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.

— *Part 0: Overview*

— *Part 1: Interoperability framework*

— *Part 2: Grammar-based validation — RELAX NG*

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

— *Part 4: Selection of validation candidates*

— *Part 5: Datatypes*

— *Part 6: Path-based integrity constraints*

— *Part 7: Character reportoire validation*

— *Part 8: Declarative document manipulation*

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

Introduction

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 the use of ISO Schematron elements in external vocabularies.

— Finally, Clause 8 describes conformance requirements for implementations of ISO Schematron validators.

— Annex A is a normative annexes providing the Part 2 (RELAX NG) schema for ISO Schematron.

— Annex B is a normative annexes 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 annexes providing the default query language binding to XSLT.

— Annex D is a non-normative annex providing a DTD 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.

this part of ISO/IEC 19757 is based on the Schematron[1] assertion language. The `let` element is based on XCSL[2]. 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.

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 the 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 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.

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.

XPath, *XML Path Language (XPath) Version 1.0* , W3C Recommendation, http://www.w3.org/TR/1999/REC-xpath-19991116

XSLT, *XSL Transformations (XSLT) Version 1.0*, W3C Recommendation, http://www.w3.org/TR/1999/REC-xslt-19991116

3 Terms and definitions

The definitions of Part 1 and Part 2 also apply to this part of ISO/IEC 19757.

3.1 schema

specification of a set of XML documents

3.2 correct schema

schema that satisfies all the requirements of this committee-draft standard

3.3 good schema

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

3.4 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.

3.5 phase

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

3.6 active phase

one particular phase, whose patterns are used for validation

3.7 pattern

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

3.8 active pattern

a pattern belonging to the active phase

3.9 abstract pattern

a pattern in a rule to some extent parameterized

3.10 subject

a particular information item which matches the context expression of a rule and which corresponds to the object of interest of the natural-language assertions

3.11 assertion

a natural-language assertion with corresponding assertion test and ancilliary attributes

3.12 natural-language assertion

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

3.13 assertion test

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

3.14 rule

unordered collection of assertions with a rule-context expresssion and ancilliary attributes

3.15 rule context

a selection of elements; a rule is said to fire when an information item matches its query

3.16 rule-context expression

a query to specify subjects; a rule-context is said to match an information item when that information item that has 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

3.17 abstract rule

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

3.18 variable

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

3.19 diagnostic

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

3.20 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

3.21 implementation

an implementation of a Schematron validator

3.22 name

a token with no whitespace characters, possibly with other constraints given in the Schema for Schematron.

4 Notation

4.1 XPath

this part of ISO/IEC 19757 uses XPath to identify information items in Schematron schemas. In this part of ISO/IEC 19757 the prefix `sch` is bound to the Schematron namespace URI.

4.2 Predicate Logic

this part of ISO/IEC 19757 uses predicate logic to express the semantics of Schematron schema. The following functions are defined:

—
is member of, an infix relation, used in the set-operation sense.

— `x ∈ y`
If y is an element in a simplified schema, `x ∈ y` is defined here as the `child::x` path from context of subject y as defined by XPath.

— `x ∈ y`
If y is an element in a simplified schema, `x ∈ y` is defined here as the `child::x` path from context of subject y as defined by XPath.

— `x ∈ y`
If y is the instance being validated, `x ∈ y` is defined here as all the subjects (information items) in the instance that can be accessed by the query language, as specified in the query language binding.

— `x ∈ y`
If y is the name of the active active-phase with the reserved value `#ALL`, `x ∈ y` is defined here as the path `//sch:pattern`

— `x ∈ y`

  If y is the name of the active active-phase with the reserved value `#DEFAULT`, `x ∈ y` is defined here as the path `//sch:pattern[@id=/sch:schema/@default-phase]`

— `x ∈ y`

  If y is the name of the active active-phase, not being `#ALL` or `#DEFAULT`, `x ∈ y` is defined here as the path `../sch:pattern[@id=//sch:phase[id="y"]/active/@pattern]`

— `position( r )`
  the XSLT function `position()` of a rule `r` in its parent pattern

— `match ( r, s, d )`
  a function returning boolean provided by the query language binding: it returns true iff the subject s from the document d matches the context expression of rule `r`

— `assert ( t, s, d)`
  a function returning boolean provided by the query language binding: it returns true iff the assertion test `t` is true when applied to the subject s from the document d

5 Syntax

5.1 Namespace and Whitespace

All elements shown in the grammar are qualified with the namespace URI:

`http://www.ascc.net/xml/schematron`

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.

