be used between two attributes for an element and between an attribute and an element in specifying the content of an element.

The next Relax NG schema is a variation of the phone example with two changes:

- A *name* element may have a *gender* attribute or an *age* attribute (or no attribute at all).
- An *entry* element may have a *phone* element or a *phone* attribute.

File: phoneC.rnc

```
element phoneNumbers
{
   element title { text },
   element entries
   {
      element entry
      {
         element name
         {
            (attribute gender { text } | attribute age { text })?,
            element first { text },
            element middle { text }?,
            element last { text }
         },
         (element phone { text } | attribute phone { text }),
         element city { text }?
      }*
   }
}
```

Later we see how the text content and attribute values can be specified more accurately using XML Schema datatypes.

Comments

In the Relax NG compact syntax, a comment is indicated by a sharp symbol (#), which defines a comment from that point to the end of the current line.

```
# this is comment in RNC
```

Mixed Content

Mixed content refers to the situation where an element has both text and elements in its content.

With Relax NG mixed content is specified with a pattern represented by the keyword *mixed*. The body of the *mixed* specification describes the elements that can be combined with text to form the mixed content.

Example

DTD Specification

```
<!ELEMENT narrative (#PCDATA | bold | italics | underline)*>
<!ELEMENT bold (#PCDATA>
<!ELEMENT italics (#PCDATA>
<!ELEMENT underline (#PCDATA>
```

Relax NG Specification

```
element narrative
{
    mixed  { ( element bold { text }    |
               element italics { text }  |
               element underline { text } )*
             }
}
```

The following XML document can be validated relative to the previous Relax NG definition.

```
<?xml version="1.0"?>
<narrative>
    Teach a child to be <italics>polite</italics> and
    <italics>courteous</italics> in the home and, when
    he grows up, he'll <bold>never</bold> be able to
    merge his car onto a <underline>freeway</underline>.
</narrative>
```

Interleaving

As an alternative to sequencing (,) and choice (|), Relax NG allows a combination of elements in which any order is permitted for the elements. This interleaving is specified by an ampersand (&).

For example, suppose that in the XML document for the phone number entries, we do not care about the order of the *name*, *phone*, and *city* elements inside each of the *entry* elements.

The Relax NG definition on the next page shows the changes needed to permit these elements to appear in any order.

File: phoneI.rnc

```
element phoneNumbers
{
    element title { text },
    element entries
    {
        element entry
        {
            element name
            {
                attribute gender { text }?,
                element first { text },
                element middle { text }?,
                element last { text }
            } &
            element phone { text }  &
            element city { text }?
        }*
    }
}
```

A special case of interleaving occurs when text is interspersed with elements in mixed content.

In fact, the pattern for *mixed* can be defined using interleaving.

mixed { pat } is equivalent to *text* & pat?

The Relax NG specification for the *narrative* element can be written as shown below.

File: narrative.rnc

```
element narrative
{
     text  &  (element bold { text }  |
          element italics { text }  |  element underline { text } )*
}
```

To justify the term "interleave" for this combining operation, observe the following definition.

File: interleave.rnc

```
element root
{
  element item { text } &
  ( element first { text },
    element second { text },
    element third { text } )
}
```

The elements *first*, *second*, and *third* must appear in that order, but the element *item* may be placed at any of the four gaps in that sequence.

Enumerations

The values of an element or an attribute may be constrained to a particular set of values by defining an enumeration using strings and the choice operations (|).

Examples

```
attribute gender { "female" | "male" }?,

element city { "Iowa City" | "Coralville" |
               "North Liberty" | "Hills" |
               "Solon" }

element year { "2003" | "2004" | "2005" | "2006" }
```

This mechanism can be used to define constant (fixed) attributes and elements.

```
attribute gender { "female" }?,
element city { "Iowa City" }
```

Note on String Matching

A literal string pattern will consider the string in the pattern to match the string in the XML document if the strings are the same after the whitespace in both strings is normalized.

A normalized string has each sequence of interior whitespace replaced by a single space and has all leading and trailing whitespace trimmed.

The following elements and attributes are valid with respect to the Relax NG specifications shown above.

```
<city>Iowa    City   </city>

<city> North
       Liberty</city>

<name gender="            female      ">
```


This normalized string matching can be circumvented by using a built-in datatype with the keyword *string*.

attribute gender { *string* "female" | *string* "male" }?,

Now the values for the *gender* attribute must be exactly "female" or "male".

A companion datatype with the keyword provides the default behavior that we saw above.

attribute gender { *token* "female" | *token* "male" }?,

This specification matches the attribute shown below:

<name gender=" female">

Lists

The list pattern matches a sequence of tokens separated by white space.

Example

element stateCodes { *list* { *text* } }

specifies an XML instance with the element

<stateCodes>
IA IL IN MN MI
NY NJ NM
</stateCodes>

Later we see how XML Schema simple types can be used in Relax NG. Here is a specification that allows lists of even length of floating-point numbers

element evenVec { *list* { (xsd:double, xsd:double)* } }

XML Schema Datatypes

All predefined simple types from XML Schema are recognized by Relax NG.

