

Validation of Character Repertoires for XML Documents

Erik Wilde
Computer Engineering and Networks Laboratory
ETH Zürich

<http://dret.net/projects/crvx/>

Abstract

XML is based on Unicode, and therefore XML documents may use the full Unicode character repertoire. However, XML-based applications often use XML interfaces to legacy software which in many cases is not capable of dealing with the full Unicode character repertoire. We therefore propose a schema language for XML which is capable of limiting the character repertoire of XML documents.

This schema language, called *Character Repertoire Validation for XML (CRVX)*, has features to permit or disallow character repertoire subsets from certain parts of an XML document, for example only for element and attribute names. CRVX uses information from the *Unicode Character Database (UCD)* to make character repertoire specification as easy as possible.

CRVX is not intended to be the only schema language in an XML application scenario, but it provides useful additional schema-based validation to protect applications from unsupported characters. XML applications typically combine different schema languages before processing XML documents, and CRVX is intended to complement other schema languages such as grammar-based languages (DTD, XML Schema) or rule-based languages (Schematron).

CRVX can be implemented in various ways. One simple solution is to use XSLT to transform an CRVX schema into an XSLT program, which is then used to validate XML documents. We briefly describe such an implementation. Other (and more efficient) implementations could be based on DOM or SAX parsers.

XML is Good!

- universally accepted data structuring
 - published at the right time (Internet hype)
 - published by the right people (no OS vendor)
- simple mechanism
 - basically, HTML with DIY elements/attributes
 - users can define their own vocabularies
 - large set of layered technologies
 - XML Schema, XSLT, XQuery, SOAP, ...
- defined on the character level
 - e.g., STag ::= '<' Name (S Attribute)* S? '>'
 - encoding is application-dependent
 - UTF-8 & UTF-16 must be supported everywhere

XML and Characters

```

<文書 改訂日付="1999年3月1日">
  <題>サンプル</題>
  <段落>これはサンプル文書です。</段落>
  <!-- コメント -->
  <段落>&会社名;</段落>
  <図面 図面実体名="サンプル"/>
</文書>

```

Outline

- Motivation
- XML Schema Languages
- CRVX Schema Language
 - Restrictions
 - Structures
 - Contexts
- Implementation
- What now?
- Q&A

Motivation

- XML is often used as transport format only
 - XML data is generated from existing software
 - XML is shipped over the net (maybe in SOAP)
 - data is extracted from XML and fed into software
- Unicode often is not fully supported
 - no problem for XML generation
 - but software should be protected from Unicode
 - block and filter XML containing unsupported characters
- programming is expensive and error-prone
- provide a declarative way for standard tasks
 - easy to author, understand and modify
 - can be portable if based on portable components

XML Schema Languages

- XML documents must be well-formed
 - respecting the syntax and additional constraints
- XML documents may be valid
 - also respecting the validity constraints
 - based on the DTD schema language
- XML documents may be schema-valid
 - based on the W3C XML Schema language
 - complex and heavy-weight language
 - supports regular expressions for simple types
 - Unicode support built into the regular expressions
- schema languages are constraint languages
 - they can be arbitrarily simple or complex

Problems with XML Schema

- hard to learn
 - introduces a type system with type derivation
 - even harder to use in a smart way
- hard to implement
 - very few (if any) correct implementations exist
 - processing time may be an issue
- *mixed content* has no type
 - hard to integrate into the existing type system
 - the only type-free character data in XML Schema
- *markup* has no type
 - no way to restrict element or attribute names
 - traditional separation of *markup* and *character data*

Modular Schema Languages

- design a small and specialized language
 - makes only one thing
 - can be easily implemented
 - will often be combined with other languages
- *Document Schema Definition Languages*
 - interesting but lacking support
 - framework for combining schema languages
 - different schema languages for different areas
 - RELAX NG, Schematron, XML Schema datatypes, ...
 - more DSDL information at <http://dSDL.org>
- *Character Validation for XML Documents*
 - CRVX as yet another part of the DSDL framework

CRVX Overview

- a simple schema language for a simple task
 - restricting the characters used in XML documents
- each CRVX schema is a set of restrictions
 - that's all there is!
- two orthogonal concepts are supported
 - structures refer to structural components of XML
 - elements, attributes, comments, ...
 - contexts refer to the hierarchical nature of XML
 - everything inside the third TABLE element
- both concepts can be combined
 - all comments inside the third TABLE element

Simple CRVX Examples

```
<crvx structures="namespaceXML" version="1.0"
  xmlns="http://dret.net/xmlns/crvx10">
  <restrict charrep="\p{IsBasicLatin} \p{IsLatin-1Supplement}"/>
</crvx>
```

```
<crvx structures="namespaceXML" version="1.0"
  xmlns="http://dret.net/xmlns/crvx10">
  <restrict structure="elementLocalName attributeLocalName" maxLength="8"/>
</crvx>
```

Restrictions

- restrictions are the core of CRVX
 - everything else is only there to support them
- restrictions restrict the character repertoire
 - using a small subset of XML Schema regexes
 - character class expressions: [a-z]
 - category escapes: \p{L1}
 - lists are allowed: [a-z] [A-Z]
- they may also restrict lengths
 - minLength and maxLength may be specified
- restrictions can be limited
 - based on structures and/or contexts

CRVX Restriction Example

```
<crvx structures="namespaceXML" version="1.0"
  xmlns="http://dret.net/xmlns/crvx10">
  <restrict structure="elementLocalName attributeLocalName PITarget"
    charrep="\p{IsBasicLatin}"/>
  <restrict structure="elementContent"
    charrep="\p{IsBasicLatin} \p{IsLatin-1Supplement}"/>
  <restrict structure="PITarget" minlength="3" maxlength="3"/>
</crvx>
```

Structures

- structures identify XML's structural nature
- XML is viewed differently
 - XML only has an implicit information model
 - XML Infoset: 11 information item types
 - XPath: 7 node types
- not every XML document has an Infoset
 - must be namespace-compliant
- CRVX supports two XML information models
 - pureXML and namespaceXML
 - different handling of QName's such as `html:h1`

CRVX Structure Example

```
<crvx structures="namespaceXML" version="1.0"
  xmlns="http://dret.net/xmlns/crvx10">
  <restrict structure="elementLocalName attributeLocalName PITarget"
    charrep="\p{IsBasicLatin}"/>
  <restrict structure="elementContent"
    charrep="\p{IsBasicLatin} \p{IsLatin-1Supplement}"/>
  <restrict structure="PITarget" minlength="3" maxlength="3"/>
</crvx>
```

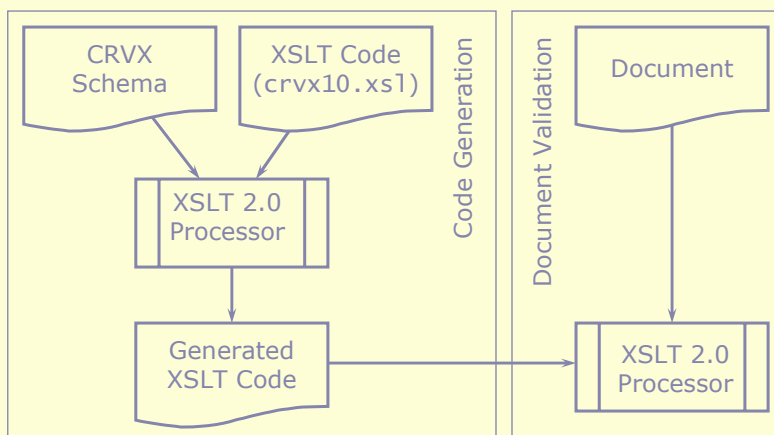
Contexts

- contexts are defined by XSLT patterns
 - consequently only applicable for namespace XML
 - identify arbitrary subtrees of the document
- contexts may be nested
 - they can contain other contexts
 - or restrictions (which apply to the context only)
- contexts may be named
 - names can be referred to in contexts
 - references must be acyclic
 - names can be referred to in restrictions
 - allows one restriction for several contexts

CRVX Context Example

```
<crvx structures="namespaceXML" version="1.0"
  xmlns="http://dret.net/xmlns/crvx10">
  <context path="figure/caption">
    <restrict charrep="\p{IsBasicLatin} \p{IsLatin-1Supplement}"/>
    <context path="link">
      <restrict structure="elementContent" maxLength="10"/>
    </context>
  </context>
</crvx>
```

Implementation



What now?

- CRVX improvements/extensions
 - handle character normalization requirements
 - handle the referencing of characters
 - literal vs. character references (i.e., ü vs. ü)
 - adopt XPath 2.0
 - design standardized error reporting
 - DTD vs. XML Schema approach
- XML processing should be modular
 - advertise the concept of small processing modules
 - CRVX as one part of a processing pipeline
- XML-processor based implementation
 - XSLT has built-in limitations

IUC24 Validation of Character Repertoires for XML Documents (©2003 Erik Wilde) 19

Thank You! — Q&A

Project Page: <http://dret.net/projects/xscs/>

Paper: <http://dret.net/netdret/publications#wi103h>