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.

Leading and trailing whitespace is allowed for the value of any Schematron-namespace attribute, and shall be stripped. Whitespace in Schematron-namespace elements should be collapsed.

5.2 Core Elements

5.2.1 `active` element

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

5.2.2 `assert` element

An assertion made about the context nodes. The data context 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 1: The natural-language assertion may contain information about actual values in addition to expected values and may contain diagnostic information, however 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.

5.2.3 `extends` element

Abstract rules are name lists of assertions without a context expression. The required `rule` attribute references an abstract rule. The current rule uses all the assertions from the abstract rule it extends.

5.2.4 `include` element

The required `src` attribute references an external XML document with a schema fragment. The schema fragment contains elements which are valid when substituted for the `include` element.

5.2.5 `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 constant. The required `value` attribute is an expression evaluated in the current context that returns a string.

It is an error to reference a variable that has not been defined in the current schema, phase, pattern, or rule. 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 they are used. The query language binding specifies which lexical conventions are used to detect references to constants.

5.2.6 name element

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

The name is replaced by the string result of evaluating the `select` attribute in the current context.

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

5.2.7 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.

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.

NOTE 2: Namespace prefixes should not use the namespace bindings in scope for element and attribute names.

NOTE 3: When elements other than `schema` are used for external vocabularies, the namespace bindings shall be provided by that external vocabulary.

5.2.8 `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.

5.2.9 `pattern` element

A structure, simple or complex. A set of rules giving constraints that are in some way related. The required `name` attribute is the name of the pattern.

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

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

5.2.10 phase element

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

The name `#ALL` is reserved and available for use by implementations as a phase name, denoting that all patterns are active. The name `#DEFAULT` is reserved and available for use by implementations as a phase name, denoting that the name given in in the `phase` attribute on the `schema` element should be used. If there is no such attribute, or implementation-provided phases name supplied, all patterns are active.

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

5.2.11 `report` element

An assertion made about the context nodes. The data context 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 4: The natural-language assertion may contain information about actual values in addition to expected values and may contain diagnostic information, however 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.

5.2.12 `rule` element

A list of assertions tested within a context. The `context` attribute provides the rule context expression.

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

5.2.13 schema element

The top-level element of a Schematron schema. `empty`

The optional `version` attribute gives the version of Schematron to distinguish earlier versions of Schematron. Its value, if supplied, must be "`ISO`".

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

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

5.2.14 `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 either a string.

Variable references in the string 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.

5.3 Ancilliary Elements and Attributes

5.3.1 `diagnostics` element

Section containing invidual diagnostic elements.

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

5.3.2 `diagnostic` element

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

NOTE 5: 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 `diagnostics` elements in the `diagnostics` attribute of the assertion.

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

5.3.3 `dir` element

A section of natural-language text with a direction specified by the `dir` 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.

5.3.4 emph element

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

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

5.3.5 `flag` attribute

A boolean variable with 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.

5.3.6 `fpi` attribute

A formal public identifier for the schema or phase.

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

5.3.7 `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.

5.3.8 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.

5.3.9 `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.

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

5.3.10 see attribute

The URI of some external information of interest to maintainers and users of the schema.

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

5.3.11 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.

5.3.12 `subject` attribute

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

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

5.3.13 `title` element

A summary of the purpose or role of the schema or pattern.

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 performs notationally performs two steps: transforming the schema into a simple syntax, then testing the instance against the simple syntax.

NOTE 6: 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 customized 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 phase attribute on the schema element shall be used

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

6.2 Simplified Syntax

To simplifying the specification of semantics later, the following transformation steps are first applied to a schema, resulting in a schema in the simplified 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

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

— Negate all sch:report elements into sch:assert elements

The resulting simplified syntax is also a valid Schematron instance in the full syntax. The simplified 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`

6.3 Schema Semantics

This clause gives the semantics of a good schema that has been transformed into the simple syntax.