The prefix "xsd" is automatically linked to the XML Schema datatypes.

```
element number { xsd:integer }
```

To use a different prefix, define it with the keyword *datatypes*.

```
datatypes xs=
            "http://www.w3.org/2001/XMLSchema-datatypes"
element number { xs:integer }
```

Restriction

If the children of an element or the value of an attribute matches a datatype pattern, then the complete content of the element or attribute value must match that datatype pattern.

Illegal Elements

```
element bad1
{
    xsd:integer,
    element note { text }
}
```

```
element bad2
{
    xsd:integer,
    text
}
```

A Legal Element

```
element good
{
    xsd:integer,
    attribute note { text }
}
```

Facets

An XML Schema datatype may be modified by parameters given by the facets of the simple types.

Facets are called "parameters" in Relax NG.

Two facets are disallowed in Relax NG:

whitespace and *enumeration*.

The values of the parameters (facets) must be delimited by quotes or apostrophes.

Examples

```
element age { xsd:integer { minInclusive="0"
                            maxInclusive="120" } }

element currency { xsd:decimal { fractionDigits="2" } }

element password { xsd:string { minLength="8"
                                maxLength="12" } }

element zipcode { xsd:string { pattern="[0-9]{5}" } }

element license { xsd:string { pattern="[A-Z]{3} \d{3}" } }

element tla { xsd:string { pattern="[A-Z][A-Z][A-Z]"  } }

element pm { xsd:time { minInclusive="12:00:00"
                        maxInclusive="23:59:59" } }
```

With these additional tools to specify the contents of elements, we can define a Relax NG schema for the XML document *phoneX.xml* with both the *gender* and *areaCode* attributes.

Some elements are constrained by facet patterns and some using enumerations.

File: phoneX.rnc

```
datatypes xs=
            "http://www.w3.org/2001/XMLSchema-datatypes"
start = element phoneNumbers
        {
            element title { text },
            element entries
            {
                element entry { name, phone, city? }*
            }
        }
name = element name
         {
             attribute gender { "female" | "male" }?,
             element first { text },
             element middle { text }?,
             element last { text }
         }
phone = element phone
          {
              attribute areaCode
                          { xs:integer { pattern="\d{3}" } }?,
              xsd:string { pattern="\d{3}-\d{4}" }
          }
city = element city { "Iowa City" | "Coralville" |
                      "North Liberty" | "Hills" | "Solon" }
```

Recursion in Relax NG

When a pattern has a name, it may refer to itself inside of its definition, producing a recursively defined pattern.

References to recursively defined patterns must occur inside an element pattern.

The recursive reference inside of the pattern must be optional so that the recursion is not endless.

In the next example, the elements *bold*, *italics*, and *span* can be nested inside of each other.

File: rec.rnc

```
start = root

root = element root { inline }

inline = ( text | element bold { inline }
                | element italics { inline }
                | element span
                  {
                     attribute style { text }?,
                     inline
                  }
         )*
```

File: rec.xml

```
<?xml version="1.0"?>
<root>
    I have always wished that my <italics>computer would
    be as easy to <bold>use</bold> as my telephone
    </italics>.  <span>My wish has come true.
    <italics>I</italics> no longer <bold>know</bold> how to
    <bold>use</bold> my telephone</span>.
     <italics>Bjarne Stroustrup</italics>
</root>
```

Namespaces in Relax NG

Relax NG schemas can recognize namespaces in XML documents by means of simple declarations using the keyword *namespace*.

These declarations can bind a prefix to a URI or create a default namespace with no prefix.

As always, it is the URI that determines the namespace, not the prefix.

File: phoneNS.xml

This simple version of the phone number database has no attributes and uses a simple type for the *name* element.

```
<?xml version="1.0"?>
<ph:phoneNumbers
            xmlns:ph="http://slonnegr.cs.uiowa.edu/phone">
  <ph:title>Phone Numbers</ph:title>
  <ph:entries>
    <ph:entry>
      <ph:name>Rusty Nail</ph:name>
      <ph:phone>335-0055</ph:phone>
      <ph:city>Iowa City</ph:city>
    </ph:entry>
    <ph:entry>
      <ph:name>Justin Case</ph:name>
      <ph:phone>354-9876</ph:phone>
      <ph:city>Coralvile</ph:city>
    </ph:entry>
  </ph:entries>
</ph:phoneNumbers>
```

In the Relax NG schema that follows we use a different prefix, although we could have used "ph" just as well.

File: phoneNS.rnc

```
namespace xyz = "http://slonnegr.cs.uiowa.edu/phone"

element xyz:phoneNumbers
{
    element xyz:title { text },
    element xyz:entries
    {
        element xyz:entry
        {
            element xyz:name { text },
            element xyz:phone { text },
            element xyz:city { text }?
        }*
    }
}
```

We can as well define the schema using a default namespace, which involves adding the keyword *default*.

