INTERNATIONAL STANDARD

ISO/IEC 19756

First edition
2011-06-15

Information technology — Topic Maps — Constraint Language (TMCL)

Technologies de l'information — Plans relatifs à des sujets — Contraintes de langage (TMCL)

Reference number
ISO/IEC 19756:2011(E)

Contents

Page

Foreword v
Introduction vi
1 Scope 1
2 Normative references 1
3 Notation and Conventions 1
3.1 General 1
3.2 tmdm:subject 2
3.3 Following an Association 2
3.4 Following an Association to Roles of Given Type 2
3.5 Playing a Role 2
3.6 Being a Subtype 2
3.7 Being an Instance 2
3.8 Matching a Regular Expression 2
3.9 The Value of an Occurrence 2
3.10 Comparison of iso:ctm-integer values 3
4 Validation Semantics 3
5 TMCL Syntax 3
6 TMCL Declarations 4
6.1 General 4
6.2 Topic Type 4
6.3 Name Type 4
6.4 Occurrence Type 4
6.5 Association Type 4
6.6 Role Type 4
6.7 Overlap Declaration 5
7 TMCL Constraint Types 5
7.1 General 5
7.2 Abstract Topic Type Constraint 5
7.3 Subject Identifier Constraint 5
7.4 Subject Locator Constraint 6
7.5 Item Identifier Constraint 7
7.6 Topic Name Constraint 7
7.7 Variant Name Constraint 8
7.8 Topic Occurrence Constraint 9
7.9 Topic Role Constraint 9
7.10 Scope Constraint 10
7.11 Scope Required Constraint 11
7.12 Reifier Constraint 12
7.13 Topic Reifies Constraint 12
7.14 Association Role Constraint 13
7.15 Role Combination Constraint 14
7.16 Occurrence Data Type Constraint 15
7.17 Unique Value Constraint 15
7.18 Regular Expression Constraint 16
8 Additional TMCL Templates 16
9 User-defined Constraints 16
9.1 General 16
9.2 Denial Constraint 17
9.3 Requirement Constraint 17
10 Schema Documentation 17
10.1 General 17
10.2 The Schema Topic 17

10.3 Documentation Occurrences.......... 18
10.4 The Topic Map Topic Type.......... 19

11 Conformance.......... 19

Annex A (normative) TMCL Templates CTM.......... 20

Annex B (normative) TMCL meta-schema.......... 25

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 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

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.

ISO/IEC 19756 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 34, *Document description and processing languages*.

Introduction

TMCL is a constraint language for Topic Maps, allowing definitions of Topic Maps schemas to be written in a precise and machine-readable form. This makes it possible to validate a topic map against a TMCL schema to see if it conforms to the constraints in the schema, and also enables other uses, such as schema-driven editors, object mappings, and so on.

TMCL is defined as a Topic Maps vocabulary consisting of a number of topic, association, occurrence, and role types, identified by Published Subject Identifiers (PSIs), and defined using English prose. TMCL defines the concept of validation, by which a given topic map is valid according to a schema if it conforms to all the constraints in that schema and a number of global validation rules which apply to all topic maps independent of schema.

TMCL does not have any syntax of its own, since it is defined simply as a Topic Maps vocabulary. However, a number of CTM templates are defined in this International Standard in order to facilitate authoring of TMCL schemas using CTM.

Information technology — Topic Maps — Constraint Language (TMCL)

1 Scope

This International Standard defines a Topic Maps vocabulary for representing constraints on Topic Maps instance data and CTM templates for authoring TMCL schemas.

It does not define a syntax for representing constraints on Topic Maps instance data.

2 Normative references

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

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

ISO/IEC 13250-2:2006, *Information technology — Topic Maps — Part 2: Data model*

ISO/IEC 18048, *Information technology — SGML applications — Topic Map Query Language (TMQL)* [1)]

ISO/IEC 13250-6, *Information technology — Topic Maps — Part 6: Compact syntax*

XML Schema-2, *XML Schema Part 2: Datatypes Second Edition*, W3C Recommendation, 28 October 2004, available at <http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/>

3 Notation and Conventions

3.1 General

The TMCL validation rules are defined in English prose using constructs from ISO/IEC 13250-2:2006 (the TMDM), and written according to certain conventions, whereby some of the phrases in the text have particular interpretations. These phrases and their interpretations are given in the following clauses.

Throughout this clause the defined phrases contain placeholders given as variables (written thus: ***v***) in the definitions. When the phrases are used, these placeholders are defined either by reference to topics already defined in the context where the phrase is used, or using a qname (an identifying token of the form foo:bar). Qnames are expanded into full IRIs using the prefix declarations in Clause 5, and refer to the topic item which has that IRI in its [subject identifiers] property.

The following namespace prefixes are used throughout this International Standard:

```
%prefix tmcl http://psi.topicmaps.org/tmcl/
%prefix tmdm http://psi.topicmaps.org/iso13250/model/
%prefix xsd http://www.w3.org/2001/XMLSchema#
```

Thus, tmdm:bar is a shorthand for the IRI http://psi.topicmaps.org/iso13250/model/bar.