A good schema with no use of variables satisfies the following predicate:

```
      ( instance, schema, active-phase ),
 ∀( subject, pattern, rule, assertion ) :
  subject ∈ instance,
  subject ∈ schema,
  pattern ∈ active-phase,
  rule ∈ pattern,
  assertion ∈ rule :
   match ( rule, subject, instance )
    ∧ ( ∀(previous-rule ) :
     previous-rule ∈ pattern,
     position (previous-rule ) < position( rule ) :
      ¬ ( match ( previous-rule, subject, instance )))
    ⇒ assert ( assertion,  subject, instance ) = true
```

NOTE 7: In natural language, that is "*There exists an instance, schema and active-phase combination where, for each subject, pattern, rule and assertion (the subject being a member of that instance, the pattern being a member of that 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 subject in an instance matches the rule, and that subject has not been matched by a previous rule in the same pattern, then the particular assertion evaluates to true when applied to the particular subject 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 coerceable into boolean.

NOTE 8: The following other query language names are reserved without further definition. Implementions which use different query language bindings are encouraged to use one of these names if appropriate : `stx`, `xslt` `xslt1.1`, `exslt`, `xslt2`, `xpath`, `xpath2`, `xquery`.

A Schematron implementation which does not support the specification language shall fail with an error.

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 coerceable into a string.

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

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

— 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 recognized.

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

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

A Schematron implementation which does not support the specification language 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 elaborated rule-contexts are matched 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 sch:rule and sch:let elements.

An sch:rule element acts as an if-then-else statement within each pattern. An implementation may make order non-significant by converting rules contexts to elaborated rule contexts. An elaborated rule context consists the negated union of all the lexically previous rule contexts in the same pattern interected with the current rule context.

NOTE 9: The behaviour of the sch:rule element allows consraints that would require a complex context expression to be factored into simpler expressions in different rules.

An sch:let element may use lexically previous variables within the same rule or global variables.

NOTE 10: 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.

The only element which has a side-effect is key, which may provide extra index information for other queries.

7 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. Where an element requires other preceding or ancestor element to be complete, the information supplied by those elements shall be provided by the other standard or schema.

— `schema`

— `pattern`

— `rule`

— `assert`

— `report`

8 Conformance

8.1 Simple Conformance

A simple-conformance implementation shall be able to report for any XML document whether it may not be a valid Schematron schema.

— A valid schema conforms to the constraints of Annex A, the normative Part 2 (RELAX NG) 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 11: 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 12: 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.

8.2 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 Part 2 (RELAX NG) 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.

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 13: 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 14: 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.

Annex A
(normative)

RELAX NG schema for Schematron

A correct Schematron schema shall be valid with respect to the following Part 2 (RELAX NG) schema.

```
#    (c) International Organization for Standardization 2005.
#    Permission to copy in any form is granted for use with conforming
#    SGML systems and applications as defined in ISO 8879,
#    provided this notice is included in all copies.

# Based on
# http://www.ascc.net/xml/schematron/schematron1-5.dtd, version of 2002/08/16

# James' RNC for Schematron 1.5 with:
# key removed
#schema/@ns removed
# include added
#abstract patterns and param added
#let added
# subject added with role in linkable production
# flag added
# pattern/name optional
# attribute see and fpi on more elements, with rich production
# patterns have titles rather than names
# declarations for xml:space and xml:lang


default namespace sch = "http://www.ascc.net/xml/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 version { non-empty-string }?,
    attribute language { non-empty-string }?,
    (foreign
     & inclusion*
     &  (title?,
        ns*,
        p*,
        let*,
        phase*,
        pattern+,
        p*,
        diagnostics?))
  }
```

```
active =
  element active {
    attribute pattern { xsd:IDREF },
    (foreign & (text | dir | emph | span)*)
  }
assert =
  element assert {
    attribute test { expr },
    attribute flag { flag }?,
    attribute id { xsd:ID }?,
    attribute diagnostics { xsd:IDREFS }?,
    rich,
    linkable,
    (foreign & (text | name | 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 { name },
  attribute value { string }
}
name =
  element name {
    attribute path { path }?,
    foreign-empty
  }
ns =
  element ns {
    attribute uri { uri },
    attribute prefix { xsd:NMTOKEN }?,
    foreign-empty
  }
p =
  element p {
    attribute id { xsd:ID }?,
    attribute class { class }?,
    attribute icon { uri }?,
    (foreign & (text | dir | emph | span)*)
  }
param =
 element param {
  attribute name { name },
  attribute value { non-empty-string }
}
pattern =
  element pattern {
    attribute id { xsd:ID }?,
    rich,
    (foreign & inclusion* &
     ( (attribute abstract { "true" }, title?, (p*, let*, rule*))
     |  (attribute abstract { "false" }?, title?, (p*, let*, rule*))
     | (attribute abstract { "false" }?, attribute is-a { xsd:IDREF }, title?, (p*, param*)
     )
  }
phase =
  element phase {
    attribute id { xsd:ID },
    rich,
    (foreign & inclusion* & (p*, let*, active*))
  }
report =
  element report {
    attribute test { expr },
    attribute flag { flag }?,
    attribute id { xsd:ID }?,
    attribute diagnostics { xsd:IDREFS }?,
    rich,
    linkable,
    (foreign &  (text | name | emph | dir | span)*)
  }
rule =
  element rule {
    attribute id { xsd:ID }?,
    rich,
    linkable,
    (foreign & inclusion*
     & ((attribute abstract { "true" }, let*, (assert | report | extends)+)
        | (attribute context { path },
           attribute abstract { "false" }?,
           let*, (assert | report | extends)+)))
  }
span =
  element span {
    attribute class { class },
    (foreign & text)
  }
title = element title { (text | dir)* }
value-of =
  element value-of {
    attribute select { path },
    foreign-empty
  }

# common declarations
rich =
    attribute icon { uri }?,
    attribute see { uri }?,
    attribute fpi { fpi }?,
    attribute xml:lang { text }?,
    attribute xml:space { "preserve" | "default" }?

inclusions =
 element include {
  attribute src { uri }
}

linkable =
    attribute role { role }?,
    attribute subject { path }?

foreign = foreign-attributes, foreign-element*

foreign-empty = foreign-attributes

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

# Data types

uri = xsd:anyURI
path = string
expr = string
fpi = string
lang = xsd:language

role = xsd:NCName
flag = xsd:NCName
name = xsd:NCName
class = xsd:NCName

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 schema.