File: phoneNSD.rnc

```
default namespace = "http://slonnegr.cs.uiowa.edu/phone"

element phoneNumbers
{
    element title { text },
    element entries
    {
        element entry
        {
            element name { text },
            element phone { text },
            element city { text }?
        }*
    }
}
```

Two Additional Features

A last example will illustrate two additional features of Relax NG, one that allows a specification to be built in separate files and one to supply information in a specification that is not part of the grammar of patterns.

Merging Grammars: *include*

The *include* directive allows one specification to absorb the definitions from another Relax NG grammar into its grammar.

Basic syntax: *include* "otherFile.rnc"

Annotations

Sometimes we want to include information in a Relax NG specification that will not be used for validation.

An annotation can be written inside brackets that occur immediately preceding the construct to be annotated.

The content of an annotation is a fragment of XML consisting of zero or more attributes followed by zero or more elements.

An attribute in an annotation must be qualified with a prefix that has been declared with a non-empty URI.

In the next example, only attributes appear in the annotations.

The information in an annotation is used primarily as documentation for the (human) readers of the document.

The example that illustrates these two features is a Relax NG specification corresponding to the *product.dtd* schema from the DTD chapter. We begin by repeating that schema.

Example: product.dtd

Describe a catalog of tools to be sold by some company.

```
<!ELEMENT catalog (product+)>

<!ELEMENT product
                (specifications+, options?, price+, notes?)>
  <!ATTLIST product name CDATA #REQUIRED>
  <!ATTLIST product category
                (HandTool|Table|ShopPro) "HandTool">
  <!ATTLIST product partnum NMTOKEN #REQUIRED>
  <!ATTLIST product plant
                (Boston|Buffalo|Chicago) "Chicago">
  <!ATTLIST product inventory
                (InStock|BackOrdered|Discontinued) "InStock">

<!ELEMENT specifications (#PCDATA)>
  <!ATTLIST specifications weight CDATA #IMPLIED>
  <!ATTLIST specifications power NMTOKEN #IMPLIED>

<!ELEMENT options EMPTY>
  <!ATTLIST options finish (Metal|Polished|Matte) "Matte">
  <!ATTLIST options adapter
                (Included|Optional|NotApplicable) "Included">
  <!ATTLIST options case
                (HardShell|Soft|NotApplicable) "HardShell">

<!ELEMENT price (#PCDATA)>
  <!ATTLIST price msrp CDATA #IMPLIED>
  <!ATTLIST price wholesale CDATA #IMPLIED>
  <!ATTLIST price street CDATA #IMPLIED>
  <!ATTLIST price shipping CDATA #IMPLIED>

<!ELEMENT notes (#PCDATA)>
```

Relax NG Version

One difference with this new version is that the pattern syntax of Relax NG has no way to specify default values for attributes. The best we can do is supply these values in annotations that are attached to the attributes in question.

The annotation attribute *defaultValue* defines a default value for the attributes that require one.

Remember that this attribute must have a prefix.

We use the namespace "http://uxt.examples/annotations" to define the prefix.

The second new feature in this example is illustrated by dividing the specification into two files and using the *include* directive to combine them.

File: product.rnc

```
namespace a = "http://uxt.examples/annotations"

include "pdefs.rnc"

start = element catalog { product+ }

product =
      element product
      {
         attribute name { text },

         [ a:defaultValue = "HandTool" ]
         attribute category
         { "HandTool" | "Table" | "ShopPro" }?,

         attribute partnum { xsd:NMTOKEN },

         [ a:defaultValue = "Chicago" ]
         attribute plant { "Boston" | "Buffalo" | "Chicago" }?,
```

```
        [ a:defaultValue = "InStock" ]
        attribute inventory
        { "InStock" | "BackOrdered" | "Discontinued" }?,
        specifications+,
        options?,
        price+,
        element notes { text }?
    }
```

File: pdefs.rnc

```
namespace a = "http://uxt.examples/annotations"
specifications =
        element specifications
        {
            attribute weight { text }?,
            attribute power { xsd:NMTOKEN }?,
            text
        }

options =
        element options
        {
            [ a:defaultValue = "Matte" ]
            attribute finish { "Metal" | "Polished" | "Matte" }?,

            [ a:defaultValue = "Included" ]
            attribute adapter
            { "Included" | "Optional" | "NotApplicable" }?,

            [ a:defaultValue = "HardShell" ]
            attribute case
            { "HardShell" | "Soft" | "NotApplicable" }?
        }
```

```
price = element price
        {
           attribute msrp { text }?,
           attribute wholesale { text }?,
           attribute street { text }?,
           attribute shipping { text }?,
           text
        }
```

Notes on the Example

The full power of Relax NG data types has not been used in this example.

For compatibility, the types from the DTD specification have been translated mechanically. A careful analysis of the original XML document might lead to a more precise specification of the types of elements and attributes in this example.

The XML document *products.xml* from the DTD chapter validates with respect to this Relax NG specification, as shown by the following application of *jing*.

```
% java -jar jing.jar -c products.rnc products.xml
%
```

Keywords in Compact Relax NG

attribute

default

datatypes

div

element

empty

external

grammar

include

inherit

list

mixed

namespace

notAllowed

parent

start

string

text

token