ISO/IEC CD 19757-3	
Date: 2010-04-14	Reference number: ISO/JTC 1/SC 34 **N 1419**
Supersedes document	

THIS DOCUMENT IS STILL UNDER STUDY AND SUBJECT TO CHANGE. IT SHOULD NOT BE USED FOR REFERENCE PURPOSES.

ISO/IEC JTC 1/SC 34 Document Description and Processing Languages Secretariat: Japan (JISC)	Circulated to P- and O-members, and to technical committees and organizations in liaison for: - discussion at x comment by x voting by (P-members only) **2010-07-14** Please vote and comment via the CIB on the ISO/TC server by the due date indicated.

ISO/IEC CD 19757-3

Title: Information Technology — Document Schema Definition Languages (DSDL) — Part 3: Rule-based validation — Schematron

Project: 1.34.19757.03.00.00.02

Introductory note:

In accordance with Resolution 6 of the SC 34 plenary meeting held in Seattle, WA, USA, 2009-09-17, SC 34 approved initiation of this subproject. The project subdivision proposal submitted to JTC 1 for endorsement is contained in JTC 1 N 9821. In accordance with Resolution 6 of the SC 34 Seattle plenary meeting, this document is circulated to the SC 34 national bodies for a three-month CD ballot. Please vote and comment via the E-balloting system.

No. of pages: 47

Secretariat, ISO/IEC JTC 1/SC 34
IPSJ/ITSCJ (Information Processing Society of Japan/Information Technology Standards Commission of Japan)*
Room 308-3, Kikai-Shinko-Kaikan Bldg., 3-5-8, Shiba-Koen, Minato-ku, Tokyo 105-0011 JAPAN
Tel: +81 3 3431 2808; Fax: +81 3 3431 6493; E-mail: kimura@itscj.ipsj.or.jp
*A Standard Organization accredited by JISC

Contents

Page

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 3.

ISO/IEC 19757-3 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information Technology*, Subcommittee SC 34, Document Description and Processing Languages.

ISO/IEC 19757 consists of the following parts, under the general title *Document Schema Definition Languages (DSDL)*:

— *Part 1: Overview*

— *Part 2: Regular-grammar-based Validation — RELAX NG*

— *Part 3: Rule-based Validation — Schematron*

— *Part 4: Namespace-based Validation Dispatching Language — NVDL*

— *Part 5: Extensible Datatypes*

— *Part 7: Character Repertoire Description Language — CREPDL*

— *Part 8: Document Schema Renaming Language — DSRL*

— *Part 9: Datatype- and Namespace-aware DTDs*

— *Part 11: Schema Association*

Introduction

ISO/IEC 19757 defines a set of Document Schema Definition Languages (DSDL) that can be used to specify one or more validation processes performed against Extensible Stylesheet Language (XML) or Standard Generalized Markup Language (SGML) documents. (XML is an application profile SGML ISO 8879:1986.)

A document model is an expression of the constraints to be placed on the structure and content of documents to be validated with the model. A number of technologies have been developed through various formal and informal consortia since the development of Document Type Definitions (DTDs) as part of ISO 8879, notably by the World Wide Web Consortium (W3C) and the Organization for the Advancement of Structured Information Standards (OASIS). A number of validation technologies are standardized in DSDL to complement those already available as standards or from industry.

To validate that a structured document conforms to specified constraints in structure and content relieves the potentially many applications acting on the document from having to duplicate the task of confirming that such requirements have been met. Historically, such tasks and expressions have been developed and utilized in isolation, without consideration of how the features and functionality available in other technologies might enhance validation objectives.

The main objective of ISO/IEC 19757 is to bring together different validation-related tasks and expressions to form a single extensible framework that allows technologies to work in series or in parallel to produce a single or a set of validation results. The extensibility of DSDL accommodates validation technologies not yet designed or specified.

In the past, different design and use criteria have led users to choose different validation technologies for different portions of their information. Bringing together information within a single XML document sometimes prevents existing document models from being used to validate sections of data. By providing an integrated suite of constraint description languages that can be applied to different subsets of a single XML document, ISO/IEC 19757 allows different validation technologies to be integrated under a well-defined validation policy.

The structure of this part of ISO/IEC 19757 is as follows. Clause 5 describes the syntax of an ISO Schematron schema. Clause 6 describes the semantics of a correct ISO Schematron schema; the semantics specify when a document is valid with respect to an ISO Schematron schema. Clause 7 describes conformance requirements for implementations of ISO Schematron validators. Annex A is a normative annex providing the ISO/IEC 19757-2 (RELAX NG) schema for ISO Schematron. Annex B is a normative annex providing the ISO Schematron schema for constraints in ISO Schematron that cannot be expressed by the schema of Annex A. Annex C is a normative annex providing the default query language binding to XSLT1. Annex D is a non-normative annex providing a ISO/IEC 19757-2 (RELAX NG compact syntax) schema and corresponding ISO Schematron schema for a simple XML language Schematron Validation Report Language.Annex E is a non-normative annex providing motivating design requirements for ISO Schematron. Annex F is a normative annex allowing certain Schematron elements to be used in external vocabularies. Annex G is a non-normative annex with a simple example of a multi-lingual schema.

This edition is backwards compatible with IS19757-3:2006, supercedes it, and augments it with the following capabilities: patterns may validate different documents, the inclusion mechanism has been supplemented by an enhanced extension mechanism, assertions may have linked properties, and SVRL may take richer text. As well, this edition provides extra query language bindings, in particular for XSLT2.

Considered as a document type, a Schematron schema contains natural-language assertions concerning a set of documents, marked up with various elements and attributes for testing these natural-language assertions, and for simplifying and grouping assertions.

Considered theoretically, a Schematron schema reduces to a non-chaining rule system whose terms are Boolean functions invoking an external query language on the instance and other visible XML documents, with syntactic features to reduce specification size and to allow efficient implementation.

Considered analytically, Schematron has two characteristic high-level abstractions: the pattern and the phase. These allow the representation of non-regular, non-sequential constraints that Part 2 cannot specify,

and various dynamic or contingent constraints.

This part of ISO/IEC 19757 is based on the Schematron[2] assertion language. The `let` element is based on XCSL[4]. Other features arise from the half-dozen early Open Source implementations of Schematron in diverse programming languages and from discussions in electronic forums by Schematron users and implementers.

Document Schema Definition Languages (DSDL) — Part 3: Rule-based validation — Schematron

1 Scope

This part of ISO/IEC 19757 specifies Schematron, a schema language for XML. This part of ISO/IEC 19757 establishes requirements for Schematron schemas and specifies when an XML document matches the patterns specified by a Schematron schema.

2 Normative references

The following normative documents contain provisions which, through reference in this text, constitute provisions of this part of ISO/IEC 19757. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this part of ISO/IEC 19757 are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative document referred to applies. Members of ISO and IEC maintain registers of currently valid International Standards.

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

NOTE Each of the following documents has a unique identifier that is used to cite the document in the text. The unique identifier consists of the part of the reference up to the first comma.

IRI, *RFC 3987*, IETF Request for Comment, Current version, http://tools.ietf.org/html/rfc3987

W3C XML 1.0, *Extensible Markup Language (XML) 1.0 (Fourth Edition)*, W3C Recommendation, 14 June 2006, http://www.w3.org/TR/2006/PER-xml-20060614/

XPath, *XML Path Language (XPath) Version 1.0*, W3C Recommendation, 16 November 1999, http://www.w3.org/TR/xpath

XPath2, *XML Path Language (XPath) 2.0*, W3C Recommendation, 23 January 2007, http://www.w3.org/TR/xpath20/

XPath2 Functions, *XQuery 1.0 and XPath 2.0 Functions and Operators*, W3C Recommendation, 23 January 2007, http://www.w3.org/TR/xpath-functions/

XDM, *XQuery 1.0 and XPath 2.0 Data Model (XDM)*, W3C Recommendation, 23 January 2007, http://www.w3.org/TR/xpath-datamodel/

XSLT1, *XSL Transformations (XSLT) Version 1.0*, W3C Recommendation, 16 November 1999, http://www.w3.org/TR/xslt

XSLT2, *XSL Transformations (XSLT) Version 2.0*, W3C Recommendation, 23 January 2007, http://www.w3.org/TR/xslt20/

3 Terms and definitions

For the purposes of this document, the following terms and definitions apply:

3.1 abstract pattern

pattern in a rule that has been parameterized to enable reuse

3.2 abstract rule

collection of assertions which can be included in other rules but which does not fire itself

3.3 active pattern

pattern belonging to the active phase

3.4 active phase

one particular phase, whose patterns are used for validation

3.5 assertion

natural-language assertion with corresponding assertion test and ancillary attributes: assertions are marked up with `assert` and `report` elements

3.6 assertion test

assertion modelled or implemented by a Boolean query; an assertion test "succeeds" or "fails"

3.7 compound document

a notional instance document which has been divided into the original instance and a number of subordinate well-formed XML documents. These subordinate documents may also contain schema-like information.