```
<!--
    (c) International Organization for Standardization 2005.
    Permission to copy in any form is granted for use with conforming
    SGML systems and applications as defined in ISO 8879,
    provided this notice is included in all copies.
 -->

<sch:schema xmlns:sch="http://www.ascc.net/xml/schematron"
 version="ISO" xml:lang="en" >
 <sch:title>Schema for Additional Constraints in Schematron</sch:title>
 <sch:ns prefix="sch" uri="http://www.ascc.net/xml/schematron" />

 <sch:p>This schema supplies some constraints in addition to those
 given in the ISO RELAX NG Schema for Schematron.
 </sch:p>

 <sch:pattern>
  <sch:rule context="sch:schema">
   <sch:assert text="not(@version) or @version='ISO'">
   The version attribute on the schema element, if present, should
   have the value "ISO"
   </sch:assert>
  </sch:rule>

  <sch:rule context="sch:active">
   <sch:assert test="//sch:pattern[@id=current()/@pattern]">
   The pattern attribute of the active element shall match an
   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: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 XSLT. Consequently, the data model used is the data model of those specifications.

— The rule context is interpreted according to the production 1 of XSLT. 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.

— 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.

— A Schematron let expression is treated as an XSLT variable. The XSLT $ delimiter signifies the use of a variables in an assertion test, name query or value-of query.

— The notation for signifying abstract pattern is to prefix the token with the **$** character. This is a character not found in URLs or XPaths.

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

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. this part of ISO/IEC 19757 does not specify such a conformance suite.

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

D.2 DTD

```
<!--
    (c) International Organization for Standardization 2005.
    Permission to copy in any form is granted for use with conforming
    SGML systems and applications as defined in ISO 8879,
    provided this notice is included in all copies.
 -->

  <!ELEMENT schematron-output
        ( text*, ns*, (active-pattern, (fired-rule, ( failed-assert | successful-report)* )

   <!-- only active patterns are reported -->
   <!ELEMENT active-pattern  EMPTY>
   <!-- only references are repor/08, not the diagnostic -->
   <!ELEMENT diagnostic-reference (#PCDATA)  >
   <!-- only failed assertions are reported -->
   <!ELEMENT failed-assert  ( diagnostic-reference*, text )>
   <!-- only rules that are fired are reported, abstract/extend handling
            should have been done before -->
   <!ELEMENT fired-rule EMPTY >
   <!-- only namespaces from sch:ns need to be reported -->
   <!ELEMENT ns EMPTY >
   <!-- only successful asserts are reported -->
   <!ELEMENT successful-report (  diagnostic-reference*, text ) >
   <!ELEMENT text (#PCDATA )>

    <!ATTLIST schematron-output
        title CDATA #IMPLIED
        phase NMTOKEN #IMPLIED
 schemaVersion CDATA #IMPLIED >

   <!ATTLIST active-pattern
          id ID #IMPLIED
          name CDATA #IMPLIED
          role NMTOKEN #IMPLIED >

   <!ATTLIST diagnostic-reference
         diagnostic NMTOKEN #REQUIRED >
   <!ATTLIST failed-assert
          id ID #IMPLIED
          location CDATA #REQUIRED
          test CDATA #REQUIRED
          role NMTOKEN #IMPLIED
          flag NMTOKEN #IMPLIED >

   <!ATTLIST fired-rule
          id ID #IMPLIED
          context CDATA #REQUIRED
          role NMTOKEN #IMPLIED
          flag NMTOKEN #IMPLIED >

   <!ATTLIST ns
          prefix NMTOKEN #REQUIRED
          uri  CDATA #REQUIRED >

   <!ATTLIST successful-report
          id ID #IMPLIED
          location CDATA #REQUIRED
          test  CDATA #REQUIRED
          role NMTOKEN #IMPLIED
          flag NMTOKEN #IMPLIED >
```