Throughout this International Standard the syntax [foo], that is, a name in square brackets, not italicized, is used to refer to property names from ISO/IEC 13250-2:2006 (the TMDM).

1) To be published

3.2 tmdm:subject

The topic tmdm:subject (http://psi.topicmaps.org/iso13250/model/subject) represents the concept of a subject as defined in ISO/IEC 13250-2:2006. It is the type of which all topics, statements, and association roles are instances, and the common supertype of all types. In TMCL it is used in constraints to make it clear that any topic whatsoever may appear in a particular position in an ontology.

3.3 Following an Association

To follow an association of type ***at*** from a topic ***t*** means traversing all associations of type ***at***. ***A*** is the set of all the association items ***a*** in the [parent] properties of all the role items in the [roles played] property of ***t*** where ***a***'s [type] property contains ***at*** or some subtype of ***at***.

The end result of the traversal is the set of topic items in the [player] property of each role item ***r*** in ***a***'s [roles] property, except where ***r*** is contained in ***t***'s [roles played] property.

3.4 Following an Association to Roles of Given Type

To follow an association of type ***at*** to roles of type ***rt*** from a topic ***t*** means traversing all associations of type ***at***. ***A*** is the set of all the association items ***a*** in the [parent] properties of all the role items in the [roles played] property of ***t*** where ***a***'s [type] property contains ***at*** or a subtype of ***at***.

The end result of the traversal is the set of topic items in the [player] property of each role item ***r*** in ***a***'s [roles] property, except where ***r*** is contained in ***t***'s [roles played] property, or where ***r***'s [type] property does not contain ***rt*** or some subtype of ***rt***.

3.5 Playing a Role

A topic ***t*** is said to play a role of type ***rt*** in an association of type ***at*** when ***t***'s [roles played] property contains at least one role item ***r*** whose [type] property contains ***rt*** (or some subtype of ***rt***) and ***r***'s [parent] property contains an association item whose [type] property contains ***at*** or some subtype of ***at***.

3.6 Being a Subtype

A topic type ***t1*** is said to be an subtype of the topic type ***t2*** if following the association type tmdm:supertype-subtype from ***t1*** to roles of type tmdm:supertype produces either ***t2*** or a subtype of ***t2***. If ***t2*** is tmdm:subject then ***t1*** is a subtype of it even if the necessary tmdm:supertype-subtype associations are not present.

3.7 Being an Instance

A topic ***t*** is said to be an instance of the topic type ***tt*** if following the association type tmdm:type-instance from ***t*** produces either ***tt***, a subtype of ***tt***, or if ***tt*** is tmdm:subject.

A statement ***s*** (as defined in ISO/IEC 13250-2:2006) is said to be an instance of the statement type ***st*** if the ***s***'s [type] property contains either ***st***, a subtype of ***st***, or if ***st*** is tmdm:subject.

3.8 Matching a Regular Expression

A string ***s*** matches a regular expression ***r*** if the string is a member of the set of strings L(r) denoted by ***r*** as defined in appendix F of [XML Schema-2].

3.9 The Value of an Occurrence

The value of a topic ***t***'s occurrence of type ***ot*** is referred to as ***v***, and produced by finding the occurrence item ***o*** in ***t***'s [occurrences] property whose [type] property contains ***ot*** or some subtype of ***ot***. If no such occurrence exists there is no given value. It is an error if there is more than one such occurrence.

3.10 Comparison of iso:ctm-integer values

When comparing tmcl:card-min and tmcl:card-max values, in addition to the ordinary integers, one may encounter the special value * of datatype iso:ctm-integer. This special value is used to indicate unlimited cardinality, and compares as larger than any specific integer.

4 Validation Semantics

This International Standard defines two kinds of rules:

— Constraint validation rules, which specify how to validate the constraint represented by an individual topic that is an instance of some subclass of tmcl:constraint.
— Global validation rules, which are not tied to any specific constraint topic, and which apply to the entire topic map.

A schema is a set of constraint topics, declaration topics, topic types, association types, role types, name types, and occurrence types contained in a topic map.

A topic map is valid according to a schema if the topic map is valid according to each individual constraint in the schema, and if the topic map is valid according to all the global validation rules specified in this International Standard.

NOTE 1 This is a very strict form of validation, which may not be suitable for all situations. Implementations may allow users to turn off any set of constraints they choose in order to allow more flexible forms of validation.

A topic map is valid according to a constraint if the topic map satisifies the constraint validation rule defined for the constraint type of which the constraint topic is an instance.

NOTE 2 TMCL does not dictate when constraints should be applied, nor does it state what it means to a controlling application when it is found that a given constraint is violated.

The validation rules defined in this International Standard assume that the schema and the instance data are located in the same topic map. If this is not the case, implementations shall behave as though the schema and instance topic maps were merged. This includes any schemas included by using tmcl:includes-schema, even though these schemas may physically reside in other topic maps.

