Résumé Relax-NG

Relax NG

- Espace de noms : `http://relaxng.org/ns/structure/1.0`
- Librairie de type XML-Schema : `http://www.w3.org/2001/XMLSchema-datatypes`
- Syntaxe compacte et syntaxe XML

Construction

- Appel :

```
<element name="a">
 <text />
</element>
```

```
<attribute name="a">
 <text />
</attribute>
```

- Occurences :

```
<zeroOrMore>, <oneOrMore>,
<optional>
```

- Combinaisons :

```
<group>, <choice>,
<interleave>, <mixed>
```

- Appel :

```
element a {
 text
}
```

```
attribute a {
 text
}
```

- Occurences : `* + ?`
- Combinaisons : `, | & mixed`

Utilisation de `(` et `)` pour combinaisons complexes.

Patterns

- Centralisation des définitions
- Permet la récursion

```
<define name="elt-def">
 <element name="elt">
  <attribute name="att" />
  <text />
 </element>
</define>

<element name="parent">
 <attribute name="att2" />
 <ref name="elt-def" />
</element>
```

```
elt-def = element elt {
    attribute att { text },
    text
}

element parent {
 attribute att2 { text },
 elt-def
}
```

- La référence peur contenir n'importe quelle combinaison de patterns, y compris un appel à elle-même.

Élément racine

```
<grammar>
 <define name="pattern1">
  ...
 </define>
 ...
 <start>
  <element name="racine">
   ...
  </element>
 </start>
</grammar>
```

```
pattern1 = ...
pattern2 = ...
...

start = element racine …
```


Types simples

- Les seuls types natifs dans Relax NG sont les types `string` et `token`.
- Pour avoir plus de types, il faut faire appel à une bibliothèque de types ; par exemple, celle de XML Schema. E.g.

```
<element name="elt"
    xmlsn="http://relaxng.org/ns/structure/1.0"
    datatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes">
 <data type="nonNegativeInteger">
</element>
```

Ou

```
datatypes xsd="http://www.w3.org/2001/XMLSchema-datatypes"
element elt { xsd:nonNegativeInteger }
```


Librairie XMLSchema

- Chaines de caractères : en plus des type Relax NG,
  - `NMTOKEN` : comme pour les DTDs, pas d'espaces ni de ponctuation
  - `ID`, `IDREF`, `IDREFS` : comme pour les DTDs
  - `language` : une langue dans le code standard (ex: `en`, `en-US`, `fr`, `it`...)
- URIs :
  - `anyURI` : une adresse internet, un fichier, … Attention aux caractères accentués.
- Nombres et booléens :
  - `boolean` : true, false, 0 or 1
  - `decimal`, `integer`, `nonPositiveInteger`, `PositiveInteger`, `nonNegativeInteger`, `NegativeInteger`, `int` (32 bits), `short` (16 bits), `byte` (8 bits), …
- Date et heure :
  - `dateTime` : CCYY-MM-DDThh:mm:ss(timezone)
  - `date` : CCYY-MM-DD(timezone)
  - `time` : hh:mm:ss(timezone). Ex : 15:20:00 ou 18:00:15+02:00
  - `gYear` (ex. 2001), …

Types complexes

- Valeur fixée :

```
<element name=”isbn”>
<value>012345567</value>
</element>
```

ou

```
element isbn { “01234567” }
```

- Énumération :

```
attribute état { “présent” |
“absent” | “ne sais pas” }
```

ou

```
<attribute name=”état”>
  <choice>
    <value>présent</value>
    <value>absent</value>
    <value>ne sais pas</value>
  </choice>
</attribute>
```

- Avec type :

```
attribute nombres {xsd:integer “1”}
```

ou

```
<attribute name=”nombre”
datatypeLibrary=”http://www.w3.org/
2001/XMLSchema-datatypes”>
  <value type=”int”>1</value>
</attribute>
```