3.8 correct schema

schema that satisfies all the requirements of this part of ISO/IEC 19757

3.9 diagnostic

named natural language statements providing information to end-users of validators concerning the expected and actual values together with repair hints

3.10 elaborated rule context expression

single rule context expression which explicitly disallows items selected by lexically previous rule contexts in the same pattern

3.11 good schema

correct schema with queries which terminate and do not add constraints to those of the natural-language assertions. Note: It may not be possible to compute that a schema is good.

3.12 implementation

implementation of a Schematron validator

3.13 name

token with no whitespace characters

3.14 natural-language assertion

natural-language statement expressing some part of a pattern; a natural-language assertion is "met" or "unmet"

3.15 pattern

named structure in instances specified in a schema by a lexically-ordered collection of rules

3.16 phase

named, unordered collection of patterns; patterns may belong to more than one phase; two names, `#ALL` and `#DEFAULT`, are reserved with particular meanings

3.17 progressive validation

the validation of constraints in stages determined or grouped to some extent by the schema author rather than, for example, entirely determined by document order

3.18 property

named data giving additional metadata on an assertion or report

3.19 query language binding

named set, specified in a document called a Query Language Binding, of the languages and conventions used for assertion tests, rule-context expressions and so on, by a particular Schematron implementation.

3.20 rule

unordered collection of assertions with a rule-context expression and ancillary attributes

3.21 rule context

element or other information item used for assertion tests; a rule is said to fire when an information item matches the rule context

3.22 rule-context expression

a query to specify subjects; a rule-context is said to match an information item when that information item has not been matched by any lexically-previous rule context expressions in the same pattern and the information item is one of the information items that the query would specify

3.23 schema

specification of a set of XML documents

3.24 subject

particular information item which corresponds to the object of interest of the natural-language assertions and typically is matched by the context expression of a rule

3.25 valid with respect to a schema

member of the set of XML documents described by the schema: an instance document is valid if no assertion tests in fired rules of active patterns fail

3.26 variable

constant value, evaluated within the parent schema, phase, pattern or rule and scoped within the parent schema, phase, pattern or rule

4 Notation

4.1 XPath

This part of ISO/IEC 19757 uses XPath to identify information items in Schematron schemas.

4.2 Predicate Logic

This part of ISO/IEC 19757 uses predicate logic to express the semantics of Schematron schema. The following symbols are defined for use in s6.3:

— () Grouping delimiters

— ∀ "for all". Prefix operator.

— ¬ "not". Prefix operator.

— ∈ "is member of", in set operative sense. Prefix operator.

— , "and" (sequence). Infix operator.

— : "where". Such that. Infix operator.

— ⇔ "if and only if". Infix operator.

5 Syntax

5.1 Well-formedness

A Schematron schema shall be a well-formed XML document, according to the version of XML used.

5.2 Namespace

All elements shown in the grammar for Schematron are qualified with the namespace URI [IRI]:

`http://purl.oclc.org/dsdl/schematron`

In subsequent clauses, the prefix `sch` is taken as bound to the Schematron namespace URI for exposition purposes. The prefix `sch` is not reserved or required by this part of ISO/IEC 19757.

Any element can also have foreign attributes in addition to the attributes shown in the grammar. A foreign attribute is an attribute with a name whose namespace URI is neither the empty string nor the Schematron namespace URI. Any non-empty element may have foreign child elements in addition to the child elements shown in the grammar. A foreign element is an element with a name whose namespace URI is not the Schematron namespace URI. There are no constraints on the relative position of foreign child elements with respect to other child elements.

5.3 Whitespace

Any element can also have as children strings that consist entirely of whitespace characters, where a whitespace character is one of U+0020, U+009, U+00D or U+00A. There are no constraints on the relative

position of whitespace string children with respect to child elements.

NOTE Leading and trailing whitespace should be stripped from attributes defined by this part. Whitespace should be collapsed in elements defined by this part that allow text. Whitespace may be stripped from elements defined by this part that do not allow text.

5.4 Core Elements

The grammar for Schematron elements is given in Annex A.

5.4.1 `active` element

The required `pattern` attribute is a reference to a pattern that is active in the current phase.

5.4.2 `assert` element

An assertion made about the context nodes. The data content is a natural-language assertion. The required `test` attribute is an assertion test evaluated in the current context. If the test evaluates positive, the report succeeds. The optional `diagnostics` attribute is a reference to further diagnostic information.

The natural-language assertion shall be a positive statement of a constraint.

NOTE The natural-language assertion may contain information about actual values in addition to expected values and may contain diagnostic information. Users should note, however, that the `diagnostic` element is provided for such information to encourage clear statement of the natural-language assertion.

The `icon`, `see` and `fpi` attributes allow rich interfaces and documentation. They are defined below.

The `flag` attribute allows more detailed outcomes. It is defined below.

The `role` and `subject` attributes allow explicit identification of some part of a pattern. They are defined below.

5.4.3 `extends` element

The extends element allows reference to the contents of other declarations. The extends element shall either have an href attribute or a rule attribute but not both.

Abstract rules are named lists of assertions without a context expression. An extends element with a rule attribute shall reference an abstract rule. The current rule uses all the assertions from the abstract rule it extends.

An extends attribute with an href attribute shall reference external declarations. The href attribute is an IRI reference to an external well-formed XML document or to an element in external well-formed XML document that is Schematron element of the same type as the parent element of the extends element. The contents of that referenced element shall be inserted in place of the extends element.

In such a case, the relative position of elements in the post-inclusion document may be to that extent invalid against the schema for Schematron Annex A, however other schema constraints such as containment shall still apply.

NOTE The capability to extend with multiple elements was not part of IS19757-3:2006.

5.4.4 `include` element

The required `href` attribute shall be an IRI reference to a well-formed XML document or to an element in a

well-formed XML document.

The referenced element shall be inserted in place of the include element. The referenced element shall be a type which is allowed by the grammar for Schematron at the location of the include element.

5.4.5 `let` element

A declaration of a named variable. If the `let` element is the child of a `rule` element, the variable is calculated and scoped to the current rule and context. Otherwise, the variable is calculated with the context of the instance document root.

The required `name` attribute is the name of the variable. The `value` attribute is an expression evaluated in the current context. If no `value` attribute is specified, the value of the attribute is the element content of the `let` element.

It is an error to reference a variable that has not been defined in the current schema, phase, pattern, or rule, if the query language binding allows this to be determined reliably. It is an error for a variable to be multiply defined in the current schema, phase, pattern and rule.

The variable is substituted into assertion tests and other expressions in the same rule before the test or expression is evaluated. The query language binding specifies which lexical conventions are used to detect references to variables.

An implementation may provide a facility to override the values of top-level variables specified by `let` elements under the `schema` element. For example, an implementation may allow top-level variables to be supplied on the command line. The values provided are strings or data objects, not expressions.

5.4.6 `name` element

Provides the names of nodes from the instance document to allow clearer assertions and diagnostics. The optional `path` attribute is an expression evaluated in the current context that returns a string that is the name of a node. In the latter case, the name of the node is used.

An implementation which does not report natural-language assertions is not required to make use of this element.

5.4.7 `ns` element

Specification of a namespace prefix and URI. The required `prefix` attribute is an XML name with no colon character. The required `uri` attribute is a namespace URI [IRI].

NOTE Because the characters allowed as names may change in versions of XML subsequent to W3C XML 1.0, the ISO/IEC 19757-2 (RELAX NG Compact Syntax) schema for Schematron does not constrain the prefix to particular characters.

In an ISO Schematron schema, namespace prefixes in context expressions, assertion tests and other query expressions should use the namespace bindings provided by this element. Namespace prefixes should not use the namespace bindings in scope for element and attribute names.

5.4.8 `param` element

A name-value pair providing parameters for an abstract pattern. The required `name` attribute is an XML name with no colon. The required `value` attribute is a fragment of a query.

5.4.9 `pattern` element

A structure, simple or complex. A set of rules giving constraints that are in some way related. The `id` attribute provides a unique name for the pattern and is required for abstract patterns.

The optional documents attribute provides IRIs of subordinate document the rule contexts are relative to. If the expression evaluates to more than one IRI, then the pattern is sought in each of the documents. The documents attribute is evaluated in the context of the original instance document root.

NOTE The documents attribute was not part of IS19757-3:2006.

The `title` and `p` elements allow rich documentation.

The `icon`, `see` and `fpi` attributes allow rich interfaces and documentation.

When a `pattern` element has the attribute `abstract` with a value `true`, then the pattern defines an abstract pattern. An abstract pattern shall not have a `is-a` attribute and shall have an `id` attribute.

Abstract patterns allow a common definition mechanism for structures which use different names and paths, but which are at heart the same. For example, there are different table markup languages, but they all can be in large part represented as an abstract pattern where a table contains rows and rows contain entries, as defined in the following example using the default query language binding:

```
<sch:pattern abstract="true" id="table">
    <sch:rule context="$table">
        <sch:assert test="$row">
        The element <name/> is a table. Tables contain rows.
        </sch:assert>
    </sch:rule>

    <sch:rule context="$row">
        <sch:assert test="$entry">
        The element <name/> is a table row. Rows contain entries.
        </sch:assert>
    </sch:rule>

</sch:pattern>
```

When a `pattern` element has the attribute `is-a` with a value specifying the name of an abstract pattern, then the pattern is an instance of an abstract pattern. Such a pattern shall not contain any `rule` elements, but shall have `param` elements for all parameters used in the abstract pattern.