Implementations shall also behave as though the topic map in Annex B were merged into the topic map being validated.

5 TMCL Syntax

TMCL has no syntax of its own, but since TMCL schemas are represented as topic maps, any Topic Maps representation can be used to create a TMCL schema. The schema shall, however, be valid according to the TMCL meta-schema in Annex B. The values of all occurrences of type tmcl:regexp shall be valid regular expressions as defined in [XML Schema-2].

Throughout this International Standard all examples are given using the CTM syntax (defined in ISO/IEC 13250-6), and to facilitate the authoring of TMCL in CTM a number of templates are defined. These templates exist in a resource http://www.isotopicmaps.org/tmcl/templates.ctm that can be downloaded and included in any CTM file.

To include the TMCL templates the following CTM include directive can be used:

```
%include http://www.isotopicmaps.org/tmcl/templates.ctm
```

6 TMCL Declarations

6.1 General

TMCL provides a number of constructs known as declarations, which allow Topic Maps constructs to be used in ways which would otherwise be forbidden by the global validation rules. Declarations are in a sense the opposite of constraints, in that their effect is to relax constraints built into TMCL.

6.2 Topic Type

Making a topic an instance of tmcl:topic-type is a declaration that the topic may be used as a topic type.

EXAMPLE Declaring person a topic type:

```
person isa tmcl:topic-type .
```

Global Validation Rule: any topic ***t*** which plays a role of type tmdm:type in an association of type tmdm:type-instance, but which is not an instance of tmcl:topic-type, is invalid.

6.3 Name Type

Making a topic an instance of tmcl:name-type is a declaration that the topic may be used as a name type.

EXAMPLE Declaring nickname a name type:

```
nickname isa tmcl:name-type .
```

Global Validation Rule: any topic name item ***n*** whose [type] property contains a topic item ***t*** where ***t*** is not an instance of tmcl:name-type is invalid.

6.4 Occurrence Type

Making a topic an instance of tmcl:occurrence-type is a declaration that the topic may be used as an occurrence type.

EXAMPLE Declaring date-of-birth an occurrence type:

```
date-of-birth isa tmcl:occurrence-type .
```

Global Validation Rule: any occurrence item ***o*** whose [type] property contains a topic item ***t*** where ***t*** is not an instance of tmcl:occurrence-type is invalid.

6.5 Association Type

Making a topic an instance of tmcl:association-type is a declaration that the topic may be used as an association type.

EXAMPLE Declaring that works-for is an association type:

```
works-for isa tmcl:association-type .
```

Global Validation Rule: any association item ***a*** whose [type] property contains a topic item ***t*** where ***t*** is not an instance of tmcl:association-type is invalid.

6.6 Role Type

Making a topic an instance of tmcl:role-type is a declaration that the topic may be used as a role type.

EXAMPLE Declaring containee a role type:

```
containee isa tmcl:role-type .
```

Global Validation Rule: any association role item ***r*** whose [type] property contains a topic item ***t*** where ***t*** is not an instance of tmcl:role-type is invalid.

6.7 Overlap Declaration

The tmcl:overlap-declaration is used to declare that the sets of instances of two or more topic types are non-disjoint (that is, that they may overlap). The default is that the instance sets of different topic types are disjoint.

EXAMPLE The following declares that the topic types person and employee overlap:

```
person isa tmcl:topic-type;
  overlaps(employee).
```

The CTM template for the overlaps declaration is defined as:

```
def overlaps($tt1, $tt2)
  ?c isa tmcl:overlap-declaration.
  tmcl:overlaps(tmcl:allows : ?c, tmcl:allowed : $tt1)
  tmcl:overlaps(tmcl:allows : ?c, tmcl:allowed : $tt2)
end
```

Global Validation Rule: the set of all the topic types of which a given topic ***t*** is an instance is referred to as ***TT***. ***t*** is invalid if there exists a pair of different topics ***t1*** and ***t2*** in ***TT*** such that:

- ***t1*** is not a subtype of ***t2*** (or vice versa), and
- there does not exist a ***c*** which is an instance of tmcl:overlap-declaration and where following the tmcl:overlaps association from ***c*** produces both ***t1*** and ***t2***.

7 TMCL Constraint Types

7.1 General

The tmcl:constraint topic type is used as the base type for all topic types that are considered constraint types. It is an abstract type used simply to group the constraint types for ease of schema introspection.

7.2 Abstract Topic Type Constraint

The tmcl:abstract-constraint states that a given topic type shall not have any direct instances.

EXAMPLE The following states that creature is an abstract topic type:

```
creature isa tmcl:topic-type;
  is-abstract().
```

The CTM template for the abstract constraint is defined as:

```
def is-abstract($tt)
  ?c isa tmcl:abstract-constraint.
  tmcl:constrained-topic-type(tmcl:constraint : ?c, tmcl:constrained : $tt)
end
```

Constraint Validation Rule for all constraints ***c*** of type tmcl:abstract-constraint: ***c*** applies to a topic type ***t***, which can be found by following associations of type tmcl:constrained-topic-type from the ***c*** topic. ***t*** violates the constraint if it plays the role of type tmdm:type in any association of type tmdm:type-instance.

7.3 Subject Identifier Constraint

A subject identifier constraint constrains the subject identifiers of instances of a given topic type. The constraint has the following properties:

— card-min, indicating the minimum number of subject identifiers a valid instance shall have
— card-max, indicating the maximum number of subject identifiers a valid instance can have, and
— regexp, a regular expression the subject identifier shall match.

EXAMPLE The following states that topics of type person shall have zero or one subject identifier:

```
person isa tmcl:topic-type;
  has-subject-identifier(0, 1, ".*").
```

The CTM template for the subject identifier constraint is defined as:

```
def has-subject-identifier($tt, $min, $max, $regexp)
  ?c isa tmcl:subject-identifier-constraint;
    tmcl:card-min: $min;
    tmcl:card-max: $max;
    tmcl:regexp: $regexp.
  tmcl:constrained-topic-type(tmcl:constraint : ?c, tmcl:constrained : $tt)
end
```

Constraint Validation Rule for all constraints ***c*** of type tmcl:subject-identifier-constraint: ***c*** applies to a topic type ***t***, which can be found by following the tmcl:constrained-topic-type association from c. The constraint has the following properties:

— ***r***, which is the value of ***c***'s occurrence of type tmcl:regexp if given, and ".*" if not,
— ***min***, which is the value of ***c***'s tmcl:card-min occurrence, if given, and 0, if not, and
— ***max***, which is the value of ***c***'s tmcl:card-max occurrence, if given, and undefined, if not.

For each instance ***i*** of ***t*** the set of its subject identifiers which match ***r*** is referred to as ***S***. ***i*** violates ***c*** if the cardinality of ***S*** is lower than ***min*** or greater than ***max*** (provided ***max*** is not undefined).

7.4 Subject Locator Constraint

A subject locator constraint constrains the subject locators of instances of a given topic type. The constraint has the following properties:

— card-min, indicating the minimum number of subject locators a valid instance shall have
— card-max, indicating the maximum number of subject locators a valid instance can have, and
— regexp, a regular expression that the subject locator shall match.

EXAMPLE The following states that topics of type document shall have at least one subject locator:

```
document isa tmcl:topic-type;
  has-subject-locator(1, *, ".*").
```

The CTM template for the subject locator constraint is defined as:

```
def has-subject-locator($tt, $min, $max, $regexp)
  ?c isa tmcl:subject-locator-constraint;
    tmcl:card-min: $min;
    tmcl:card-max: $max;
    tmcl:regexp: $regexp.
  tmcl:constrained-topic-type(tmcl:constraint : ?c, tmcl:constrained : $tt)
end
```

Constraint Validation Rule for all constraints ***c*** of type tmcl:subject-locator-constraint: ***c*** applies to a topic type ***t***, which can be found by following the tmcl:constrained-topic-type association from c. The constraint has the following properties:

— ***r***, which is the value of ***c***'s occurrence of type tmcl:regexp if given, and ".*" if not,
— ***min***, which is the value of ***c***'s tmcl:card-min occurrence, if given, and 0, if not, and

— ***max***, which is the value of ***c***'s tmcl:card-max occurrence, if given, and undefined, if not.

For each instance ***i*** of ***t*** the set of its subject locators which match ***r*** is referred to as ***S***. ***i*** violates ***c*** if the cardinality of ***S*** is lower than ***min*** or greater than ***max*** (provided ***max*** is not undefined).

7.5 Item Identifier Constraint

An item identifier constraint constrains the item identifiers of topic map constructs of a given type. The constraint has the following properties:

— card-min, indicating the minimum number of item identifiers a valid instance shall have
— card-max, indicating the maximum number of item identifiers a valid instance can have, and
— regexp, a regular expression that the item identifier shall match.

EXAMPLE The following states that topics of type document shall have at least one item identifier:

```
document isa tmcl:topic-type;
  has-item-identifier(1, *, ".*").
```

The CTM template for the item identifier constraint is defined as:

```
def has-item-identifier($tt, $min, $max, $regexp)
  ?c isa tmcl:item-identifier-constraint;
    tmcl:card-min: $min;
    tmcl:card-max: $max;
    tmcl:regexp: $regexp.
  tmcl:constrained-construct(tmcl:constraint : ?c, tmcl:constrained : $tt)
end
```

Constraint Validation Rule for all constraints ***c*** of type tmcl:item-identifier-constraint: ***c*** applies to a Topic Maps construct ***t***, which can be found by following the tmcl:constrained-construct association from c. The constraint has the following properties:

— ***r***, which is the value of ***c***'s occurrence of type tmcl:regexp if given, and ".*" if not,
— ***min***, which is the value of ***c***'s tmcl:card-min occurrence, if given, and 0, if not, and
— ***max***, which is the value of ***c***'s tmcl:card-max occurrence, if given, and undefined, if not.

