Extreme Markup Languages 2003

Montréal, Québec
August 4-8, 2003

The difficulty of schema conformance problems

Robert C. Lyons
Unidex, Inc.

Abstract

This paper examines the difficulty of the schema conformance problem for the following XML schema languages:

- DTD
- RELAX NG with W3C XML Schema Datatypes
- Schematron
- W3C XML Schema
- Namespace Routing Language

The schema conformance problem is defined as follows: Given a schema, is there an XML document that conforms to the schema?

For these five XML schema languages, the schema conformance problem is intractable (ranging from NP-HARD to undecidable).

The difficulty of schema conformance problems

Table of Contents

The difficulty of schema conformance problems

Robert C. Lyons

§ Introduction

This paper examines the difficulty of the schema conformance problem, which is defined as follows:

> Given a schema, is there an XML [eXtensible Markup Language] document that conforms to the schema?
>
> If there is an XML document that conforms to a given schema, then we say that the schema is **satisfiable**.

For example, the following DTD [Document Type Definition] [ZVON02], [W3C02] is not satisfiable because each ***section*** element must contain one or more ***section*** elements:

```
<!ELEMENT section ( section+ ) >
```

One could argue that this DTD can be satisfied by an infinitely large XML document and that the XML recommendation [W3C02] does not explicitly state that a well-formed XML document must be finite. However, as a practical matter, we don't consider an XML schema to be satisfiable if it can only be satisfied by an infinite XML document.

It would be nice if your favorite XML schema editor could tell you whether or not your schema was satisfiable. Unfortunately, as we'll see, the schema conformance problem is *intractable*[1] for the following XML schema languages:

- DTD
- RELAX NG with W3C XML Schema Datatypes
- Schematron
- W3C XML Schema
- Namespace Routing Language

Thus, for these XML schema languages, it's not feasible to build an XML schema editor that can do either of the following for any given schema:

- Tell you whether or not your schema is satisfiable.
- Generate an XML document that conforms to your schema.

The remainder of this paper examines the difficulty of the schema conformance problem for the XML schema languages listed above. We assume that the reader has some familiarity with these XML schema languages. We don't assume that the reader is familiar with concepts from computational complexity (*e. g.*, NP-HARD, undecidable, etc.). We use footnotes to provide a brief introduction to these terms.

The following is the outline of this paper:

§ DTD conformance problem

The DTD conformance problem can be solved in *linear time*[2] when the DTDs do not contain `ID`s and `IDREF`s. [Arenas et al.], [Fan/Libkin]

For DTDs that declare `IDREF` attributes that have fixed values, determining the satisfiability of the DTD can become a difficult problem.

The following is an example of a DTD that declares `IDREF` attributes that have fixed values:

```
<!-- A nyc-census document specifies the population of the five
     boroughs of New York City. The borough elements may appear
     in any order. There must be one borough element for each
     of the five boroughs. The order independence of the
     boroughs is achieved through the use of ID and IDREF
     attributes.
-->
<!ELEMENT nyc-census ( borough, borough, borough, borough, borough )>

<!ATTLIST nyc-census borough-1 IDREF #FIXED "Bronx">
<!ATTLIST nyc-census borough-2 IDREF #FIXED "Brooklyn">
<!ATTLIST nyc-census borough-3 IDREF #FIXED "Manhattan">
<!ATTLIST nyc-census borough-4 IDREF #FIXED "StatenIsland">
<!ATTLIST nyc-census borough-5 IDREF #FIXED "Queens">

<!ELEMENT borough EMPTY>

<!ATTLIST borough name ID #REQUIRED>
<!ATTLIST borough population CDATA #REQUIRED>
```

This DTD specifies the schema for a class of documents that provide New York City census data. A ***nyc-census*** document specifies the population of the five boroughs of New York City. The ***borough*** elements may appear in any order. There must be one ***borough*** element for each of the five boroughs. The order independence of the boroughs is achieved through the use of `ID` and `IDREF` attributes.

The following is an XML document that conforms to the DTD above:

```
<?xml version="1.0"?>
<!DOCTYPE nyc-census SYSTEM "fixed-idref.dtd">
<nyc-census>
    <borough name="Brooklyn"     population="2465326"/>
    <borough name="Queens"       population="2229379"/>
    <borough name="Manhattan"    population="1537196"/>
    <borough name="Bronx"        population="1332650"/>
    <borough name="StatenIsland" population= "443728"/>
</nyc-census>
```

Note that a simple mistake in the DTD for the nyc-census documents can make the DTD unsatisfiable. For example, the following DTD is unsatisfiable because the ***nyc-census*** element must contain exactly four ***borough*** elements, but the `IDREF` attributes refer to five distinct boroughs:

```
<!-- Warning: the following element declaration
     is incorrect. The nyc-census element should have
     five borough subelements, but only four have
     been declared.
-->
<!ELEMENT nyc-census ( borough, borough, borough, borough )>

<!ATTLIST nyc-census borough-1 IDREF #FIXED "Bronx">
<!ATTLIST nyc-census borough-2 IDREF #FIXED "Brooklyn">
<!ATTLIST nyc-census borough-3 IDREF #FIXED "Manhattan">
<!ATTLIST nyc-census borough-4 IDREF #FIXED "StatenIsland">
<!ATTLIST nyc-census borough-5 IDREF #FIXED "Queens">

<!ELEMENT borough EMPTY>

<!ATTLIST borough name ID #REQUIRED>
<!ATTLIST borough population CDATA #REQUIRED>
```

Note that an XML document is not valid if it contains an `IDREF` attribute value that is not declared by one of the `ID` attributes.

In a posting to the xml-dev mailing list [Thompson], Henry Thompson showed that the DTD conformance problem is *NP-HARD*.[3]

The following is a sketch of the proof that the DTD conformance problem is NP-HARD:[3]

1. An instance of the *3SAT problem*[4] can be encoded as a DTD, such that the DTD is satisfiable if and only if the 3SAT instance has a solution. The encoding of the 3SAT instance can be done in *polynomial time*.[1]
2. If we had an algorithm that solved the DTD conformance problem, then we could use this algorithm (and the algorithm that encodes instances of the 3SAT problem into DTDs) to solve the 3SAT problem. Thus, the DTD conformance problem is at least as hard as the 3SAT problem.
3. Therefore, the DTD conformance problem is NP-HARD[3], since the 3SAT problem is *NP-COMPLETE*.[5]

Consider the following instance of the 3SAT problem:

```
( x1 OR x2 OR x3 ) AND
( x2 OR NOT(x3) OR NOT(x4) ) AND
( x3 OR x4 OR NOT(x1) )
```

The following DTD encodes this instance of the 3SAT problem:

```
<!ELEMENT solution ( variableAssignments, clauses )>

<!ELEMENT variableAssignments ( x1Assignment,
                                x2Assignment,
                                x3Assignment,
                                x4Assignment )>

<!ELEMENT x1Assignment ( x1isTrue | x1isFalse )>
<!ATTLIST x1Assignment value ID #REQUIRED>

<!ELEMENT x2Assignment ( x2isTrue | x2isFalse )>
<!ATTLIST x2Assignment value ID #REQUIRED>

<!ELEMENT x3Assignment ( x3isTrue | x3isFalse )>
<!ATTLIST x3Assignment value ID #REQUIRED>

<!ELEMENT x4Assignment ( x4isTrue | x4isFalse )>
<!ATTLIST x4Assignment value ID #REQUIRED>

<!ELEMENT clauses ( ( x1isTrue  | x2isTrue  | x3isTrue  ),
                    ( x2isTrue  | x3isFalse | x4isFalse ),
                    ( x1isFalse | x3isTrue  | x4isTrue  ) )>

<!ELEMENT x1isTrue  EMPTY>
<!ATTLIST x1isTrue  value IDREF #FIXED "x1isTrue">
<!ELEMENT x1isFalse EMPTY>
<!ATTLIST x1isFalse value IDREF #FIXED "x1isFalse">
<!ELEMENT x2isTrue  EMPTY>
<!ATTLIST x2isTrue  value IDREF #FIXED "x2isTrue">
<!ELEMENT x2isFalse EMPTY>
<!ATTLIST x2isFalse value IDREF #FIXED "x2isFalse">
<!ELEMENT x3isTrue  EMPTY>
<!ATTLIST x3isTrue  value IDREF #FIXED "x3isTrue">
<!ELEMENT x3isFalse EMPTY>
<!ATTLIST x3isFalse value IDREF #FIXED "x3isFalse">
<!ELEMENT x4isTrue  EMPTY>
<!ATTLIST x4isTrue  value IDREF #FIXED "x4isTrue">
<!ELEMENT x4isFalse EMPTY>
<!ATTLIST x4isFalse value IDREF #FIXED "x4isFalse">
```

The DTD is satisfiable if and only if the 3SAT instance has a solution. The following XML document conforms to this DTD and solves the instance of the 3SAT problem:

```
<?xml version="1.0"?>
<!DOCTYPE solution SYSTEM "3sat-01.dtd">
<solution>
    <variableAssignments>
        <x1Assignment value="x1isTrue">
            <x1isTrue/>
        </x1Assignment>
        <x2Assignment value="x2isFalse">
            <x2isFalse/>
        </x2Assignment>
        <x3Assignment value="x3isFalse">
            <x3isFalse/>
        </x3Assignment>
        <x4Assignment value="x4isTrue">
            <x4isTrue/>
        </x4Assignment>
    </variableAssignments>

    <clauses>
        <x1isTrue/>
        <x3isFalse/>
        <x4isTrue/>
    </clauses>
</solution>
```

The following XML document does not conform with the DTD and does not solve the instance of the 3SAT problem:

```
<?xml version="1.0"?>
<!DOCTYPE solution SYSTEM "3sat-01.dtd">
<solution>
    <variableAssignments>
        <x1Assignment value="x1isTrue">
            <x1isTrue/>
        </x1Assignment>
        <x2Assignment value="x2isTrue">
            <x2isTrue/>
        </x2Assignment>
        <x3Assignment value="x3isFalse">
            <x3isFalse/>
        </x3Assignment>
        <x4Assignment value="x4isFalse">
            <x4isFalse/>
        </x4Assignment>
    </variableAssignments>

    <clauses>
        <x1isTrue/>
        <x3isFalse/>
        <x4isTrue/>
    </clauses>
</solution>
```

In summary, the DTD conformance problem is solvable in linear time when the DTDs do not contain `ID` and `IDREF` attributes. [Arenas et al.], [Fan/Libkin] The `ID/IDREF` feature of DTDs is very useful, but the downside of this feature is that it makes the DTD conformance problem NP-HARD. [Thompson]

§ RELAX NG conformance problem

Next we examine the schema conformance problem for RELAX NG [Regular Language description for XML Next Generation] [OASIS01], [ThaiOpen], [Cover02], which is an elegant and powerful schema language that was developed by James Clark, MURATA Makoto, and other members of the RELAX NG technical committee. At the Extreme Markup Languages 2003 conference, MURATA Makoto offered a proof that the RELAX NG conformance problem is solvable in linear time[2] (when the RELAX NG schemas use only the built-in datatypes).

For RELAX NG schemas that use the W3C XML Schema datatypes [W3C01], [Cover03], [xFront], determining the satisfiability of the schema can become a difficult problem. A datatype in such schemas may specify multiple constraints (*e.g.*, a regular expression, a maximum length, etc.), all of which must be satisfied. Specifically, the RELAX NG ***data*** element, which specifies a datatype, may contain multiple ***param*** elements, which are ANDed together; each ***param*** element specifies a constraint (*e.g.*, a regular expression). The ability to specify multiple ***param*** elements within a ***data*** element is very useful; however, it's possible to specify two ***param*** elements that are mutually exclusive.

The following is an example of a RELAX NG schema in which the ***data*** element contains two conflicting ***param*** elements:

```
<?xml version="1.0" encoding="UTF-8"?>
<element name="conflicted"
         xmlns="http://relaxng.org/ns/structure/1.0"
         datatypeLibrary=
           "http://www.w3.org/2001/XMLSchema-datatypes">
```

```
    <data type="string">
        <param name="pattern">(aa)+</param>
        <param name="pattern">a(aa)+</param>
    </data>
</element>
```

The first ***param*** element specifies that the ***conflicted*** element must contain an even number of “a” characters. The second ***param*** element specifies that the ***conflicted*** element must contain an odd number of “a” characters. There are no XML documents that conform to this schema because the content of the ***conflicted*** element cannot match both patterns.

For ***data*** elements that contain multiple ***param*** elements such that each ***param*** element specifies a complex regular expression, it can be very difficult to determine if the datatype is satisfiable.

It turns out that the RELAX NG conformance problem is NP-HARD[3] when the schemas are allowed to use W3C XML Schema Datatypes. [W3C01], [Cover03], [xFront] The following is a brief sketch of the proof:

1. An instance of the 3SAT problem[4] can be encoded as a RELAX NG schema, which uses W3C XML Schema Datatypes, such that the schema is satisfiable if and only if the 3SAT instance has a solution. The encoding of the 3SAT instance can be done in polynomial time.[1]
2. If we had an algorithm that solved the RELAX NG conformance problem, then we could use this algorithm (and the algorithm that encodes instances of the 3SAT problem into RELAX NG schemas) to solve the 3SAT problem. Thus, the RELAX NG conformance problem is at least as hard as the 3SAT problem, which is NP-COMPLETE.[5]
3. Therefore, the RELAX NG conformance problem is NP-HARD[3] when the schemas are allowed to contain W3C XML Schema Datatypes.

Let’s look at a RELAX NG schema that encodes the following instance of the 3SAT problem:

```
( x1 OR x2 OR x3 ) AND
( x2 OR NOT(x3) OR NOT(x4) ) AND
( x3 OR x4 OR NOT(x1) )
```

The following RELAX NG schema encodes this 3SAT instance. Note that each of the four variables (*i.e.*, x1, x2, x3, and x4) must be equal to either “0” (*i.e.*, FALSE) or “1” (*i.e.*, TRUE).

```
<?xml version="1.0" encoding="UTF-8"?>
<element name="three_sat_solution"
         xmlns="http://relaxng.org/ns/structure/1.0"
         datatypeLibrary=
           "http://www.w3.org/2001/XMLSchema-datatypes">
    <data type="string">
        <param name="pattern">x1=[01],x2=[01],x3=[01],x4=[01]</param>
        <param name="pattern">(.*x1=1.*)|(.*x2=1.*)|(.*x3=1.*)</param>
        <param name="pattern">(.*x2=1.*)|(.*x3=0.*)|(.*x4=0.*)</param>
        <param name="pattern">(.*x3=1.*)|(.*x4=1.*)|(.*x1=0.*)</param>
    </data>
</element>
```

This RELAX NG schema is satisfiable if and only if the 3SAT instance has a solution. The following XML document conforms to the RELAX NG schema and solves the instance of the 3SAT problem:

```
<?xml version="1.0" encoding="UTF-8"?>
<three_sat_solution>x1=1,x2=0,x3=0,x4=1</three_sat_solution>
```

The following XML document does not conform with the RELAX NG schema and does not solve the instance of the 3SAT problem:

```
<?xml version="1.0" encoding="UTF-8"?>
<three_sat_solution>x1=1,x2=1,x3=0,x4=0</three_sat_solution>
```

In summary, the RELAX NG conformance problem is solvable in linear time[2] when the schemas use only the built-in datatypes. The RELAX NG conformance problem is NP-HARD[3] when the schemas are allowed to use W3C XML Schema Datatypes (or other datatype libraries that support regular expressions).

§ Schematron conformance problem

When I first looked at the schema conformance problem for DTDs and RELAX NG with W3C XML Schema datatypes, I didn't suspect that these problems are intractable.[1] On the other hand, when I first looked at the Schematron conformance problem, it was immediately clear to me that the problem is difficult, if not intractable. Schematron [Schematron], [Cover01], [SourceForge], [ZVON01] schemas define validity constraints using XPath 1.0 expressions [W3C03], and it can be very difficult to determine if there exists an XML document that satisfies a set of XPath expressions. For example, it's fairly straightforward to encode an instance of the 3SAT problem as a Schematron schema, such that the 3SAT instance is solvable if and only if the schema is satisfiable.

It turns out that the Schematron conformance problem is so difficult that it is *undecidable*[6] (*i.e.*, unsolvable). In the article entitled *The Undecidability of the Schematron Conformance Problem* [Lyons], the author used the Post Correspondence Problem [Post01], [PCP], [Post02], [Post03], [Lyons] to prove that Schematron conformance problem is undecidable.[6] This is true even when the problem is restricted to Schematron schemas that do not use the `document` and `key` functions of XPath.

The undecidability of the Schematron conformance problem can also be proved more easily using the undecidability of *Hilbert's Tenth Problem*.[7]

The following is a sketch of the proof that the Schematron conformance problem is undecidable:[6]

1. We assume that the Schematron conformance problem is decidable and then show that this assumption leads to a contradiction. In other words, we assume that there is an algorithm that solves the problem.
2. Next we extend this algorithm into a new algorithm that solves Hilbert's Tenth Problem.[7]
3. The input to the new algorithm is an instance of Hilbert's Tenth problem (*i.e.*, a Diophantine equation[7]).
4. The following is the new algorithm that solves Hilbert's Tenth Problem:
    1. We transform the Diophantine equation into a Schematron schema, such that the schema is satisfiable if and only if there is a solution (among the integers) to the Diophantine equation. The transformation is straightforward since XPath 1.0 includes arithmetic and relational operators (*e.g.*, "+", "-", "*", "=", etc.). XPath 1.0 does not include an exponentiation operator; however, this is not a problem, since the exponents in Diophantine equations are whole numbers. For example, the equation $\mathbf{(x + y)^2 = 4}$ is equivalent to $\mathbf{(x + y)(x + y) = 4}$.
    2. We use the algorithm that solves the Schematron conformance problem to determine whether or not there are any XML documents that match our Schematron schema. If there is an XML document that matches our Schematron schema, then our new algorithm responds with "Yes" (*i.e.*, yes, there is a solution to the instance of Hilbert's Tenth Problem). If there are no XML documents that match our Schematron schema, then our new algorithm responds with "No" (*i.e.*, no, there is no solution to the instance of Hilbert's Tenth Problem).
5. We now have an algorithm that solves Hilbert's Tenth Problem. However, Hilbert's Tenth Problem is unsolvable. [Wikipedia06] Thus, our initial assumption (that there is an algorithm for solving the Schematron conformance problem) must be false. Therefore, the Schematron conformance problem is undecidable.[6]

The fact that the Schematron conformance problem is undecidable[6] doesn't mean that you can't devise a **specific** algorithm that determines if there is an XML document that conforms to a **particular** schema. The undecidability of the problem does mean that it is impossible to devise a **general** algorithm that can make this determination for **any** Schematron schema.

Let's look at a Schematron schema that encodes the following Diophantine equation:[7]

$$(2x^3 - 5y^2)(z^2 + 1) + 20 = 0$$

The following Schematron schema encodes this Diophantine equation:

```
<?xml version="1.0"?>
<schema xmlns="http://www.ascc.net/xml/schematron">
```

```
  <!-- This schematron schema verifies that an XML document
       contains a solution to the following Diophantine equation:

           (2x^3 - 5y^2)(z^2 + 1) + 20 = 0

  -->

  <pattern name="check-solution">

    <rule context="/solution">

      <assert test="(2*x*x*x - 5*y*y)*(z*z + 1) + 20 = 0">
        The solution does not satisfy the Diophantine equation.
      </assert>

    </rule>

  </pattern>

  <pattern name="check-syntax">

    <rule context="/">

      <assert test="solution">
        The document element must be the solution element.
      </assert>

    </rule>

    <rule context="/solution">

      <assert test="count(x) = 1">
        The solution element must contain one x subelement.
      </assert>

      <assert test="count(y) = 1">
        The solution element must contain one y subelement.
      </assert>

      <assert test="count(z) = 1">
        The solution element must contain one z subelement.
      </assert>

      <assert test="string(number(x)) != 'NaN' and floor(x) = x">
        The value of the x element must be an integer.
      </assert>

      <assert test="string(number(y)) != 'NaN' and floor(y) = y">
        The value of the y element must be an integer.
      </assert>

      <assert test="string(number(z)) != 'NaN' and floor(z) = z">
        The value of the z element must be an integer.
      </assert>

    </rule>

  </pattern>

</schema>
```

This Schematron schema is satisfiable if and only if the Diophantine equation has a solution among the integers. The following XML document conforms to this Schematron schema and specifies a solution among the integers to our Diophantine equation:

```
<?xml version="1.0"?>
<solution>
  <x>2</x>
  <y>2</y>
  <z>2</z>
</solution>
```

The following XML document does not conform to the Schematron schema and does not specify a solution among the integers to the Diophantine equation:

```
<?xml version="1.0"?>
<solution>
```

```
  <x>1</x>
  <y>1</y>
  <z>1</z>
</solution>
```

It should be noted that Guido Moerkotte proved the undecidability of determining whether or not an XPath expression is satisfiable. [Moerkotte] Specifically, he proved the following theorem:

> Given an XPath expression ***P***, it is undecidable whether there exist a document ***D*** and a node ***v*** in ***D*** such that evaluating ***P*** with current node ***v*** results in the empty set.

Guido Moerkotte provides several proofs for this theorem. One of them is based on the undecidability of Hilbert's Tenth problem. Another is based on the undecidability of the Post Correspondence Problem. Of course, his theorem implies that the Schematron conformance problem is undecidable.[6]

§ W3C XML Schema conformance problem

Recall that the DTD conformance problem is NP-HARD[3] because an instance of the 3SAT problem can be encoded as a DTD, such that the 3SAT instance has a solution if and only if the DTD is satisfiable. `ID` and `IDREF` attributes were required to encode a 3SAT instance as a DTD. Similarly, we can prove that the W3C XML Schema conformance problem is NP-HARD[3] using the 3SAT problem, since W3C XML Schema supports `ID` and `IDREF` attributes.

W3C XML Schema [W3C01], [Cover03], [xFront] also includes key and key reference features that are more powerful and flexible than `ID/IDREF` attributes. W3C XML Schema supports multiple keys, globally unique keys, locally unique keys, composite keys, keys on elements and/or attributes, etc. Marcelo Arenas, Wenfei Fan and Leonid Libkin have shown that the key and key reference features of W3C XML Schema make the W3C XML Schema conformance problem undecidable[6]; they use Hilbert's Tenth Problem to prove the undecidability of the W3C XML Schema conformance problem. [Arenas et al.]

§ Namespace Routing Language conformance problem

The Namespace Routing Language [Clark] is a new schema language that was developed by James Clark. An NRL [Namespace Routing Language] schema maps namespaces and/or elements to subschemas. Each subschema, which may be a RELAX NG schema, a W3C XML Schema, etc., is used to validate the corresponding namespace and/or element. NRL supports concurrent validation, where a namespace and/or element must conform to multiple subschemas.

It turns out that the NRL conformance problem is NP-HARD[3] even when the subschemas are restricted to RELAX NG schemas that use only the built-in datatypes. The following is a brief sketch of the proof:

1. An instance of the 3SAT problem[4] can be encoded as an NRL schema, such that the schema is satisfiable if and only if the 3SAT instance has a solution. The encoding of the 3SAT instance can be done in polynomial time.[1] Each clause in the 3SAT instance can be easily encoded as a RELAX NG subschema. These RELAX NG subschemas do not need the W3C XML Schema Datatypes.
2. If we had an algorithm that solved the NRL conformance problem, where the subschemas are limited to RELAX NG schemas, then we could use this algorithm (and the algorithm that encodes instances of the 3SAT problem into RELAX NG schemas) to solve the 3SAT problem, which is NP-COMPLETE.[5]
3. Therefore, the NRL conformance problem is NP-HARD[3] even when the subschemas are restricted to RELAX NG schemas.

Let's look at an NRL schema that encodes the following instance of the 3SAT problem:

```
( x1 OR x2 OR x3 ) AND
( x2 OR NOT(x3) OR NOT(x4) ) AND
( x3 OR x4 OR NOT(x1) )
```

The following NRL schema encodes this 3SAT instance:

```
<?xml version="1.0" encoding="UTF-8"?>
<rules xmlns="http://www.thaiopensource.com/validate/nrl">
  <namespace ns="">
    <validate schema="clause_1.rnc" schemaType="application/x-rnc"/>
    <validate schema="clause_2.rnc" schemaType="application/x-rnc"/>
    <validate schema="clause_3.rnc" schemaType="application/x-rnc"/>
  </namespace>
</rules>
```

This NRL schema is satisfiable if and only if the 3SAT instance has a solution. The NRL schema specifies that elements in the empty namespace must conform concurrently to three RELAX NG subschemas, which are written in the RELAX NG compact syntax. [OASIS02] Each subschema corresponds to one of the clauses of the 3SAT instance. The three subschemas include some common patterns that are defined in common.rnc.

The following is the clause_1.rnc subschema, which encodes the first clause of the 3SAT instance:

```
# Clause 1: ( x1 OR x2 OR x3 )
start = element solution {
    ( x1isTrue,        x2isTrueOrFalse, x3isTrueOrFalse, x4isTrueOrFalse ) |
    ( x1isTrueOrFalse, x2isTrue,        x3isTrueOrFalse, x4isTrueOrFalse ) |
    ( x1isTrueOrFalse, x2isTrueOrFalse, x3isTrue,        x4isTrueOrFalse )
}
include "common.rnc"
```

The following is the clause_2.rnc subschema, which encodes the second clause of the 3SAT instance:

```
# Clause 2: ( x2 OR NOT(x3) OR NOT(x4) )
start = element solution {
    ( x1isTrueOrFalse, x2isTrue,        x3isTrueOrFalse, x4isTrueOrFalse ) |
    ( x1isTrueOrFalse, x2isTrueOrFalse, x3isFalse,       x4isTrueOrFalse ) |
    ( x1isTrueOrFalse, x2isTrueOrFalse, x3isTrueOrFalse, x4isFalse )
}
include "common.rnc"
```

The following is the clause_3.rnc subschema, which encodes the third clause of the 3SAT instance:

```
# Clause 3: ( x3 OR x4 OR NOT(x1) )
start = element solution {
    ( x1isTrueOrFalse, x2isTrueOrFalse, x3isTrue,        x4isTrueOrFalse ) |
    ( x1isTrueOrFalse, x2isTrueOrFalse, x3isTrueOrFalse, x4isTrue        ) |
    ( x1isFalse,       x2isTrueOrFalse, x3isTrueOrFalse, x4isTrueOrFalse )
}
include "common.rnc"
```

The following are the patterns that are defined in common.rnc:

```
# Common patterns used by all the subschemas.
x1isTrue  = element x1isTrue  { empty }
x1isFalse = element x1isFalse { empty }
x2isTrue  = element x2isTrue  { empty }
x2isFalse = element x2isFalse { empty }
x3isTrue  = element x3isTrue  { empty }
x3isFalse = element x3isFalse { empty }
x4isTrue  = element x4isTrue  { empty }
x4isFalse = element x4isFalse { empty }
x1isTrueOrFalse = ( x1isTrue | x1isFalse )
x2isTrueOrFalse = ( x2isTrue | x2isFalse )
x3isTrueOrFalse = ( x3isTrue | x3isFalse )
x4isTrueOrFalse = ( x4isTrue | x4isFalse )
```

The following XML document conforms to the NRL schema and solves the instance of the 3SAT problem:

```
<?xml version="1.0"?>
<solution>
   <x1isTrue/>
   <x2isFalse/>
   <x3isFalse/>
   <x4isTrue/>
</solution>
```

The following XML document does not conform to the NRL schema and does not solve the instance of the 3SAT problem:

```
<?xml version="1.0"?>
<solution>
```

```
   <x1isTrue/>
   <x2isTrue/>
   <x3isFalse/>
   <x4isFalse/>
</solution>
```

In summary, the NRL conformance problem is NP-HARD[3] even when the subschemas are restricted to RELAX NG schemas.

§ Summary and conclusions

Table 1 summarizes the difficulty of the schema conformance problems that we discussed in this paper.

The schema conformance problem can probably be solved in linear time[2] for any XML schema language that is purely grammar-based.

Thus, an XML schema editor could probably do the following for any given schema:

- Tell you whether or not the set of grammar rules in the schema is satisfiable.
- Generate an XML document that conforms to the grammar rules in the schema.

The schema conformance problem is likely to be NP-HARD[3] for any XML schema language that provides one of the following features:

- grammar-based validity constraints combined with a simple key mechanism, such as the `ID/IDREF` feature of DTDs
- a data type system that can require a data value to match multiple regular expressions
- concurrent validation against multiple grammars

The schema conformance problem is likely to be undecidable[6] (*i.e.*, unsolvable) for any XML schema language that provides either of the following two features:

- grammar-based validity constraints combined with a flexible system for keys and key references
- validity constraints that are specified as XPath 1.0 expressions

Table 1

Schema Language	Difficulty of the Conformance Problem
DTD (with fixed-value IDREF attributes)	NP-HARD
DTD without `ID`s & `IDREF`s	Linear time
RELAX NG with W3C XML Schema Datatypes	NP-HARD
RELAX NG (using only the built-in datatypes)	Linear time
Schematron	Undecidable
W3C XML Schema	Undecidable
NRL (even when restricted to RELAX NG subschemas)	NP-HARD

Notes

1. A problem is *intractable* [Garey/Johnson] if it can't be solved in polynomial time. A problem can be solved in polynomial time if the number of steps required to solve the problem is bounded by a polynomial function of the size of the problem. An example of such a polynomial function is $\mathbf{T} <= \mathbf{aN}^{\mathbf{k}}$ where **T** is the number of steps required to solve the problem, **N** is the size of the problem, and **a** and **k** are constants.
2. A problem can be solved in *linear time* if the number of steps required to solve the problem is bounded by a linear function of the size of the problem. An example of such a linear function is **T <= aN + b** where **T** is the number of steps required to solve the problem, **N** is the size of the problem, and **a** and **b** are constants.
3. Problems that are NP-HARD [Non-deterministic Polynomial hard] [Wikipedia03], [Garey/Johnson] are at least as hard as NP-COMPLETE[5] problems. Problems that are NP-COMPLETE

[Wikipedia02] are decidable, but the amount of time required to solve an NP-COMPLETE problem is thought to grow exponentially as the size of the problem increases. Note that NP-HARD problems also include undecidable[6] problems.

4. The 3SAT [3-Satisfiability] problem (*a.k.a.*, the 3-CNF [3-Conjunctive Normal Form] problem) [Wikipedioa04], [Garey/Johnson] is a boolean logic problem that is NP-COMPLETE.[5]

   An instance of the 3SAT problem is a boolean expression that includes of one or more boolean variables. (The value of a boolean variable may be TRUE or FALSE.) Also, the boolean expression in a 3SAT instance is structured as one or more clauses that are ANDed together. Each clause consists of three literals that are ORed together; each literal is either a boolean variable (*e.g.*, x4) or the negation of a boolean variable (*e.g.*, NOT(x2)). Each boolean variable may appear multiple times in the boolean expression.

   For example, the following instance of the 3SAT problem includes four boolean variables (*i.e.*, x1, x2, x3, and x4) and 3 clauses:

```
( x1 OR x2 OR x3 ) AND
         ( x2 OR NOT(x3) OR NOT(x4) ) AND
         ( x3 OR x4 OR NOT(x1) )
```

   A 3SAT instance can be solved if there is an assignment of values to the boolean variables, such that the boolean expression evaluates to TRUE.

5. Problems that are NP-COMPLETE [Non-deterministic Polynomial complete] [Wikipedia02], [Garey/Johnson] are decidable (*i.e.*, solvable), but the amount of time required to solve an NP-COMPLETE problem is thought to grow exponentially as the size of the problem increases. For example, the amount of time required to solve the 3SAT[4] problem increases exponentially as the number of boolean variables increases. No one has found a polynomial time algorithm that solves an NP-COMPLETE problem, and it's very unlikely that anyone ever will.

6. A problem is *undecidable* [Wikipedia05], [Garey/Johnson] if it is a decision problem that cannot be solved. A decision problem is a problem that is posed as a "yes or no" question (*e.g.*, for any given number, is it a prime number?). A decision problem is undecidable if it is impossible to devise an algorithm that solves the problem. It's widely believed that if a problem is undecidable, then it can't be solved using software.

   Perhaps the most famous undecidable problem is Alan Turing's Halting Problem. [Wikipedia01] The halting problem asks: for any given algorithm and any given input to the algorithm, will the algorithm eventually halt (*i.e.*, stop) on that input?

7. *Hilbert's Tenth Problem* is to find a general algorithm that determines whether or not any given Diophantine equation [Wikipedia07] has a solution among the integers. [Wikipedia06] The problem is undecidable.[6]

   A Diophantine equation is a polynomial equation with integer coefficients. The following are examples of Diophantine equations:

   - $x + y = 0$
   - $x^3 + y^3 - z^3 = 0$
   - $(2x^3 - 5y^2)(z^2 + 1) + 20 = 0$

Acknowledgements

Thanks to the peer review panel of Extreme Markup Languages 2003 for their helpful and thoughtful comments. Thanks also to John Cowan for his comments on DTD conformance.

Bibliography

[Arenas et al.] Arenas, M., W. Fan, and L. Libkin. What's Hard about XML Schema Constraints? **http://www.cs.toronto.edu/~marenas/publications/xsc_dexa02.pdf**.

[Clark] Clark, J. Namespace Routing Language (NRL), June, 2003. **http://www.thaiopensource.com/relaxng/nrl.html**.

[Cover01] Cover Pages: Schematron: XML Structure Validation Language Using Patterns in Trees. **http://xml.coverpages.org/schematron.html**.

[Cover02] Cover Pages: RELAX NG. **http://xml.coverpages.org/relax-ng.html**.

[Cover03] Cover Pages: XML Schema. **http://xml.coverpages.org/schemas.html**.

[Fan/Libkin] Fan, W., and L. Libkin. On XML integrity constraints in the presence of DTDs. In PODS'01, pages 114-125.

[Garey/Johnson] Garey, M., and D. Johnson. *Computers and Intractability; A Guide to the Theory of NP-COMPLETEness*, 1979.

[Lyons] Lyons, R. The Undecidability of the Schematron Conformance Problem, Oct. 2003. **http://www.unidex.com/scp/**.

[Moerkotte] Moerkotte, G. Incorporating XSL Processing Into Database Engines. Technical Report 7, University of Mannheim, 2002.

[OASIS01] OASIS Technical Committee: RELAX NG. **http://www.oasis-open.org/committees/relax-ng/**.

[OASIS02] OASIS Technical Committee: RELAX NG Compact Syntax. **http://relaxng.org/compact.html**.

[PCP] PCP: A Nice Problem. **http://www.cs.ualberta.ca/~zhao/PCP/intro.htm**.

[Post01] Post's Correspondence Problem. **http://www.cis.ohio-state.edu/~gurari/theory-bk/theory-bk-fourse7.html**.

[Post02] The Undecidability of Predicate Logic and the Post Correspondence Problem. **http://www.cis.udel.edu/~chester/courses/604hw/undecidable**.

[Post03] Post's Correspondence Problem. **http://www.dcs.napier.ac.uk/~andrew/pcp.html**.

[Schematron] Academia Sinica Computing Centre's Schematron Home Page. **http://www.ascc.net/xml/resource/schematron/schematron.html**.

[SourceForge] Schematron Project on SourceForge.net. **http://sourceforge.net/projects/schematron**.

[ThaiOpen] Thai Open Source: RELAX NG. **http://www.thaiopensource.com/relaxng/**.

[Thompson] Thompson, H. ID/IDREF makes XML generation NP-hard. **http://aspn.activestate.com/ASPN/Mail/Message/xml-dev/1584981**.

[W3C01] W3C. XML Schema. W3C Recommendation, May 2001. **http://www.w3.org/XML/Schema**.

[W3C02] W3C. Extensible Markup Language (XML) 1.0, Second Edition. W3C Recommendation, Oct. 2000. **http://www.w3.org/TR/2000/REC-xml-20001006**.

[W3C03] W3C. XML Path Language (XPath). W3C Recommendation, Nov. 1999. **http://www.w3.org/TR/1999/REC-xpath-19991116**.

[Wikipedia01] Wikipedia. Halting problem. **http://www.wikipedia.org/wiki/Halting_Problem**.

[Wikipedia02] Wikipedia. NP-complete. **http://www.wikipedia.org/wiki/NP-Complete**.

[Wikipedia03] Wikipedia. NP-hard. **http://www.wikipedia.org/wiki/NP-hard**.

[Wikipedia05] Wikipedia. Decision problem. **http://www.wikipedia.org/wiki/Decision_problem**.

[Wikipedia06] Wikipedia. Hilbert's 10th Problem. **http://www.wikipedia.org/wiki/Hilbert's_tenth_problem**.

[Wikipedia07] Wikipedia. Diophantine equation. **http://www.wikipedia.org/wiki/Diophantine_equation**.

[Wikipedioa04] Wikipedia. Boolean satisfiability problem. **http://www.wikipedia.org/wiki/Boolean_satisfiability_problem**.

[xFront] xFront.com: XML Schema Tutorial. **http://www.xfront.com/**.

[ZVON01] ZVON.org: Schematron Tutorial. **http://www.zvon.org/xxl/SchematronTutorial/General/contents.html**.

[ZVON02] ZVON.org: DTD Tutorial. **http://www.zvon.org/xxl/DTDTutorial/General/book.html**.

The Author

Robert C. Lyons
Unidex, Inc.
8 Stoecker Road
Holmdel
New Jersey
US
07733
tel: +1-732-975-9877
fax: +1-732-975-9866
boblyons@unidex.com
http://www.unidex.com/

Bob Lyons is an XML and Java consultant with Unidex, Inc. Bob designs and develops e-business applications using XML, XSLT, Java, etc. Bob developed XML Convert, which is a Java application that converts flat files to XML and vice versa. He invented the Turing Machine Markup Language and developed a Universal Turing Machine in XSLT. Bob has given numerous presentations on XML, B2B integration, and EDI at XTech '99, XML '99, and other conferences.

Extreme Markup Languages 2003
Montréal, Québec, August 4-8, 2003
This paper was formatted from XML source via XSL
by Mulberry Technologies, Inc.