The following example uses the abstract pattern for tables given above to create three patterns for tables with different names or structures.

```
<sch:pattern is-a="table" id="HTML_Table">
    <sch:param name="table" value="table"/>
    <sch:param name="row"   value="tr"/>
    <sch:param name="entry" value="td|th"/>
</sch:pattern>

<sch:pattern is-a="table" id="CALS_Table">
    <sch:param name="table" value="table"/>
    <sch:param name="row"   value=".//row"/>
    <sch:param name="entry" value="cell"/>
</sch:pattern>

<sch:pattern is-a="table" id="calendar">
    <sch:param name="table" value="calendar/year"/>
    <sch:param name="row"   value="week"/>
    <sch:param name="entry" value="day"/>
</sch:pattern>
```

When creating an instance of an abstract pattern, the parameter values supplied by the `param` element replace the parameter references used in the abstract patterns. The examples above use the default query language binding in which the character is used as the delimiter for parameter references.

Thus, given the abstract patterns defined earlier in this clause, the patterns defined above are equivalent to the following, with the `id` elements shown expanded:

```
<sch:pattern id="HTML_table">
    <sch:rule context="table">
        <sch:assert test="tr">
        The element table is a table. Tables containing rows.
        </sch:assert>
    </sch:rule>

    <sch:rule context="tr">
        <sch:assert test="td|th">
        The element tr is a table row. Rows contain entries.
        </sch:assert>
    </sch:rule>

</sch:pattern>

<sch:pattern id="CALS_table">
    <sch:rule context="table">
        <sch:assert test=".//row">
        The element table is a table. Tables containing rows.
        </sch:assert>
    </sch:rule>

    <sch:rule context=".//row">
        <sch:assert test="cell">
        The element row is a table row. Rows contain entries.
        </sch:assert>
    </sch:rule>

</sch:pattern>

<sch:pattern id="calendar">
    <sch:rule context="calendar/year">
        <sch:assert test="week">
        The element year is a table. Tables containing rows.
        </sch:assert>
    </sch:rule>

    <sch:rule context="week">
        <sch:assert test="day">
        The element week is a table row. Rows contain entries.
        </sch:assert>
    </sch:rule>

</sch:pattern>
```

5.4.10 `phase` element

A grouping of patterns, to name and declare variations in schemas, for example, to support progressive validation. The required `id` attribute is the name of the phase. The implementation determines which phase to use for validating documents, for example by user command.

Two names, `#ALL` and `#DEFAULT`, have special meanings. The name `#ALL` is reserved and available for use by implementations to denote that all patterns are active. The name `#DEFAULT` is reserved and available for use by implementations to denote that the name given in the `defaultPhase` attribute on the `schema` element should be used. If no `defaultPhase` is specified, then all patterns are active.

NOTE The names `#ALL` and `#DEFAULT` shall not be used in a Schematron schema. They are for use when invoking or configuring schema validation, for example as a command-line parameter.

The `icon`, `see` and `fpi` attributes allow rich interfaces and documentation.

5.4.11 `report` element

An assertion made about the context nodes. The data content is a natural-language assertion. The required `test` attribute is an assertion test evaluated in the current context. If the test evaluates positive, the report succeeds. The optional `diagnostics` attribute is a reference to further diagnostic information.

The natural-language assertion shall be a positive statement of a found pattern or a negative statement of a constraint.

NOTE The natural-language assertion may contain information about actual values in addition to expected values and may contain diagnostic information. Users should note, however, that the `diagnostic` element is provided for such information to encourage clear statement of the natural-language assertion.

The `icon`, `see` and `fpi` attributes allow rich interfaces and documentation. They are defined below.

The `flag` attribute allows more detailed outcomes. It is defined below.

The `role` and `subject` attributes allow explicit identification of some part of a pattern. They are defined below.

5.4.12 `rule` element

A list of assertions tested within the context specified by the required `context` attribute. The `context` attribute specifies the rule context expression.

NOTE It is not an error if a rule never fires in a document. In order to test that a document always has some context, a new pattern should be created from the context of the document, with an assertion requiring the element or attribute.

The `icon`, `see` and `fpi` attributes allow rich interfaces and documentation.

The `flag` attribute allows more detailed outcomes. It is defined below.

The `role` and `subject` attributes allow explicit identification of some part of a pattern as part of the validation outcome. They are defined below.

When the `rule` element has the attribute `abstract` with a value `true`, then the rule is an abstract rule. An abstract rule shall not have a `context` attribute. An abstract rule is a list of assertions that will be invoked by other rules belonging to the same pattern using the `extends` element. Abstract rules provide a mechanism for reducing schema size.

5.4.13 `schema` element

The top-level element of a Schematron schema.

The optional `schemaVersion` attribute gives the version of the schema. Its allowed values are not defined by this part of ISO/IEC 19757 and its use is implementation-dependent.

The optional `queryBinding` attribute provides the short name of the query language binding in use. If this attribute is specified, it is an error if it has a value that the current implementation does not support.

The `defaultPhase` attribute may be used to indicate the phase to use in the absence of explicit user-supplied information.

The `title` and `p` elements allow rich documentation.

The `icon`, `see` and `fpi` attributes allow rich interfaces and documentation.

5.4.14 `value-of` element

Finds or calculates values from the instance document to allow clearer assertions and diagnostics. The required `select` attribute is an expression evaluated in the current context that returns a string.

Variable references in the `select` attribute are resolved in the scope of the current schema, phase, pattern and rule.

An implementation which does not report natural-language assertions is not required to make use of this element.

5.5 Ancillary Elements and Attributes

5.5.1 `diagnostic` element

A natural-language message giving more specific details concerning a failed assertion, such as found *versus* expected values and repair hints.

NOTE Diagnostics in multiple languages may be supported by using a different `diagnostic` element for each language, with the appropriate `xml:lang` language attribute, and referencing all the unique identifiers of the `diagnostic` elements in the `diagnostics` attribute of the assertion. Annex G gives a simple example of a multi-lingual schema.

An implementation is not required to make use of this element.

NOTE Typical values for the role attribute on a diagnostic element might be `warning`, `caution` or `note`.

5.5.2 `diagnostics` element

A section containing individual diagnostic elements.

An implementation is not required to make use of this element.

5.5.3 `dir` element

A section of natural-language text with a direction specified by the `value` attribute. The value `ltr` indicates left-to-right text; the value `rtl` indicates right-to-left text.

An implementation is not required to make use of this element.

5.5.4 `emph` element

A portion of text that should be rendered with some emphasis.

An implementation is not required to make use of this element.

5.5.5 `flag` attribute

A Boolean variable with initial value false. A flag is implicitly declared by an assertion or rule having a `flag` attribute with that name. The value of a flag becomes true when an assertion with that flag fails or a rule with that flag fires.

The purpose of flags is to convey state or severity information to a subsequent process.

An implementation is not required to make use of this attribute.

5.5.6 `fpi` attribute

A formal public identifier for the schema, phase or other element.

An implementation is not required to make use of this attribute.

5.5.7 `icon` attribute

The location of a graphics file containing some visible representation of the severity, significance or other grouping of the associated element.

An implementation is not required to make use of this attribute.

5.5.8 `p` element

A paragraph of natural language text containing maintainer and user information about the parent element. The schema can nominate paragraphs that should be rendered in a distinct way, keyed with the `class` attribute.

An implementation is not required to make use of this element.

5.5.9 `properties` element

A section containing individual property elements. An implementation is not required to make use of this element

NOTE The properties element was not part of IS19757-3:2006.

5.5.10 `property` element

An element to declare additional arbitrary properties for the subjects of failed assertions and successful reports.

NOTE The property element is suitable for linking assertions or reports to actions, to additional metadata, to datatyping, and to dynamically extracted text related to the subject.

The optional scheme element should be an IRI or other public identifier which specifies the notation used for the metadata value.

NOTE Where the property value contains elements in a well-known namespace or where the scheme used is otherwise obvious or unnecessary, the scheme element may be omitted. For example, if the property element contains an IS19757-7 Character Repertoire Description Language schema, no scheme attribute is appropriate.

NOTE Properties are defined as assertions in order to associate assertion text with the property. A property of an assert element typically should be information for validation. A property of a report element typically should be information for document augmentation (post schema validation information set.)

An implementation is not required to make use of this element.

NOTE The property element was not part of IS19757-3:2006.

5.5.11 `role` attribute

A name describing the function of the assertion or context node in the pattern. If the assertion has a `subject` attribute, then the role labels the arc between the context node and any nodes which match the path expression given by the `subject` attribute.

An implementation is not required to make use of this attribute.

5.5.12 `see` attribute

The URI [IRI] of external information of interest to maintainers and users of the schema.

An implementation is not required to make use of this attribute.

5.5.13 `span` element

A portion of some paragraph that should be rendered in a distinct way, keyed with the `class` attribute.

An implementation is not required to make use of this element.