D.3 Schematron Schema

The corresponding Schematron schema is:

```
<!--
    (c) International Organization for Standardization 2005.
    Permission to copy in any form is granted for use with conforming
    SGML systems and applications as defined in ISO 8879,
    provided this notice is included in all copies.
 -->

 <sch:schema xmlns:sch="http://www.ascc.net/xml/schematron"
 version="ISO" xml:lang="en" >
 <sch:title>Schema for Schematron Report Validation Language</sch:title>

 <sch:p>The Schematron Report Validation 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 SRVL 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" name="second-level">
   <sch:assert test="../schematron-output">
    The <sch:name/> element is a child of schematron-output
   </sch:assert>
  </sch:rule>
```

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

<sch:rule abstract="true" name="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="schematron-output">
 <sch:assert test="not(../*)">
  The <sch:name/> element is the root element.
 </sch:assert>
 <sch:assert test="count(text) + count(ns) + count(active-pattern) +
  count(fired-rule) + count(failed-assert) +
  count(successful-report) = count(*)">
  <sch:name/> may only contain the following elements: text, ns,
  active-pattern, fired-rule, failed-assert and successful-report.
 </sch:assert>
 <sch:assert test="active-pattern">
  <sch:name/> should have at least one active pattern
 </sch:assert>
</sch:rule>

<sch:rule context="text">
 <sch:extends rule="childless" />
</sch:rule>
<sch:rule context="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="ns">
 <sch:extends rule="second-level" />
 <sch:extends rule="empty" />
 <sch:assert test="following-sibling::active-pattern or
  following-sibling::ns">
  A <sch:name/> comes before an active-pattern or another ns element
 </sch:assert>
</sch:rule>
<sch:rule context="active-pattern">
 <sch:extends rule="second-level" />
 <sch:extends rule="empty" />
</sch:rule>
<sch:rule context="fired-rule">
 <sch:extends rule="second-level" />
 <sch:extends rule="empty" />
 <sch:assert test="preceding-sibling::active-pattern |
  preceding-sibling::fired-rule |
   preceding-sibling::failed-assert |
  preceding-sibling::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="failed-assert | successful-report">
   <sch:extends rule="second-level" />
   <sch:assert test="count(diagnostic-reference) + count(text) = count(*)">
    The <sch:name/> element should only contain a text element and diagnostic
    reference elements.
   </sch:assert>
   <sch:assert test="count(text) = 1">
    The <sch:name/> element should only contain a text element.
   </sch:assert>
   <sch:assert test="preceding-sibling::fired-rule |
    preceding-sibling::failed-assert |
    preceding-sibling::successful-report">
    A <sch:name/> comes after a fired-rule, a failed-assert or a
    succesful-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" name="requiredAttribute>
  <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:pattern>

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

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

 <sch:pattern is-a="requiredAttribute">
```

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

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


 <sch:pattern is-a="requiredAttribute">
  <sch:param name="context" value="ns" />
  <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

— Include assertions or abstract rules from an external file

— Support a line-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 contraints that are simply expressed by grammar-based validation, such as Part 2 (RELAX NG) schemas or DTDs.

— 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 data across multiple documents for the purpose of consistency-checking.

Bibliography

[1] *Resource Description for Schematron (web page)*, Rick Jelliffe, Computing Centre, Academia Sinica, Taipei, http://www.ascc.net/xml/schematron

[2] *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/

Summary of editorial comments: