INTERNATIONAL STANDARD

ISO/IEC 19757-3

Third edition
2020-06

Information technology — Document Schema Definition Languages (DSDL) —

Part 3: Rule-based validation using Schematron

Technologies de l'information — Langages de définition de schéma de documents (DSDL) —

Partie 3: Validation basée sur des règles à l'aide de Schematron

Reference number
ISO/IEC 19757-3:2020(E)

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Fax: +41 22 749 09 47
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

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.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents) or the IEC list of patent declarations received (see http://patents.iec.ch).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT), see www.iso.org/iso/foreword.html.

This document was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 34, *Document description and processing languages.*

This third edition cancels and replaces the second edition (ISO/IEC 19757-3:2016), which has been technically revised.

The main changes compared to the previous edition are as follows:

— query language bindings have been added for XSLT 3.0 (Annex J) and XPath 3.0 (Annex K);

— annexes pertaining to XPath and XSLT query language bindings (Annexes H to K) are now all normative, while those for EXSLT (Annex L) and STX (Annex M) remain informative.

A list of all parts in the ISO/IEC 19757 series can be found on the ISO website.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html.

Introduction

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

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 the industry.

Through the validation that a structured document conforms to specified constraints in structure and content, the potentially many applications acting on the document are relieved from duplicating 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 can enhance validation objectives.

The main objective of ISO/IEC 19757 (all parts) 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 (all parts) allows different validation technologies to be integrated under a well-defined validation policy.

The structure of this document 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 provides the ISO/IEC 19757-2 (RELAX NG) schema for ISO Schematron. Annex B provides the ISO Schematron schema for constraints in ISO Schematron that cannot be expressed by the schema of Annex A. Annex C provides the default query language binding to XSLT1. Annex D provides an 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 provides motivating design requirements for ISO Schematron. Annex F specifies certain Schematron elements to be used in external vocabularies. Annex G provides a simple example of a multi-lingual schema. Annexes H to M provide query language bindings. Annex N shows example usage of Schematron properties.

This edition is backwards compatible with ISO/IEC 19757-3:2016, supersedes it and provides extra query language bindings, in particular for XSLT3.

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 ISO/IEC 19757-2 cannot specify and various dynamic or contingent constraints.

This document 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.

Information technology — Document Schema Definition Languages (DSDL) —

Part 3:
Rule-based validation using Schematron

1 Scope

This document specifies Schematron, a schema language for XML. This document establishes requirements for Schematron schemas and specifies when an XML document matches the patterns specified by a Schematron schema. Schematron uses query languages such as XPath for writing assertions.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements 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.

XPath[1], *XML Path Language (XPath) Version 1.0*, W3C Recommendation, 16 November 1999

XPath2[2], *XML Path Language (XPath) 2.0*, W3C Recommendation, 23 January 2007

XPath3[3], *XML Path Language (XPath) 3.0*, W3C Recommendation, 8 April 2014

XPath2 Functions[4], *XQuery 1.0 and XPath 2.0 Functions and Operators*, W3C Recommendation, 23 January 2007

XPath3 Functions[5], *XPath and XQuery Functions and Operators 3.0*, W3C Recommendation, 8 April 2014

XSLT1[6], *XSL Transformations (XSLT) Version 1.0*, W3C Recommendation, 16 November 1999

XSLT2[7], *XSL Transformations (XSLT) Version 2.0*, W3C Recommendation, 23 January 2007

XSLT3[8], *XSL Transformations (XSLT) Version 3.0*, W3C Recommendation, 8 June 2017

ISO/IEC 19757-2, *Information technology — Document Schema Definition Language (DSDL) — Part 2: Regular-grammar-based validation — RELAX NG*

3 Terms and definitions

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

1) Available at http://www.w3.org/TR/xpath.
2) Available at http://www.w3.org/TR/xpath20/.
3) Available at https://www.w3.org/TR/xpath-30/.
4) Available at http://www.w3.org/TR/xpath-functions/.
5) Available at https://www.w3.org/TR/xpath-functions-30/.
6) Available at http://www.w3.org/TR/xslt.
7) Available at http://www.w3.org/TR/xslt20/.
8) Available at .https://www.w3.org/TR/xslt-30/.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at https://www.iso.org/obp

— IEC Electropedia: available at http://www.electropedia.org/

3.1
abstract pattern
pattern (3.13) in a *rule* (3.18) that has been parameterized to enable reuse

3.2
abstract rule
collection of *assertions* (3.5) which can be included in other *rules* (3.18) but which does not fire itself

3.3
active pattern
pattern (3.13) belonging to the *active phase* (3.4)

3.4
active phase
phase (3.14) whose *patterns* (3.13) are used for validation

3.5
assertion
natural-language statement with associated *assertion test* (3.6) and ancillary attributes

3.6
assertion test
Boolean query

Note 1 to entry: An assertion test "succeeds" or "fails".

3.7
Schematron schema
document that satisfies all the requirements of this document

3.8
diagnostic
named natural language statements providing information to end-users of validators concerning the expected and actual values together with repair hints

3.9
elaborated rule-context expression
single *rule-context expression* (3.20) which explicitly disallows items selected by lexically previous *rule contexts* (3.19) in the same *pattern* (3.13)

3.10
implementation
implementation of a Schematron validator

3.11
name
mixture of name characters with a restricted set of initial characters

Note 1 to entry: See Production 5 of XML1.

3.12
natural-language assertion
natural-language statement expressing some part of a *pattern* (3.13)

Note 1 to entry: A natural-language assertion is "met" or "unmet".

3.13
pattern
unordered collection of *rules* (3.18) with an optional identifier and ancillary information

3.14
phase
named, unordered collection of *patterns* (3.13)

Note 1 to entry: Patterns may belong to more than one phase.

Note 2 to entry: Two strings, `#ALL` and `#DEFAULT`, are reserved with particular meanings.

3.15
progressive validation
validation of constraints in stages determined or grouped to some extent by the schema author rather than, for example, entirely determined by the document order

3.16
property
named data giving additional metadata on an *assertion* (3.5) or report

3.17
query language binding
named set, specified in a document called a Query Language Binding, of the languages and conventions used for *assertion tests* (3.6), *rule-context expressions* (3.20) and so on, by a particular Schematron *implementation* (3.10)

3.18
rule
unordered collection of *assertions* (3.5) with a *rule-context expression* (3.20) and ancillary attributes

3.19
rule context
element or other information item used for *assertion tests* (3.6)

Note 1 to entry: A rule is said to fire when an information item matches the rule context.

3.20
rule-context expression
query to specify *subjects* (3.21)

Note 1 to entry: A *rule context* (3.19) 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* (3.13) and the information item is one of the information items that the query would specify.

3.21
subject
particular information item which corresponds to the object of interest of the *natural-language assertions* (3.12) and typically is matched by the context expression of a *rule* (3.18)

3.22
valid
<schema> passing all *assertion tests* (3.6) in fired *rules* (3.18) of *active patterns* (3.3)

3.23
variable
constant value, represented by a *name* (3.11), evaluated within the parent schema, *phase* (3.14), *pattern* (3.13) or *rule* (3.18) and scoped within the parent schema, phase, pattern or rule

4 Notation

This document uses XPath to identify information items, to the extent that items in the XPath data model can be derived from those defined by XML Infoset, in Schematron schemas.

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 document. 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 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 should be stripped from attributes defined by this document. Whitespace should be collapsed in elements defined by this document that allow text. Whitespace may be stripped from elements defined by this document that do not allow text.

5.4 Core elements

5.4.1 General

A Schematron schema shall follow the grammar given in Annex A.

For information on the use of Schematron elements in external vocabularies, see Annex F.

5.4.2 `active` element

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

5.4.3 `assert` element

This element indicates 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 assertion succeeds. The optional `diagnostics` attribute is a reference to further diagnostic information.

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

The `icon`, `see` and `fpi` attributes (5.5.7, 5.5.12 and 5.5.6) allow rich interfaces and documentation.

The `flag` attribute (5.5.5) allows more detailed outcomes.

The `role` and `subject` attributes (5.5.11 and 5.5.14) allow explicit identification of some part of a pattern.

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.

5.4.4 `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` element 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 an external well-formed XML document that is a 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 in Annex A; however, other schema constraints such as containment shall still apply.

5.4.5 `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.

5.4.6 `let` element

The `let` element declares 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.

5.4.7 `name` element

The `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. If the `path` attribute is omitted, the name of the context node is used.

5.4.8 `ns` element

The `ns` element specifies 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].

Namespace prefixes in context expressions, assertion tests and other query expressions shall be defined by this element. Namespace prefixes should not use the namespace bindings in scope for element and attribute names.

NOTE 1 Namespace declarations as specified in XML 1.0 Namespaces (e.g. `xmlns:foo="http://www.example.com"`) do not apply to namespace prefixes in context expressions, assertion tests and other query expressions.

NOTE 2 Because the characters allowed as names can 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.

5.4.9 `param` element

The `param` element specifies 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.

5.4.10 `pattern` element

The `id` attribute provides a unique name for the pattern and is required for abstract patterns. The child `rule` elements of a pattern give constraints that are in some way related.

The optional `documents` attribute provides IRIs of the subordinate documents 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.

The `icon`, `see` and `fpi` attributes (5.5.7, 5.5.12 and 5.5.6) 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 an `is-a` attribute and shall have an `id` attribute.

5.4.11 `phase` element

The `phase` element provides the means to declare named groups of patterns, for example to support progressive validation. Additional information on design requirements for progressive validation is provided in Annex E.

The required `id` attribute is the name of the phase. The element specifies the phase to be used for validating documents, for example, by user command.

Two strings, `#ALL` and `#DEFAULT`, have special meanings when specifying active phases. The string `#ALL` is reserved to denote that all patterns are active. The string `#DEFAULT` is 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.

The `icon`, `see` and `fpi` attributes (5.5.7, 5.5.12 and 5.5.6) allow rich interfaces and documentation.

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

5.4.12 `report` element

The `report` element specifies 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.

The `icon`, `see` and `fpi` attributes (5.5.7, 5.5.12 and 5.5.6) allow rich interfaces and documentation.

The `flag` attribute (5.5.5) allows more detailed outcomes.

The `role` and `subject` attributes (5.5.11 and 5.5.14) allow explicit identification of some part of a pattern.

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.

5.4.13 `rule` element

The `rule` element provides a list of assertions tested within the context specified by the `context` attribute. The `context` attribute, required unless the `rule` element has the attribute `abstract` with a value `true`, specifies the rule-context expression.

The `icon`, `see` and `fpi` attributes (5.5.7, 5.5.12 and 5.5.6) allow rich interfaces and documentation.

The `flag` attribute (5.5.5) allows more detailed outcomes.

The `role` and `subject` attributes (5.5.11 and 5.5.14) allow explicit identification of some part of a pattern as part of the validation outcome.

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.

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.

5.4.14 `schema` element

The `schema` element is 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 document and its use is implementation-dependent.

The optional `queryBinding` attribute provides the short name of the query language binding in use. The `defaultPhase` attribute may be used to indicate the phase to use in the absence of explicit user-supplied information.

The `icon`, `see` and `fpi` attributes (5.5.7, 5.5.12 and 5.5.6) allow rich interfaces and documentation.

5.4.15 `value-of` element

The `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.

5.5 Ancillary elements and attributes

5.5.1 `diagnostic` element

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

NOTE 1 Diagnostics in multiple languages can 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.

NOTE 2 Typical values for the `role` attribute on a `diagnostic` element can be `warning`, `caution` or `note`.

5.5.2 `diagnostics` element

The `diagnostics` element provides a section containing individual `diagnostic` elements.

5.5.3 `dir` element

The `dir` element provides 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.

5.5.4 `emph` element

The `emph` element encloses a portion of text that should be rendered with some emphasis.

5.5.5 `flag` attribute

The `flag` attribute acts as a Boolean variable with the 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.

5.5.6 `fpi` attribute

The `fpi` attribute contains a formal public identifier for the schema, phase or other element.

5.5.7 `icon` attribute

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

5.5.8 `p` element

The `p` element represents 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.

5.5.9 `properties` element

The `properties` element provides a section containing individual `property` elements.

5.5.10 `property` element

The `property` element declares additional arbitrary properties that will be associated with failed assertions and successful reports.

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

NOTE 1 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.

NOTE 2 Where the property value contains elements in a well-known namespace or where the scheme used is otherwise obvious or unnecessary, the `scheme` attribute can be omitted. For example, if the `property` element contains an ISO/IEC 19757-7 Character Repertoire Description Language schema, no `scheme` attribute is appropriate.

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

NOTE 3 Properties do not participate in the validation of constraints.

For additional information on Schematron properties, see Annex N.

5.5.11 `role` attribute

The `role` attribute is 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.

5.5.12 `see` attribute

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

5.5.13 `span` element

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

5.5.14 `subject` attribute

The `subject` attribute contains 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 context node may be used.

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

5.5.15 `title` element

The `title` element contains a summary of the purpose or role of the schema, pattern or rule for the purpose of documentation or a rich user interface.

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.

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.

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

6.2 Minimal syntax (informative)

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.

NOTE In the case of an XPath-based query binding this negation can be achieved for simple cases by passing the expression to the `not()` function, whereby e.g. `<sch:report test="*">` would become `<sch:assert test="not(*)">`.

— 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 Abstract pattern processing

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="entry"/>
</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="entry">
    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>
```

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.

A query 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.

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`, `xslt3`, `xpath`, `xpath2`, `xpath3`, `xpath31`, `xquery`, `xquery3`, `xquery31`. Any mix of upper case and lower case letters in these names is permitted.

The default query language binding is `xslt`, which shall be in accordance with Annex C. For design requirements for Schematron schemas using the default query language binding, see Annex E.

The query language bindings for XSLT2, XPath2, XSLT3 and XPath3 shall be in accordance with Annexes H, I, J and K respectively. See Annexes L and M for additional information regarding other query language bindings.

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.

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

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

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.

7 Conformance

7.1 Simple conformance

A simple-conformance implementation shall be able to report for any XML document that it does not conform to the constraints expressed by a given Schematron schema.

— A Schematron schema shall conform to the constraints of Annex A, the ISO/IEC 19757-2 (RELAX NG Compact Syntax) schema.

— A Schematron schema shall conform to the constraints of Annex B, the ISO Schematron schema.

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

NOTE 1 It is not a requirement of this document that a simple-conformance implementation 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 the validation.

NOTE 2 It is not a requirement of this document that a simple-conformance implementation 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 Schematron schema shall conform to the constraints of Annex A, the ISO/IEC 19757-2 (RELAX NG Compact Syntax) schema.

— A Schematron schema shall conform to the constraints of Annex B, the ISO Schematron schema.

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

— A Schematron schema shall have one definition only in scope for any global variable name in the global context and any local variable name in the local context.

— The values of the attributes `flag`, `id`, `name` and `prefix` shall be 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 1 It is not a requirement of this document that a full-conformance implementation 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 the validation.

NOTE 2 It is not a requirement of this document that a full-conformance implementation 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.

```
# Copyright © ISO/IEC 2017
# 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,
# AND SHOULD NOT BE INTERPRETED AS COMPLYING WITH THAT STANDARD".

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

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 },
        attribute role { roleValue }?,
        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 }
         | attribute href { uriValue }),
        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 {
        attribute 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 },
        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

```
<!--
© ISO/IEC 2017
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,
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 query language binding or a `queryBinding` attribute with the value `xslt`, in any mix of upper and lower case letters, shall use the following binding.

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

— The rule context shall be interpreted according to 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 shall be interpreted according to Production 14 of XPath, as returning a Boolean value.

— The name query shall be interpreted according to Production 14 of XPath, as returning a string value. Typically, the `path` 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 shall be interpreted according to Production 14 of XPath, as returning a string value.

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

— The `documents` attribute of the `pattern` element shall be interpreted according to Production 14 of XPath, 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 shall prefix the parameter 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 shall be treated as an XSLT1 variable. The XSLT1 `$` delimiter signifies the use of a variables in a context expression, assertion test, name query, value-of query or let expression. The character `$` 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 Schematron 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 Schematron Validation Report Language namespace URI for exposition purposes. The prefix `svrl` is not reserved or required by this document.

D.2 RELAX NG compact syntax schema

```
# Copyright © ISO/IEC 2019
#
# 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,
# AND SHOULD NOT BE INTERPRETED AS COMPLYING WITH THAT STANDARD."

namespace local = ""
default namespace svrl = "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:NCName }?,
        attribute documents { text }?,
        attribute name { text }?,
        attribute role {string }?,
        empty
    }
# only rules that are fired are reported,
fired-rule =
    element fired-rule {
        attribute id { xsd:NCName }?,
        attribute name { text }?,
        attribute context { text },
        attribute role { string }?,
        attribute flag { string }?,
        attribute document { xsd:anyURI }?,
        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 * - (xml:* | local:*) { text }*
foreign-element =
    element * - svrl:* {
        (attribute * { text }
         | foreign-element
         | text)*
    }
attlist.assert-and-report =
    attribute id { xsd:NCName }?,
    attribute location { text },
    attribute test { text },
    attribute role { string }?,
    attribute flag { string }?
start = schematron-output
```

D.3 Schematron Schema

The corresponding Schematron schema is:

```
<!--
Copyright © ISO/IEC 2014

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,
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(normalize-space(.)) = 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: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 |  svrl:successful-report"/>
        <sch:param name="attribute" value="@location"/>
    </sch:pattern>
    <sch:pattern is-a="requiredAttribute">
        <sch:param name="context" value="svrl:failed-assert | 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 the following:

— represent abstract patterns such as the *head+body* pattern;

— support progressive validation of compound documents; a compound document is 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;

— 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 the following.

— 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.

Certain outcomes are out-of-scope for this document:

— 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 introduced in this document include the following;

— support more query language bindings, in particular XSLT3.

Annex F
(informative)

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 should follow this document:

— `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 XSLT2

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 shall be the extended version of XPath2 specified in XSLT2 with backwards compatibility mode as false. Consequently, the data model used shall be the data model of XDM constructed 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 shall be interpreted according to 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 shall be 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 shall be 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 shall be interpreted according to Production 1 of XPath2, using `fn:string()` on the result of evaluating the expression and returning a string value.

— The `let` value shall be interpreted according to Production 1 of XPath2.

— The `documents` attribute of the element shall be interpreted according to Production 1 of XPath2, as returning a sequence of strings that are evaluated using the `document()` function.

— The notation for signifying the use of a parameter of an abstract pattern shall prefix the parameter 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 shall be treated as an XSLT2 variable. The XSLT2 `$` delimiter signifies the use of a variables in a context expression, assertion test, `name` query, `value-of` query or `let` expression. The character `$` 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 shall be reserved with their XSLT2 usage. All functions defined by these modules shall be available in addition to 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.