5.5.14 `subject` attribute

A path allowing more precise specification of nodes. The path expression is evaluated in the context of the context node of the current rule. If no subject attribute is specified, the current subject node may be used.

NOTE The `subject` attribute is required because the rule context may have been selected for reasons of convenience or performance, in association with the particular assertion tests. In such cases, the rule context may not be useful to identify users, and the nodes located by the `subject` attribute may be more useful. Similarly, it may not be possible to determine from an assertion test which nodes the assertion test has tested. In such a case, the nodes located by the `subject` attribute may be more useful.

An implementation is not required to make use of this element.

5.5.15 `title` element

A summary of the purpose or role of the schema, pattern or rule for the purpose of documentation or a rich user interface.

An implementation is not required to make use of this element.

6 Semantics

6.1 Validation Function

A general Schematron validator is a function returning "valid", "invalid" or "error". The function notionally performs two steps: transforming the schema into a minimal syntax, then testing the instance against the minimal syntax.

NOTE This part of ISO/IEC 19757 does not constrain other information provided by an implementation nor other uses of Schematron schemas. However, it is the intent of this part of ISO/IEC 19757 to support implementations to provide rich, specific diagnostics customised with values that assist in detecting and rectifying problems.

A Schematron validator is a function over the following:

— a query language binding

— a schema document

— an instance to be validated

— external XML documents addressed using information in the instance or schema

— a phase name, or `#ALL` if all patterns shall be active patterns, or `#DEFAULT` if the `defaultPhase` attribute on the schema element shall be used

— a list of name-value pairs, if the schema uses external variables.

6.2 Minimal Syntax

To simplify the specification of semantics later, the following transformation steps are first applied to a schema, resulting in a schema in the minimal syntax:

— Resolve all inclusions by replacing the `include` element by the resource linked to.

— Resolve all abstract patterns by replacing parameter references with actual parameter values in all enclosed attributes that contain queries.

— Resolve all abstract rules in the schema by replacing the `extends` elements with the contents of the abstract rule identified.

— Negate all `report` elements into `assert` elements.

— Remove elements used for diagnostics and documentation.

The resulting minimal syntax is also a valid Schematron instance in the full syntax. The minimal syntax differs from the complex syntax by not containing the following XPaths:

— `//sch:include`

— `//sch:pattern[@abstract="true"]`

— `//sch:pattern[@is-a]`

— `//sch:rule[@abstract="true"]`

— `//sch:extends`

— `//sch:report`

— `//sch:diagnostics`

— `//sch:p`

— `//sch:title`

6.3 Schema Semantics

This clause gives the semantics of a good schema that has been resolved into the minimal form.

This predicate treats an instance as a set of contexts, a schema as a set of phases, a phase as a set of patterns, a pattern as a set of rules, and a rule as a set of assertions.

A compound document is valid in respect to a schema in a phase when the following predicate is satisfied for each document:

```
∀ ( context, pattern, rule, assertion):
    ( context ∈ instance,
      active-phase ∈ schema,
```



```
      pattern ∈ active-phase,
      rule ∈ pattern,
      assertion ∈ rule :
        ( match ( rule, context, instance ),
          ∀ (previous-rule) :
            ( previous-rule ∈ pattern,
              position (previous-rule) < position( rule):
                  ¬ match(previous-rule, context, instance)
            )
        )
    )
    ⇔ assert ( assertion, context, instance) = true
```

where

— `position( rule )` is the position of the member in the set, a cardinal number,

— `match( rule, context, instance)` is a predicate supplied by the query language binding, and

— `assert( assertion, context, instance )` is a predicate supplied by the query language binding.

NOTE In natural language, a document is valid against a schema in a phase if: *There exists an instance, schema and active-phase combination where, for each context, pattern, rule and assertion (the context being a member of that instance, the active phase being a member of the schema, the pattern being a member of that active phase, the rule being a member of that pattern, the assertion being a member of that rule), the following is true: if the context of an instance matches the rule, and that context has not been matched by a previous rule in the same pattern, then the particular assertion evaluates to true when evaluated with the particular context and instance.*

6.4 Query Language Binding

A query language binding shall provide the following:

— The general query language used. A name token which identifies the query language. The data model.

— The rule context query language. The rule context scope.

— The assertion test, a function which returns a data value coercible into Boolean.

NOTE The following query language names are reserved without further definition. Implementations which use different query language bindings are encouraged to use one of these names if appropriate : `stx`, `xslt`, `exslt`, `xslt2`, `xpath`, `xpath2`, `xquery`.

A schema language binding may also provide the following:

— The data models of the various query languages, the conversion between data models, and the treatment of information items: which information items are stripped or ignored, which information items are errors, and which information items are used.

— The `name` query language, a function which returns a data value coercible into a string.

— The `value-of` query language, a function which returns a data value coercible into a string.

— The `let` value query language, a function which returns a data value.

— The `documents` selecting query language, a function which returns a data value interpretable as a list or sequence of IRIs.

— The variable delimiter convention, a lexical convention such as a delimiter by which the use of a variable in a query expression shall be recognized.

— The abstract pattern parameter convention, a lexical convention such as a delimiter by which the parameters of abstract patterns inside query expressions shall be recognised.

The query language binding may also specify the element types in other namespaces which provide query-language-specific ancillary information or pragmatic hints.

The query language binding shall specify any whitespace processing required on queries.

The query language binding shall specify any restrictions to the value of name tokens.

A Schematron implementation which does not support the query language binding, specified in a schema with the `queryBinding` attribute, shall fail with an error.

6.5 Order and side-effects

The order in which elements are validated is implementation-dependent, without altering the validity of the instance.

The order in which patterns are used is implementation-dependent, without altering the validity of the instance.

The order in which assertions are tested is implementation-dependent, without altering the validity of the instance.

The only elements for which order is significant are the `rule` and `let` elements.

A `rule` element acts as an if-then-else statement within each pattern. An implementation may make order non-significant by converting rules context expressions to elaborated rule context expressions.

NOTE The behaviour of the `rule` element allows constraints that would require a complex context expression to be factored into simpler expressions in different rules.

A `let` element may use lexically previous variables within the same rule or global variables.

NOTE A wide variety of implementation strategies are therefore possible.

All queries shall act as pure functions. Queries shall not alter the instance in any way visible to other queries. This part of ISO/IEC 19757 does not specify any outcome augmentation of the instance being validated.

7 Conformance

7.1 Simple Conformance

A simple-conformance implementation shall be able to report for any XML document that its structure does not conform to that of a valid Schematron schema.

— A valid schema conforms to the constraints of Annex A, the normative ISO/IEC 19757-2 (RELAX NG Compact Syntax) schema of this part of ISO/IEC 19757.

A simple-conformance implementation shall be able to determine for any XML document and for any good schema whether the document is valid with respect to the schema.

NOTE It is not a requirement of this part of ISO/IEC 19757 that a simple-conformance implementation shall be able to determine whether validation will terminate or whether the queries are feasible against some other schema for the instance. The ability to determine these depends on the query language used. Where the query language allows incorrectness to be established, implementations are encouraged to report this information as part of validation.

NOTE It is not a requirement of this part of ISO/IEC 19757 that a simple-conformance implementation shall be able to generate validation reports in the Schematron Validation Report Language, defined in Annex D.

7.2 Full Conformance

A full-conformance implementation shall be able to determine for any XML document whether it is a correct schema.

— A valid schema conforms to the constraints of Annex A, the normative ISO/IEC 19757-2 (RELAX NG Compact Syntax) schema of this part of ISO/IEC 19757.

— A valid schema conforms to the constraints of Annex B, the normative ISO Schematron schema of this part of ISO/IEC 19757.

— A correct schema's attributes conform to the grammars specified by the query language binding in use.

— A correct schema has one definition only in scope for any variable name in any context.

— The values of the attributes `flag`, `id`, `name` and `prefix` are well-formed names in the version of XML being used.

A full-conformance implementation shall be able to determine for any XML document and for any good schema whether the document is valid with respect to the schema.

NOTE It is not a requirement of this part of ISO/IEC 19757 that a full-conformance implementation shall be able to determine whether the validation will terminate or whether the queries are feasible against some other schema for the instance. The ability to determine these depends on the query language used. Where the query language allows incorrectness to be established, implementations are encouraged to report this information as part of validation.

NOTE It is not a requirement of this part of ISO/IEC 19757 that a full-conformance implementation shall be able to generate validation reports in the Schematron Validation Report Language, defined in Annex D.