- Liste :

```
element valeurs { list
{xsd:boolean, xsd:boolean,
xsd:boolean} }
```

ou

```
<element name=”valeurs”>
  <list>
    <data type=”boolean” />
    <data type=”boolean” />
    <data type=”boolean” />
  </list>
</element>
```


Facettes

- `length`, `maxLenth`, `minLength` : longueur de chaine de caractères
- `maxExclusive`, `minExclusive`, `maxInclusive`, `minInclusive`
- `totalDigits` : s'applique à `decimal`, `integer`
- `pattern` : expression régulière sur la chaine de caractères donnée.

```
<element name=”elt”>
 <data type=”decimal”>
  <param name=”maxInclusive”>56.7</param>
 </data>
</element>

element elt {
  xsd:decimal { maxInclusive=”56.7” }
}
```


Espaces de noms

- Espace de nom par défault :

```
<element ns=”espace-de-noms” name=”elt”>
 ...
</element>

default namespace = “espace-de-noms”
element elt {
 ...
}
```

Attention : comme d'habitude, les attributs ne prennent pas l'espace de nom par défaut.

- Sinon : avec `xmlns:...`

```
<element xmlns:pre=”espace-de-noms” name=”pre:elt”>
 ...
</element>

namespace pre = “espace-de-noms”
element pre:elt {
 ...
}
```


Exemple

ville.xml

```
<ville xmlns:g="http://www.w3.org/2003/01/geo/wgs84_pos#">
  <nom>Grenoble</nom>
  <g:lat>45.196349</g:lat>
  <g:long>5.73226</g:long>
</ville>
```

ville.rnc

```
namespace geo = “http://www.w3.org/2003/01/geo/wgs84_pos#”

element ville {
  element nom
     { xsd:token },
  element geo:lat
     { xsd:decimal },
  element geo:long
     { xsd:decimal }
}
```

ville.rng

```
<?xml version="1.0" encoding="UTF-8"?>
<element name="ville"
    xmlns:geo="http://www.w3.org/2003/01/geo/wgs84_pos#"
    xmlns="http://relaxng.org/ns/structure/1.0"
    datatypeLibrary="http://www.w3.org/2001/XMLSchema-
datatypes">
  <element name="nom">
    <data type="token"/>
  </element>
  <element name="geo:lat">
    <data type="decimal"/>
  </element>
  <element name="geo:long">
    <data type="decimal"/>
  </element>
</element>
```


Annotations

- Syntaxe XML : simplement l'utilisation d'un espace de nom autre que celui de Relax NG (voire pas d'espace de nom du tout)

```
<grammar xmlns=”http://relaxng.org/ns/structure/1.0"
         xmlns:dc=”http://purl.org/dc/elements/1.1/”>
  <dc:creator>Merlin</dc:creator>
  <dc:description>Un Schéma magique !</dc:description>
  <start>
    <element name=”elt”>
      <text />
    </element>
  </start>
</grammar>
```

- Syntaxe compacte :

```
default namespace = “http://relaxng.org/ns/structure/1.0"
namespace dc = ”http://purl.org/dc/elements/1.1/”

dc:creator [ “Merlin” ]
dc:description [ “Un Schéma magique !” ]
start = element elt { text }
```


Documentation d'element

- Syntaxe XML :

```
<grammar xmlns=”http://relaxng.org/ns/structure/1.0"
         xmlns:dc=”http://purl.org/dc/elements/1.1/”>
  <dc:creator>Merlin</dc:creator>
  <start>
  <element name=”elt”>
    <dc:description>
        Ceci est la racine
    </dc:description>
    <text />
  </element>
  </start>
</grammar>
```

- Syntaxe compacte :

```
default namespace = “http://relaxng.org/ns/structure/1.0"
namespace dc = ”http://purl.org/dc/elements/1.1/”

dc:creator [ “Merlin” ]
start = (
  [ dc:description [ “Ceci est la racine !” ] ]
  element elt {
    text
  }
)
```