For each instance ***i*** of ***t*** the set of its item identifiers which match ***r*** is referred to as ***S***. ***i*** violates ***c*** if the cardinality of ***S*** is lower than ***min*** or greater than ***max*** (provided ***max*** is not undefined).

7.6 Topic Name Constraint

A topic name constraint constrains the type and cardinality of topic names for instances of a given topic type. The constraint has the following properties:

— card-min, indicating the minimum number of names a valid instance shall have, and
— card-max, indicating the maximum number of names a valid instance shall have.

EXAMPLE The following states that topics of type person shall have exactly one name of type tmdm:topic-name:

```
person isa tmcl:topic-type;
  has-name(tmdm:topic-name, 1, 1).
```

The CTM template for the topic name constraint is defined as:

```
def has-name($tt, $nt, $min, $max)
  ?c isa tmcl:topic-name-constraint;
    tmcl:card-min: $min;
    tmcl:card-max: $max.
  tmcl:constrained-topic-type(tmcl:constraint : ?c, tmcl:constrained : $tt)
  tmcl:constrained-statement(tmcl:constraint : ?c, tmcl:constrained : $nt)
end
```

Constraint Validation Rule for all constraints ***c*** of type tmcl:topic-name-constraint: ***c*** applies to a topic type ***t***, which can be found by following associations of type tmcl:constrained-topic-type from ***c***. The constraint has the following properties:

— ***nt***, which is obtained by following associations of type tmcl:constrained-statement from ***c***,
— ***min***, which is the value of ***c***'s tmcl:card-min occurrence, if given, and 0, if not, and
— ***max***, which is the value of ***c***'s tmcl:card-max occurrence, if given, and undefined, if not.

For each instance ***i*** of ***t*** the set of its topic names of type ***nt*** is referred to as ***N***. ***i*** violates ***c*** if the cardinality of ***N*** is lower than ***min*** or greater than ***max*** (provided ***max*** is not undefined).

Global Validation Rule: any topic name ***n***, whose type is referred to as ***t***, is invalid if no constraint ***c*** that is an instance of tmcl:topic-name-constraint can be found such that the topic found by following ***c***'s tmcl:constrained-statement associations is ***t***.

7.7 Variant Name Constraint

A variant name constraint constrains the scope and cardinality of variant names for topic names on instances of a given topic type. The constraint has the following properties:

— card-min, indicating the minimum number of names a valid instance shall have for each topic name, and
— card-max, indicating the maximum number of names a valid instance shall have for each topic name.

EXAMPLE The following states that for topics of type person the tmdm:topic-names shall have exactly one variant with xtm:sort in its scope:

```
person isa tmcl:topic-type;
  has-variant(tmdm:topic-name, xtm:sort, 1, 1).
```

The CTM template for the variant name constraint is defined as:

```
def has-variant($tt, $nt, $t, $min, $max)
 ?c isa tmcl:variant-name-constraint;
   tmcl:card-min: $min;
   tmcl:card-max: $max.

 tmcl:constrained-topic-type(tmcl:constraint : ?c, tmcl:constrained : $tt)
 tmcl:constrained-statement(tmcl:constraint : ?c, tmcl:constrained : $nt)
 tmcl:constrained-scope-topic(tmcl:constraint : ?c, tmcl:constrained : $t)
end
```

Constraint Validation Rule for all constraints ***c*** of type tmcl:variant-name-constraint: ***c*** applies to a topic type ***t***, which can be found by following associations of type tmcl:constrained-topic-type from ***c***. The constraint has the following properties:

— ***nt***, which is obtained by following associations of type tmcl:constrained-statement from ***c***,
— ***s***, which is obtained by following associations of type tmcl:constrained-scope-topic from ***c***,
— ***min***, which is the value of ***c***'s tmcl:card-min occurrence, if given, and 0, if not, and
— ***max***, which is the value of ***c***'s tmcl:card-max occurrence, if given, and undefined, if not.

For each instance ***i*** of ***nt*** which is attached to a topic of type ***tt*** the set of its variant names whose scopes contain ***s*** is referred to as ***N***. ***i*** violates ***c*** if the cardinality of ***N*** is lower than ***min*** or greater than ***max*** (provided ***max*** is not undefined).

Global Validation Rule: any variant name ***v***, whose scope is referred to as ***S***, is invalid if no constraint ***c*** that is an instance of tmcl:variant-name-constraint can be found such that

— the topic found by following ***c***'s tmcl:constrained-statement associations is the same as the type of ***v***'s topic name,
— the topic found by following ***c***'s tmcl:constrained-topic-type associations is the same as the type of ***v***'s topic name's topic, and
— the topic found by following ***c***'s tmcl:constrained-scope-topic associations is an element of ***S***.

7.8 Topic Occurrence Constraint

A topic occurrence constraint constrains the type and cardinality of occurrences connected to a topic of a given type. The constraint has the following properties:

— card-min, indicating the minimum number of occurrences a valid instance shall have, and
— card-max, indicating the maximum number of occurrences a valid instance may have.

EXAMPLE The following states that topics of type person shall have exactly one occurrence of type date-of-birth:

```
person isa tmcl:topic-type;
  has-occurrence(date-of-birth, 1, 1).
```

The CTM template for the topic occurrence constraint is defined as:

```
def has-occurrence($tt, $ot, $min, $max)
  ?c isa tmcl:topic-occurrence-constraint;
    tmcl:card-min: $min;
    tmcl:card-max: $max.

  tmcl:constrained-topic-type(tmcl:constraint : ?c, tmcl:constrained : $tt)
  tmcl:constrained-statement(tmcl:constraint : ?c, tmcl:constrained : $ot)
end
```

Constraint Validation Rule for all constraints ***c*** of type tmcl:topic-occurrence-constraint: ***c*** applies to a topic type ***t***, which can be found by following the association of type tmcl:constrained-topic-type from ***c***. The constraint has the following properties:

— ***ot***, which is obtained by following associations of type tmcl:constrained-statement from ***c***,
— ***min***, which is the value of ***c***'s tmcl:card-min occurrence, if given, and 0, if not, and
— ***max***, which is the value of ***c***'s tmcl:card-max occurrence, if given, and undefined, if not.

For each instance ***i*** of ***t*** the set of its occurrences of type ***ot*** is referred to as ***O***. ***i*** violates ***c*** if the cardinality of ***O*** is lower than ***min*** or greater than ***max*** (provided ***max*** is not undefined).

Global Validation Rule: any occurrence ***o***, whose type is referred to as ***t***, is invalid if no constraint ***c*** that is an instance of tmcl:topic-occurrence-constraint can be found such that the topic found by following ***c***'s tmcl:constrained-statement associations is ***t***.

7.9 Topic Role Constraint

A topic role constraint constrains the types of roles topics of a given type can play in associations of a given type. It can also be seen as constraining the types of topics which may play roles of a given type in associations of a given type. The constraint has the following properties:

— card-min, indicating the minimum number of roles a valid instance shall have, and
— card-max, indicating the maximum number of roles a valid instance can have.

EXAMPLE The following states that topics of type person shall have exactly one role of type employee in associations of type works-for:

```
person isa tmcl:topic-type;
  plays-role(employee, works-for, 1, 1).
```

The CTM template for the topic role constraint is defined as:

```
def plays-role($tt, $rt, $at, $min, $max)
  ?c isa tmcl:topic-role-constraint;
    tmcl:card-min: $min;
    tmcl:card-max: $max.

  tmcl:constrained-topic-type(tmcl:constraint : ?c, tmcl:constrained : $tt)
  tmcl:constrained-statement(tmcl:constraint : ?c, tmcl:constrained : $at)
  tmcl:constrained-role(tmcl:constraint : ?c, tmcl:constrained : $rt)
end
```

Constraint Validation Rule for all constraints ***c*** of type tmcl:topic-role-constraint: ***c*** applies to a topic type ***tt***, which can be found by following the association of type tmcl:constrained-topic-type from ***c***. The constraint has the following properties:

— ***rt***, which is obtained by following associations of type tmcl:constrained-role from ***c***,
— ***at***, which is obtained by following associations of type tmcl:constrained-statement from ***c***,
— ***min***, which is the value of ***c***'s tmcl:card-min occurrence, if given, and 0, if not, and
— ***max***, which is the value of ***c***'s tmcl:card-max occurrence, if given, and undefined, if not.

For each instance ***i*** of ***tt*** the set of its roles of type ***rt*** is referred to as ***R***. ***i*** violates ***c*** if the cardinality of ***R*** is lower than ***min*** or greater than ***max*** (provided ***max*** is not undefined).

Global Validation Rule: any association role ***r***, whose type is referred to as ***rt***, and the type of whose containing association is referred to as ***at***, is invalid if no constraint ***c*** that is an instance of tmcl:topic-role-constraint can be found such that the topic found by following ***c***'s tmcl:constrained-statement associations is ***at*** and such that the topic found by following ***c***'s tmcl:constrained-role associations is ***rt***.

7.10 Scope Constraint

A tmcl:scope-constraint constrains the types of topics which may appear in the scope of a name, occurrence, or association of a particular type. The constraint has the following properties:

— card-min, indicating the minimum number of scoping topics a valid instance shall have, and
— card-max, indicating the maximum number of scoping topics a valid instance can have.

EXAMPLE The following states that every occurrence of type description shall have exactly one topic of type language in its scope:

```
description isa tmcl:occurrence-type;
  has-scope(language, 1, 1).
```

The CTM template for the scope constraint is defined as:

```
def has-scope($st, $tt, $min, $max)
  ?c isa tmcl:scope-constraint;
    tmcl:card-min: $min;
    tmcl:card-max: $max.

  tmcl:constrained-statement(tmcl:constraint : ?c, tmcl:constrained : $st)
  tmcl:constrained-scope(tmcl:constraint : ?c, tmcl:constrained : $tt)
end
```

Constraint Validation Rule for all constraints ***c*** of type tmcl:scope-constraint: ***c*** applies to a name, occurrence, or association type ***st***, which can be found by following associations of type tmcl:constrained-statement from ***c***. It has the following properties:

— ***t***, which can be found by following associations of type tmcl:constrained-scope from ***c***,
— ***min***, which is the value of ***c***'s tmcl:card-min occurrence, if given, and 0, if not, and
— ***max***, which is the value of ***c***'s tmcl:card-max occurrence, if given, and undefined, if not.

For any name, occurrence, or association ***s*** of type ***st*** the set of topics in its scope which are instances of ***t*** is referred to as ***S***. ***s*** violates ***c*** if the cardinality of ***S*** is lower than ***min*** or greater than ***max*** (provided ***max*** is not undefined).

Global Validation Rule: For every topic ***t*** in the scope of a name, occurrence, or association ***s***, where ***s***'s type is referred to as ***st***, ***t*** is invalid if the following does not hold. There shall exist a constraint ***c***, which shall be an instance of tmcl:scope-constraint, where following associations of type tmcl:constrained-statement from ***c*** shall produce ***st***, and following associations of type tmcl:constrained-scope from ***c*** shall produce at least one topic type of which ***t*** is an instance.

7.11 Scope Required Constraint

A tmcl:scope-required-constraint constrains the appearance of a name, occurrence, or association of a particular type on topics of a particular type with a given instance topic in its scope. The constraint has the following properties:

— card-min, indicating the minimum number of times the given statement shall appear on topics of the given type, and
— card-max, indicating the maximum number of times the given statement shall appear on topics of the given type.

EXAMPLE The following states that every topic of type category shall have a description in the scope english, and another in the scope norwegian:

```
description isa tmcl:occurrence-type;
  has-scope(language, 1, 1).

category isa tmcl:topic-type;
  has-occurrence(description, 2, 2);
  requires-scope(description, english, 1, 1);
  requires-scope(description, norwegian, 1, 1).
```

The CTM template for the scope required constraint is defined as:

```
def requires-scope($tt, $st, $t, $min, $max)
  ?c isa tmcl:scope-required-constraint;
  tmcl:card-min: $min;
  tmcl:card-max: $max.

  tmcl:constrained-topic-type(tmcl:constraint : ?c, tmcl:constrained : $tt)
  tmcl:constrained-statement(tmcl:constraint : ?c, tmcl:constrained : $st)
  tmcl:constrained-scope-topic(tmcl:constraint : ?c, tmcl:constrained : $t)
end
```

Constraint Validation Rule for all constraints ***c*** of type tmcl:scope-required-constraint: ***c*** applies to a name, occurrence, or association type ***st***, which can be found by following associations of type tmcl:constrained-statement from ***c***. It has the following properties:

— ***tt***, which can be found by following associations of type tmcl:constrained-topic-type from ***c***,
— ***t***, which can be found by following associations of type tmcl:constrained-scope-topic from ***c***,
— ***min***, which is the value of ***c***'s tmcl:card-min occurrence, if given, and 0, if not, and
— ***max***, which is the value of ***c***'s tmcl:card-max occurrence, if given, and undefined, if not.

For any topic ***i*** of type ***tt*** the set of its names, occurrences, and associations of type ***st*** whose scopes contain ***t*** is known as ***S***. ***i*** violates ***c*** if the cardinality of ***S*** is greater than ***max*** (provided ***max*** is not undefined) or less than ***min***.

7.12 Reifier Constraint

A tmcl:reifier-constraint constrains whether or not names, occurrences, and associations of a given type may be reified, and if so, what the type of the reifying topic shall be. The constraint has the following properties:

— card-min, indicating the minimum number of times the statement shall be reified (which shall be 0 or 1), and
— card-max, indicating the maximum number of times the statement can be reified (which shall be 0 or 1).

EXAMPLE The following states that occurrences of type date-of-birth may not be reified:

```
date-of-birth isa tmcl:occurrence-type;
  cannot-have-reifier().
```

The CTM templates for the reifier constraint are defined as:

```
def must-have-reifier($st, $tt)
  ?c isa tmcl:reifier-constraint;
    tmcl:card-min: 1;
    tmcl:card-max: 1.
  tmcl:constrained-statement(tmcl:constraint: ?c, tmcl:constrained: $st)
  tmcl:allowed-reifier(tmcl:allows: ?c, tmcl:allowed: $tt)
end

def cannot-have-reifier($st)
  ?c isa tmcl:reifier-constraint;
    tmcl:card-min: 0;
    tmcl:card-max: 0.
  tmcl:constrained-statement(tmcl:constraint: ?c, tmcl:constrained: $st)
  tmcl:allowed-reifier(tmcl:allows: ?c, tmcl:allowed: tmdm:subject)
end

def may-have-reifier($st, $tt)
  ?c isa tmcl:reifier-constraint;
    tmcl:card-min: 0;
    tmcl:card-max: 1.
  tmcl:constrained-statement(tmcl:constraint: ?c, tmcl:constrained: $st)
  tmcl:allowed-reifier(tmcl:allows: ?c, tmcl:allowed: $tt)
end
```