Annex A
(normative)

RELAX NG schema for Schematron

A correct Schematron schema shall be valid with respect to the following ISO/IEC 19757-2 (RELAX NG Compact Syntax) schema.

```
#        © ISO/IEC 2010


# The following permission notice and disclaimer shall be included in all
# copies of this XML schema ("the Schema"), and derivations of the Schema:

# Permission is hereby granted, free of charge in perpetuity, to any
# person obtaining a copy of the Schema, to use, copy, modify, merge and
# distribute free of charge, copies of the Schema for the purposes of
# developing, implementing, installing and using software based on the
# Schema, and to permit persons to whom the Schema is furnished to do so,
# subject to the following conditions:

# THE SCHEMA IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
# IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
# FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
# THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR
# OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,
# ARISING FROM, OUT OF OR IN CONNECTION WITH THE SCHEMA OR THE USE OR
# OTHER DEALINGS IN THE SCHEMA.

# In addition, any modified copy of the Schema shall include the following
# notice:

# THIS SCHEMA HAS BEEN MODIFIED FROM THE SCHEMA DEFINED IN ISO/IEC 19757-3:2010,
# AND SHOULD NOT BE INTERPRETED AS COMPLYING WITH THAT STANDARD.".

default namespace sch = "http://purl.oclc.org/dsdl/schematron"

namespace local = ""

start = schema

# Element declarations
schema = element schema {
    attribute id { xsd:ID }?,
    rich,
    attribute schemaVersion { non-empty-string }?,
    attribute defaultPhase { xsd:IDREF }?,
    attribute queryBinding { non-empty-string }?,
    (foreign
     & inclusion*
     &  (title?,
       ns*,
       p*,
       let*,
       phase*,
       pattern+,
       p*,
       diagnostics?,
       properties))
}

active = element active {
    attribute pattern { xsd:IDREF },
```

```
    (foreign & (text | dir | emph | span)*)
}

assert = element assert {
    attribute test { exprValue },
    attribute flag { flagValue }?,
    attribute id { xsd:ID }?,
    attribute diagnostics { xsd:IDREFS }?,
    attribute properties  { xsd:IDREFS }?,
    rich,
    linkable,
    (foreign & (text | name | value-of | emph | dir | span)*)
}

diagnostic = element diagnostic {
    attribute id { xsd:ID },
    rich,
    (foreign & (text | value-of | emph | dir | span)*)
}

diagnostics = element diagnostics {
    foreign & inclusion* & diagnostic*
}

dir = element dir {
    attribute value { "ltr" | "rtl" }?,
    (foreign & text)
}

emph = element emph { text }

extends = element extends {
    attribute rule { xsd:IDREF },
    foreign-empty
}

let = element let {
    attribute name { nameValue },
    attribute value { string } |  foreign-element+
}

name = element name {
    attribute path { pathValue }?,
    foreign-empty
}

ns = element ns {
    attribute uri { uriValue },
    attribute prefix { nameValue },
    foreign-empty
}

p = element p {
    attribute id { xsd:ID }?,
    attribute class { classValue }?,
    attribute icon { uriValue }?,
    (foreign & (text | dir | emph | span)*)
}

param = element param {
    attribute name { nameValue },
    attribute value { non-empty-string }
}

pattern = element pattern {
    documents { pathValue }?,
    rich,
    (foreign & inclusion* &
```



```
     ( (attribute abstract { "true" }, attribute id { xsd:ID },
            title?, (p*, let*, rule*))
     |  (attribute abstract { "false" }?, attribute id { xsd:ID }?,
            title?, (p*, let*, rule*))
     | (attribute abstract { "false" }?, attribute is-a { xsd:IDREF },
            attribute id { xsd:ID }?, title?, (p*, param*))
     )
    )
}

phase = element phase {
    attribute id { xsd:ID },
    rich,
    (foreign & inclusion* & (p*, let*, active*))
}

properties = element properties { property* }

property = element property {
        attribute id { xsd:ID },
        attribute role { roleValue }?,
        attribute scheme { text }?,
        { foreign & (text | name | value-of | emph | dir | span)* }

report = element report {
    attribute test { exprValue },
    attribute flag { flagValue }?,
    attribute id { xsd:ID }?,
    attribute diagnostics { xsd:IDREFS }?,
    attribute properties  { xsd:IDREFS }?,
    rich,
    linkable,
    (foreign &  (text | name | value-of | emph | dir | span)*)
}

rule = element rule {
    attribute flag { flagValue }?,
    rich,
    linkable,
    (foreign & inclusion*
     & ((attribute abstract { "true" },
            attribute id { xsd:ID }, let*,
                (assert | report | extends| p )+)
        | (attribute context { pathValue },
            attribute id { xsd:ID }?,
           attribute abstract { "false" }?,
           let*, (assert | report | extends | p)+)))
}

span = element span {
    attribute class { classValue },
    (foreign & text)
}

title = element title {
    (text | dir)*
}

value-of = element value-of {
    attribute select { pathValue },
    foreign-empty
}

# common declarations
inclusion = element include {
    attribute href { uriValue }
} |
  element extends {
```

```
    attribute href { uriValue },
    foreign-empty }

rich =
    attribute icon { uriValue }?,
    attribute see { uriValue }?,
    attribute fpi { fpiValue }?,
    attribute xml:lang { langValue }?,
    attribute xml:space { "preserve" | "default" }?

linkable =
    attribute role { roleValue }?,
    attribute subject { pathValue }?

foreign =
    foreign-attributes, foreign-element*

foreign-empty =
    foreign-attributes

foreign-attributes =
    attribute * - (local:* | xml:*) { text }*

foreign-element =  element * - sch:*  {
    (attribute * { text }
     | foreign-element
     | schema
     | text)*
}

# Data types

uriValue = xsd:anyURI
pathValue = string
exprValue = string
fpiValue = string
langValue = xsd:language
roleValue = string
flagValue = string
nameValue = string  # In the default query language binding, xsd:NCNAME
classValue = string

non-empty-string = xsd:token { minLength = "1" }
```

Annex B
(normative)

Schematron Schema for Additional Constraints

A correct Schematron schema shall be valid with respect to the following Schematron schema.

```
<!--
       © ISO/IEC 2009

The following permission notice and disclaimer shall be included in all
copies of this XML schema ("the Schema"), and derivations of the Schema:

Permission is hereby granted, free of charge in perpetuity, to any
person obtaining a copy of the Schema, to use, copy, modify, merge and
distribute free of charge, copies of the Schema for the purposes of
developing, implementing, installing and using software based on the
Schema, and to permit persons to whom the Schema is furnished to do so,
subject to the following conditions:

THE SCHEMA IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR
OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,
ARISING FROM, OUT OF OR IN CONNECTION WITH THE SCHEMA OR THE USE OR
OTHER DEALINGS IN THE SCHEMA.

In addition, any modified copy of the Schema shall include the following
notice:

THIS SCHEMA HAS BEEN MODIFIED FROM THE SCHEMA DEFINED IN ISO/IEC 19757-3:2009,
AND SHOULD NOT BE INTERPRETED AS COMPLYING WITH THAT STANDARD."
 -->

<sch:schema xmlns:sch="http://purl.oclc.org/dsdl/schematron"
     xml:lang="en" >
    <sch:title>Schema for Additional Constraints in Schematron</sch:title>
    <sch:ns prefix="sch" uri="http://purl.oclc.org/dsdl/schematron" />

    <sch:p>This schema supplies some constraints in addition to those
    given in the ISO/IEC 19757-2 (RELAX NG Compact Syntax) Schema for Schematron.
    </sch:p>

    <sch:pattern>
        <sch:rule context="sch:active">
            <sch:assert
                test="//sch:pattern[@id=current()/@pattern]">
            The pattern attribute of the active element shall match the
            id attribute of a pattern.
            </sch:assert>
        </sch:rule>

        <sch:rule context="sch:pattern[@is-a]">
            <sch:assert
                test="//sch:pattern[@abstract='true'][@id=current()/@is-a]">
            The is-a attribute of a pattern element shall match
            the id attribute of an abstract pattern.
            </sch:assert>
        </sch:rule>

        <sch:rule context="sch:extends">
```

```
            <sch:assert
                test="//sch:rule[@abstract='true'][@id=current()/@rule]">
            The rule attribute of an extends element shall match
            the id attribute of an abstract rule.
            </sch:assert>
        </sch:rule>


        <sch:rule context="sch:let">
            <sch:assert
                test = "not(//sch:pattern
                    [@abstract='true']/sch:param[@name=current()/@name])">
            A variable name and an abstract pattern parameter should not
            use the same name.
            </sch:assert>
        </sch:rule>

    </sch:pattern>
</sch:schema>
```

Annex C
(normative)

Default Query Language Binding

A Schematron schema with no language binding or a `language` attribute with the value `xslt`, in any mix of upper and lower case letters, shall use the following binding:

— The query language used is the extended version of XPath specified in XSLT1. Consequently, the data model used is the data model of those specifications.

— The rule context is interpreted according to the Production 1 of XSLT1. The rule context may be root nodes, elements, attributes, comments and processing instructions. An implementation may allow the rule context to be text nodes at user option however implementations may reject or fail to implement schemas which specify text nodes.

— The assertion test is interpreted according to Production 14 of XPath, as returning a Boolean value.

— The name query is interpreted according to Production 14 of XPath, as returning a string value. Typically, the `select` attribute contains an expression returning an element node: the name query takes the local or prefixed name of the node, not its value.

— The value-of query is interpreted according to Production 14 of XPath, as returning a string value.

— The let value is interpreted according to Production 14 of XPath, as returning a string value.

— The documents attribute of the pattern element shall be interpreted according to the Production 1 of XSLT1, as returning one or more IRI strings that are evaluated using the `document()` function.

— The notation for signifying the use of parameter of an abstract pattern is to prefix the name token with the character. This is a character not found as a delimiter in URLs or XPaths. The character not followed by the name of an in-scope parameter shall not be treated as a parameter name delimiter. Such a character may subsequently be used as a delimiter for a variable name or as a literal character.

— A Schematron let expression is treated as an XSLT1 variable. The XSLT1 $ delimiter signifies the use of a variables in an context expression, assertion test, name query, value-of query or let expression. The character not followed by the name of an in-scope variable shall be treated as a literal character.

The XSLT1 `key` element may be used, in the XSLT1 namespace, before the pattern elements.

The XSLT1 `copy-of` element may be used, in the XSLT1 namespace and without child nodes, inside the property element.

The attributes `id`, `name` and `prefix` should follow the rules for non-colonized names for the version of XML used by the document.

Annex D
(informative)

Schematron Validation Report Language

D.1 Description

The Schematron Validation Report Language (SVRL) is a simple XML language which may be used for reporting the results of Schematron validation and for conformance suites.

The order of elements in an SVRL is implementation-dependent; different implementations may generate the same elements in a different order.

All elements shown in the grammar for Simple Validation Report Language are qualified with the namespace URI:

```
http://purl.oclc.org/dsdl/svrl
```

Elements related to the same subordinate document should be grouped together.

In subsequent schemas, the prefix `svrl` is taken as bound to the Simple Validation Report Language namespace URI for exposition purposes. The prefix `svrl` is not reserved or required by this part of ISO/IEC 19757.

D.2 RELAX NG Compact Syntax Schema

```
<!--
    © ISO/IEC 2009

The following permission notice and disclaimer shall be included in all
copies of this XML schema ("the Schema"), and derivations of the Schema:

Permission is hereby granted, free of charge in perpetuity, to any
person obtaining a copy of the Schema, to use, copy, modify, merge and
distribute free of charge, copies of the Schema for the purposes of
developing, implementing, installing and using software based on the
Schema, and to permit persons to whom the Schema is furnished to do so,
subject to the following conditions:

THE SCHEMA IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR
OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,
ARISING FROM, OUT OF OR IN CONNECTION WITH THE SCHEMA OR THE USE OR
OTHER DEALINGS IN THE SCHEMA.

In addition, any modified copy of the Schema shall include the following
notice:

THIS SCHEMA HAS BEEN MODIFIED FROM THE SCHEMA DEFINED IN ISO/IEC 19757-3:2009,
AND SHOULD NOT BE INTERPRETED AS COMPLYING WITH THAT STANDARD."
 -->
default namespace = "http://purl.oclc.org/dsdl/svrl"

schematron-output =
  element schematron-output {
          attribute title { text }?,
          attribute phase { xsd:NMTOKEN }?,
          attribute schemaVersion { text }?,
```



```
    human-text*,
    ns-prefix-in-attribute-values*,
    (active-pattern,
     (fired-rule, (failed-assert | successful-report)*)+)+
  }

# only namespaces from sch:ns need to be reported
ns-prefix-in-attribute-values =
  element ns-prefix-in-attribute-values {
    attribute prefix { xsd:NMTOKEN },
    attribute uri { text },
    empty
  }

# only active patterns are reported
active-pattern =
  element active-pattern {
          attribute id { xsd:ID }?,
          documents { text }?,
         attribute name { text }?,
         attribute role { xsd:NMTOKEN }?,
          empty
  }

# only rules that are fired are reported,
fired-rule =
  element fired-rule {
          attribute id { xsd:ID }?,
         attribute name { text }?,
          attribute context { text },
          attribute role { xsd:NMTOKEN }?,
          attribute flag { xsd:NMTOKEN }?,
        empty
  }

# only references are reported, not the diagnostic
diagnostic-reference =
  element diagnostic-reference {
          attribute diagnostic { xsd:NMTOKEN },
          human-text
  }
# only failed assertions are reported
failed-assert =
  element failed-assert {
    attlist.assert-and-report,
    diagnostic-reference*,
    property-reference*,
    human-text
  }

# only successful asserts are reported
successful-report =
  element successful-report {
    attlist.assert-and-report,
    diagnostic-reference*,
    property-reference*,
    human-text
  }

# property-reference
property-reference =
  element property-reference {
         attribute property { xsd:NMTOKEN },
         attribute role { text }?,
         attribute scheme { text }?,
        human-text
}
```

```
# human text
human-text =
  element text {
        attribute xml:space {text}?,
        attribute xml:lang {text}?,
        attribute see {text}?,
        attribute icon {text}?,
        attribute fpi { text }?,
        rich-text }


# rich text
rich-text =
   {(foreign | dir | span | emph | text )* }

# directionality
dir =
  element dir {
        attribute class {text}?,
        attribute dir {text}?,
        text }

# emphasis
emph =
  element emph {
        attribute class {text}?;
        text }

# arbitrary markup
span =
  element span {
        attribute class {text},
        text }

# foreign
foreign =
        foreign-attributes, foreign-element*

foreign-attributes =
    attribute * { text }*

foreign-element =
  element * - svrl:* {
    (attribute * { text }
    | foreign-element
    | text)*}

attlist.assert-and-report =
  attribute id { xsd:ID }?,
  attribute location { text },
  attribute test { text },
  attribute role { xsd:NMTOKEN }?,
  attribute flag { xsd:NMTOKEN }?

start = schematron-output
```

D.3 Schematron Schema

The corresponding Schematron schema is:

```
<!--
   © ISO/IEC 2009

The following permission notice and disclaimer shall be included in all
copies of this XML schema ("the Schema"), and derivations of the Schema:

Permission is hereby granted, free of charge in perpetuity, to any
```



```
person obtaining a copy of the Schema, to use, copy, modify, merge and
distribute free of charge, copies of the Schema for the purposes of
developing, implementing, installing and using software based on the
Schema, and to permit persons to whom the Schema is furnished to do so,
subject to the following conditions:

THE SCHEMA IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR
OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,
ARISING FROM, OUT OF OR IN CONNECTION WITH THE SCHEMA OR THE USE OR
OTHER DEALINGS IN THE SCHEMA.

In addition, any modified copy of the Schema shall include the following
notice:

THIS SCHEMA HAS BEEN MODIFIED FROM THE SCHEMA DEFINED IN ISO/IEC 19757-3:2009,
AND SHOULD NOT BE INTERPRETED AS COMPLYING WITH THAT STANDARD."
 -->

<sch:schema
    xmlns:sch="http://purl.oclc.org/dsdl/schematron"
    xml:lang="en" >

    <sch:title>Schema for Schematron Validation Report Language</sch:title>
    <sch:ns prefix="svrl" uri="http://purl.oclc.org/dsdl/svrl" />

    <sch:p>The Schematron Validation Report Language is a simple language
        for implementations to use to compare their conformance. It is
        basically a list of all the assertions that fail when validating
        a document, in any order, together with other information such
        as which rules fire.
    </sch:p>
    <sch:p>This schema can be used to validate SVRL documents, and provides
        examples of the use of abstract rules and abstract patterns.</sch:p>

    <sch:pattern>
        <sch:title>Elements</sch:title>

        <!--Abstract Rules -->
        <sch:rule abstract="true" id="second-level">
            <sch:assert test="../svrl:schematron-output">
                The <sch:name/> element is a child of schematron-output.
            </sch:assert>
        </sch:rule>

        <sch:rule abstract="true" id="childless">
            <sch:assert test="count(*)=0">
                The <sch:name/> element should not contain any elements.
            </sch:assert>
        </sch:rule>

        <sch:rule abstract="true" id="empty">
            <sch:extends rule="childless" />
            <sch:assert test="string-length(space-normalize(.)) = 0">
                The <sch:name/> element should be empty.
            </sch:assert>
        </sch:rule>

        <!-- Rules-->
        <sch:rule context="svrl:schematron-output">
            <sch:assert test="not(../*)">
                The <sch:name/> element is the root element.
            </sch:assert>
            <sch:assert
                test="count(svrl:text) + count(svrl:ns-prefix-in-attribute-values) +
                    count(svrl:active-pattern) +
```

```
            count(svrl:fired-rule) + count(svrl:failed-assert) +
            count(svrl:successful-report) = count(*)">
        <sch:name/> may only contain the following elements:
        text, ns-prefix-in-attribute-values, active-pattern, fired-rule,
        failed-assert and successful-report.
    </sch:assert>
    <sch:assert test="svrl:active-pattern">
        <sch:name/> should have at least one active pattern.
    </sch:assert>
</sch:rule>

<sch:rule context="svrl:text">
    <sch:extends rule="childless" />
</sch:rule>

<sch:rule context="svrl:diagnostic-reference">
    <sch:extends rule="childless" />
    <sch:assert test="string-length(@diagnostic) &gt; 0">
        <sch:name/> should have a diagnostic attribute,
        giving the id of the diagnostic.
    </sch:assert>
</sch:rule>

<sch:rule context="svrl:ns-prefix-in-attribute-values">
    <sch:extends rule="second-level" />
    <sch:extends rule="empty" />
    <sch:assert
        test="following-sibling::svrl:active-pattern
              or following-sibling::svrl:ns-prefix-in-attribute-value">
        A <sch:name/> comes before an active-pattern or another
        ns-prefix-in-attribute-values element.
    </sch:assert>
</sch:rule>

<sch:rule context="svrl:active-pattern">
    <sch:extends rule="second-level" />
    <sch:extends rule="empty" />
</sch:rule>

<sch:rule context="svrl:fired-rule">
    <sch:extends rule="second-level" />
    <sch:extends rule="empty" />
    <sch:assert
        test="preceding-sibling::active-pattern |
            preceding-sibling::svrl:fired-rule |
            preceding-sibling::svrl:failed-assert |
            preceding-sibling::svrl:successful-report">
        A <sch:name/> comes after an active-pattern, an empty
        fired-rule, a failed-assert or a successful report.
    </sch:assert>
    <sch:assert test="string-length(@context) &gt; 0">
        The <sch:name/> element should have a context attribute
        giving the current context, in simple XPath format.
    </sch:assert>
</sch:rule>

<sch:rule context="svrl:failed-assert | svrl:successful-report">
    <sch:extends rule="second-level" />
    <sch:assert
        test="count(svrl:diagnostic-reference) + count(svrl:text) = count(*)">
        The <sch:name/> element should only contain a text element
        and diagnostic reference elements.
    </sch:assert>
    <sch:assert test="count(svrl:text) = 1">
        The <sch:name/> element should only contain a text element.
    </sch:assert>
    <sch:assert test="preceding-sibling::svrl:fired-rule |
        preceding-sibling::svrl:failed-assert |
```

```
                preceding-sibling::svrl:successful-report">
                A <sch:name/> comes after a fired-rule, a failed-assert or a
                successful-report.
            </sch:assert>
        </sch:rule>

        <!-- Catch-all rule-->
        <sch:rule context="*">
            <sch:report test="true()">
                An unknown <sch:name/> element has been used.
            </sch:report>
        </sch:rule>
    </sch:pattern>

    <sch:pattern>
        <sch:title>Unique Ids</sch:title>

        <sch:rule context="*[@id]">
            <sch:assert test="not(preceding::*[@id=current()/@id][1]">
                Id attributes should be unique in a document.
            </sch:assert>
        </sch:rule>
    </sch:pattern>

    <sch:pattern abstract="true" id="requiredAttribute">
        <sch:title>Required Attributes</sch:title>

        <sch:rule context=" $context ">
            <sch:assert test="string-length( $attribute ) &gt; 0">
                The <sch:name/> element should have a
                <sch:value-of select="$attribute /name()" /> attribute.
            </sch:assert>
        </sch:rule>
    </sch:pattern>

    <sch:pattern is-a="requiredAttribute">
        <sch:param name="context" value="svrl:diagnostic-reference" />
        <sch:param name="attribute" value="@diagnostic" />
    </sch:pattern>

    <sch:pattern is-a="requiredAttribute">
        <sch:param name="context" value="svrl:failed-assert or svrl:successful-report" />
        <sch:param name="attribute" value="@location" />
    </sch:pattern>

    <sch:pattern is-a="requiredAttribute">
        <sch:param name="context" value="svrl:failed-assert or svrl:successful-report" />
        <sch:param name="attribute" value="@test" />
    </sch:pattern>

    <sch:pattern is-a="requiredAttribute">
        <sch:param name="context" value="svrl:ns-prefix-in-attribute-values" />
        <sch:param name="attribute" value="@uri" />
    </sch:pattern>

    <sch:pattern is-a="requiredAttribute">
        <sch:param name="context" value="svrl:ns-prefix-in-attribute-values" />
        <sch:param name="attribute" value="@prefix" />
    </sch:pattern>

</sch:schema>
```

Annex E
(informative)

Design Requirements

Motivating design requirements for the schema language with the default query language binding include:

— Represent abstract patterns such as the *head+body* pattern.

— Support progressive validation of compound documents.

— Include assertions or abstract rules from an external file.

— Support a one-to-one mapping between the natural-language statements and artificial-language statements.

— Support the generation and labelling of arcs between information items.

Motivating design requirements for the schema language with the default query language binding do not include:

— Simple specification of constraints that are simply expressed by grammar-based validation, such as ISO/IEC 19757-2 (RELAX NG) schemas.

— Replacement of any other standard schema language for XML.

— Single-pass or streamable implementation.

As well, certain outcomes are out-of-scope for this part of ISO/IEC 19757:

— Specification of a type system.

— Generation of links between multiple occurrences of some datum across multiple documents for the purpose of consistency-checking.

Motivating requirements for the changes introduces in this part of ISO/IEC 19757 include:

— Support more query language bindings, in particular XSLT2.

— Support the declaration of simple static and dynamic properties.

Annex F
(normative)

Use of Schematron as a Vocabulary

The following Schematron element types may be used as vocabulary elements by other standards and schemas. The semantics of element types used externally shall follow this part of ISO/IEC 19757.

— `schema`

— `pattern`

— `rule`

— `assert`

— `report`

These elements should use the standard Schematron namespace specified in 5.2.

When Schematron elements other than the `schema` element are used as vocabulary elements by other standards and schemas, the other standard or schema should specify mechanisms for defining information otherwise supplied by Schematron elements, such as `ns` or `let`.

Annex G
(informative)

Use of Schematron for Multi-Lingual Schemas

The following Schematron schema shows how multiple languages may be supported.

```
<sch:schema xmlns:sch="http://purl.oclc.org/dsdl/schematron"
     xml:lang="en" >
    <sch:title>Example of Multi-Lingual Schema</sch:title>

    <sch:pattern>
        <sch:rule context="dog">
            <sch:assert test="bone" diagnostics="d1 d2">
            A dog should have a bone.
            </sch:assert>
        </sch:rule>
    </sch:pattern>

    <sch:diagnostics>
        <sch:diagnostic id="d1" xml:lang="en">
        A dog should have a bone.
        </sch:diagnostic>
        <sch:diagnostic id="d2" xml:lang="de">
        Ein Hund sollte ein Bein haben.
        </sch:diagnostic>
    </sch:diagnostics>

</sch:schema>
```

Annex H
(normative)

Query Language Binding for XSLT 2

The `xslt2` query language binding allows schemas implemented using XSLT2. All implementations that support the `xslt2` query language binding should also support `xpath2` query language binding.

A Schematron schema with a `queryBinding` attribute with the value `xslt2`, in any mix of upper and lower case letters, shall use the following binding:

— The query language used is the extended version of XPath2 specified in XSLT2 with backwards compatability mode as false. Consequently, the data model used is the data model of XDM contructed from an infoset or a PSVI. All namespaces, prefixes, functions and operators defined by XPath2 Functions shall be available. An implementation may allow user-written functions and extensions, in the appropriate namespace.

— The rule context is interpreted according to the Production Production 1 of XSLT2. The rule context may be root nodes, elements, attributes, comments and processing instructions, including sequences of these. An implementation may allow the rule context to be text nodes at user option however implementations may reject or fail to implement schemas which specify text nodes.

— The assertion test is interpreted according to Production 1 of XPath2, using `fn:boolean()` on the result of evaluating the expression and to find the effective boolean value.

— The name query is interpreted according to Production 1 of XPath2, using `fn:node-name()` on the result of evaluating the expression which should be an element or attribute node and returning a string value.

— The value-of query is interpreted according to Production Production 1 of XPath2, using `fn:string()` on the result of evaluating the expression and returning a string value.

— The let value is interpreted according to Production Production 1 of XPath2.

— The documents attribute of the element shall be interpreted according to the Production 1 of XSLT2, as returning a sequence of strings that are evaluated using the `document()` function.

— The notation for signifying the use of parameter of an abstract pattern is to prefix the name token with the character. This is a character not found as a delimiter in URLs or XPaths. The character not followed by the name of an in-scope parameter shall not be treated as a parameter name delimiter. Such a character may subsequently be used as a delimiter for a variable name or as a literal character.

— A Schematron let expression is treated as an XSLT2 variable. The XSLT2 $ delimiter signifies the use of a variables in an context expression, assertion test, name query, value-of query or let expression. The character not followed by the name of an in-scope variable shall be treated as a literal character.

— All namespaces and prefixes defined by XSLT2 for modules are reserved with their XSLT2 usage. All functions defined by these modules shall be available in addition to the the functions defined by XPath2 Functions.

The XSLT2 `key` element may be used, in the XSLT2 namespace, before the pattern elements.

The XSLT2 `function` element may be used, in the XSLT2 namespace, before the pattern elements.

The XSLT2 `copy-of` element may be used, in the XSLT2 namespace and without child nodes, inside the property element.

The attributes `id`, `name` and `prefix` should follow the rules for non-colonized names for the version of XML used by the document.

The `fn:error()` function or other dynamic errors should not be used to provide assertion text or to substitute for assertions and diagnostics.

Annex I
(informative)

Query Language Binding for XPath 2

The `xpath2` query language binding allows schemas implemented using the simplest

A Schematron schema with a `queryBinding` attribute with the value `xpath2`, in any mix of upper and lower case letters, shall use the following binding:

— By default the query language used is that of XPath2 used with no XSD[5] schema and with backwards compatability mode as false. Consequently, the data model used is the data model of XDM contructed from an infoset not a post schema validation infoset. All namespaces, prefixes, functions and operators defined by XPath2 Functions shall be available. No user-written functions or extensions shall be available.

— The rule context is interpreted according to the Production Production 1 of XPath2. The rule context may be elements, attributes, comments and processing instructions, including sequences of these.

— The assertion test is interpreted according to Production 1 of XPath2, using `fn:boolean()` on the result of evaluating the expression and to find the effective boolean value.

— The name query is interpreted according to Production 1 of XPath2, using `fn:node-name()` on the result of evaluating the expression which should be an element or attribute node and returning a string value.

— The value-of query is interpreted according to Production Production 1 of XPath2, using `fn:string()` on the result of evaluating the expression and returning a string value.

— The let element should not be used.

The attributes `id`, `name` and `prefix` should follow the rules for non-colonized names for the version of XML used by the document.

The `fn:error()` function or other dynamic errors should not be used to provide assertion text or to substitute for assertions and diagnostics.

Annex J
(informative)

Query Language Binding for EXSLT

The `exslt` query language binding is the default `xslt` binding augmented by the EXSLT[1] functions.

A Schematron schema with a `queryBinding` attribute with the value `exslt`, in any mix of upper and lower case letters, shall use the following binding:

— The query language used is the extended version of XPath specified in XSLT1. Consequently, the data model used is the data model of those specifications.

— The rule context is interpreted according to the Production 1 of XSLT1. The rule context may be elements, attributes, comments and processing instructions.

— The assertion test is interpreted according to Production 14 of XPath, as returning a Boolean value.

— The name query is interpreted according to Production 14 of XPath, as returning a string value. Typically, the `select` attribute contains an expression returning an element node: the name query takes the local or prefixed name of the node, not its value.

— The value-of query is interpreted according to Production 14 of XPath, as returning a string value.

— The let value is interpreted according to Production 14 of XPath, as returning a string value.

— The documents attribute of the pattern element shall be interpreted according to the Production 1 of XSLT1, as returning one or more IRI strings that are evaluated using the `document()` function.

— The notation for signifying the use of parameter of an abstract pattern is to prefix the name token with the character. This is a character not found as a delimiter in URLs or XPaths. The character not followed by the name of an in-scope parameter shall not be treated as a parameter name delimiter. Such a character may subsequently be used as a delimiter for a variable name or as a literal character.

— A Schematron let expression is treated as an XSLT1 variable. The XSLT1 $ delimiter signifies the use of a variables in an context expression, assertion test, name query, value-of query or let expression. The character not followed by the name of an in-scope variable shall be treated as a literal character.

— All namespaces and prefixes defined by EXSLT[1] for modules are reserved with their EXSLT[1] usage. All functions defined by these modules shall be available in addition to the the functions defined by Production 14 of XPath and Production 1 of XSLT1

NOTE As EXSLT[1] does not require all implementations to provide all modules, some schemas using this query language binding will fail if they use functions which the implementation does not provide.

The XSLT1 `key` element may be used, in the XSLT1 namespace, before the pattern elements.

The attributes `id`, `name` and `prefix` should follow the rules for non-colonized names for the version of XML used by the document.

Annex K
(informative)

Query Language Binding for STX

The `stx` query language binding provides a schema langauge suited to memory-efficient streaming implementation.

A Schematron schema with a `queryBinding` attribute with the value `stx`, in any mix of upper and lower case letters, shall use the following binding:

— The query language used is the streaming dialect of XPath specified in STX[3]. Consequently, the data model used is the data model of STX[3] which is a variant on XPath data model noting that all elements and attributes are treated as untyped in STX[3].

— The rule context is interpreted according to the Production 2 of STX[3]. The rule context may be elements, attributes, comments and processing instructions.

— The assertion test is interpreted according to Production 1 of STX[3], as returning a Boolean value.

— The name query is interpreted according to Production 1 of STX[3], as returning a string value. Typically, the `select` attribute contains an expression returning an element node: the name query takes the local or prefixed name of the node, not its value.

— The value-of query is interpreted according to Production 1 of STX[3], as returning a string value.

— The let value is interpreted according to Production 1 of STX[3], as returning a string value.

— The notation for signifying the use of parameter of an abstract pattern is to prefix the name token with the character. This is a character not found as a delimiter in URLs or XPaths. The character not followed by the name of an in-scope parameter shall not be treated as a parameter name delimiter. Such a character may subsequently be used as a delimiter for a variable name or as a literal character.

— A Schematron let expression is treated as an STX[3] variable. The STX[3] $ delimiter signifies the use of a variables in an context expression, assertion test, name query, value-of query or let expression. The character not followed by the name of an in-scope variable shall be treated as a literal character.

The attributes `id`, `name` and `prefix` should follow the rules for non-colonized names for the version of XML used by the document.

The `message` element shall not be used to provide text for assertions and diagnostics.

Annex L
(informative)

Example usage of Schematron Properties

L.1 Running Example

In the following examples, the property elements given should be placed inside the properties element in the document below. It gives examples of properties usable by a validation application on an assert element and properties usable for information set augmentation on a report element.

```
<sch:schema ...>
  <sch:title>Example schema of annotation with</sch:title>
  <sch:pattern>
    <sch:rule context=”assetValue”>
      <sch:assert test=”true()” properties=”precision3 non-negative-decimal iso8859-1”>
        The element assetValue should be a non-negative decimal number, with three
        digits of precision, in Australian dollars.
      </sch:assert>
      <sch:report test=”true()”
            properties=”australianDollar precision3a thisElement thisValue ”>
        The element assetValue should be a non-negative decimal number, with three digits
        of precision, in Australian dollars.
      </sch:report>

   </sch:pattern>
   <sch:properties>
   ...
   </sch:properties>
</sch:schema>
```

In the following examples, the values of the role attributes are neither defined nor reserved by this standard.

L.2 Annotation with controlled vocabulary information

```
<sch:property id=”australianDollar” scheme=”ISO4217” role=”currency”>AUD</sch:property>
```

L.3 Annotation with static information

```
<sch:property id=”precision3” role=”precision” >3</sch:property>
```

L.4 Annotation with static structured information

```
<sch:property id=”precision3a” >
  <precision>3</precision>
</sch:property>
```

L.5 Type annotation with XSD Simple Type

```
<sch:property id=”non-negative-decimal” >
  <xsd:simpleType type=”xs:decimal”>
    <xsd:maxInclusive value=”0” />
  <xsd:fractionDigits value='2'/>
  </xsd:simpleType>
</sch:property>
```

L.6 Type annotation with dynamic information

```
<sch:property id=”thisValue” role=”value”>
  <sch:value-of select=”.” />
</sch:property>
```

L.7 Type annotation with dynamic structured information

This can be used to draw in elements as well as values from the instance.

```
<sch:property id=”thisElement” role=”contents”>
  <xslt:copy-of select=”.”  xmlns:xslt=”” />
</sch:property>
```

L.8 Annotation with ISO19757-7 character repertoire reference

```
<sch:property id=”iso8859-1” role=”repertoire”
   xmlns:cdrl=””>
  <cdrl:charref href=”http://www.eg.com/cdrl/iso8859-1.cdr” />
</sch:property>
```

L.9 SVRL document

Let the instance be the single element which conforms to the properties above.

```
 <assetValue>1000.00</assetValue>
```

The following is an SVRL document which transfers the properties from the report element.

```
<svrl:schematron-output xmlns:svrl=”http://purl.oclc.org/dsdl/svrl
”>
  <svrl:active-pattern />
  <fired-rule context=” assetValue” />
  <svrl:successful-report test=”false()” location=”/assetValue” >

    <svrl:property-reference
      property=”australianDollar” scheme=”ISO4217” role=”currency”>AUD</sch:property>

   <svrl:property-reference property= ”precision3a” >
      <precision>3</precision>
   </svrl:property-reference>

   <svrl:property-reference property =”thisValue” role=”value”>1000.00</property>

  <svrl:property-reference property=”thisElement” role=”contents”>
     <assetValue>1000.00</assetValue>
   </svrl:property>

    <svrl:text>The element assetValue should be a non-negative decimal number, with
    three digits of precision, in Australian dollars.</svrl:text>

  </svrl:successful-report>
</svrl:schematron-output>
```

Bibliography

[1] *EXSLT*, Community library, http://www.exslt.org/

[2] *Resource Description for Schematron (web page)*, Rick Jelliffe, Computing Centre, Academia Sinica, Taipei, http://purl.oclc.org/dsdl/schematron

[3] *Streaming Transformations for XML (STX) Version 1.0*, Public working draft, 27 April 2007, http://stx.sourceforge.net/documents/spec-stx-20070427.html

[4] *XML Constraint Specification Language*, José Carlos Leite Ramalho, Department of Informatics, School of Engineering, University of Minho, http://www.di.uminho.pt/~jcr/PROJS/xcsl-www/

[5] *XML Schema Part 1: Structures Second Edition*, W3C Recommendation, 28 October 2004, http://www.w3.org/TR/xmlschema-1/