Constraint Validation Rule for all constraints ***c*** of type tmcl:reifier-constraint: ***c*** applies to a statement type ***st***, which can be found by following associations of type tmcl:constrained-statement, and has the following properties:

— ***tt***, which can be found by following associations of type tmcl:allowed-reifier,
— ***min***, which is the value of ***c***'s tmcl:card-min occurrence, if given, and 0, if not, and
— ***max***, which is the value of ***c***'s tmcl:card-max occurrence, if given, and undefined, if not.

Any topic ***t*** which is an instance of type ***tt*** and reifies a name, occurrence, or association ***s*** of type ***st*** violates ***c*** if ***max*** is 0.

Any name, occurrence, or association ***s*** of type ***st*** violates ***c*** if ***min*** is 1 and either ***s*** has no reifier, or ***s***'s reifier is not an instance of ***tt***.

7.13 Topic Reifies Constraint

A tmcl:topic-reifies-constraint constrains what types of statements topics of a given type may reify. The constraint has the following properties:

— card-min, indicating the minimum number of times the topic shall reify a statement (which shall be 0 or 1), and

— card-max, indicating the maximum number of times the topic may reify a statement (which shall be 0 or 1).

EXAMPLE The following states that topics of type person are not allowed to reify anything:

```
person isa tmcl:topic-type;
  cannot-reify().
```

The CTM templates for the topic reifies constraint are defined as:

```
def must-reify($tt, $st)
  ?c isa tmcl:topic-reifies-constraint;
    tmcl:card-min: 1;
    tmcl:card-max: 1.
  tmcl:constrained-topic-type(tmcl:constraint: ?c, tmcl:constrained: $tt)
  tmcl:constrained-statement(tmcl:constraint: ?c, tmcl:constrained: $st)
end

def cannot-reify($tt)
  ?c isa tmcl:topic-reifies-constraint;
    tmcl:card-min: 0;
    tmcl:card-max: 0.
  tmcl:constrained-topic-type(tmcl:constraint: ?c, tmcl:constrained: $tt)
end

def may-reify($tt, $st)
  ?c isa tmcl:topic-reifies-constraint;
    tmcl:card-min: 0;
    tmcl:card-max: 1.
  tmcl:constrained-topic-type(tmcl:constraint: ?c, tmcl:constrained: $tt)
  tmcl:constrained-statement(tmcl:constraint: ?c, tmcl:constrained: $st)
end
```

Constraint Validation Rule for all constraints ***c*** of type tmcl:topic-reifies-constraint: ***c*** applies to a topic type ***t***, which can be found by following the association of type tmcl:constrained-topic-type from ***c***. The constraint has the following properties:

— ***st***, which can be found by following associations of type tmcl:constrained-statement (if nothing is found, ***st*** is undefined),
— ***min***, which is the value of ***c***'s tmcl:card-min occurrence, if given, and 0, if not, and
— ***max***, which is the value of ***c***'s tmcl:card-max occurrence, if given, and 1, if not.

For each instance ***i*** of ***t*** the construct reified by ***i*** is referred to as ***r***. If there is no such construct, ***r*** is undefined.

i violates ***c*** if ***st*** is defined and ***r*** is not an instance of ***st***, or if ***min*** is 1 and ***r*** is not defined, or if ***max*** is 0 and ***r*** is defined.

7.14 Association Role Constraint

A tmcl:association-role-constraint constrains the number of roles of a particular type that may appear in associations of a given type. The constraint has the following properties:

— at, the association type being constrained,
— rt, the role type being constrained,
— card-min, indicating the minimum number of times the role can appear in the association, and
— card-max, indicating the maximum number of times the role can appear in the association.

EXAMPLE The following states that each association of type works-for shall have exactly one role of type employee and one of type employer:

```
works-for isa tmcl:association-type;
```

```
  has-role(employee, 1, 1);
  has-role(employer, 1, 1).
```

The CTM template for the association role constraint is defined as:

```
def has-role($at, $rt, $min, $max)
  ?c isa tmcl:association-role-constraint;
    tmcl:card-min: $min;
    tmcl:card-max: $max.

  tmcl:constrained-statement(tmcl:constraint : ?c, tmcl:constrained : $at)
  tmcl:constrained-role(tmcl:constraint : ?c, tmcl:constrained : $rt)
end
```

Constraint Validation Rule for all constraints ***c*** of type tmcl:association-role-constraint: ***c*** applies to an association type ***at***, which can be found by following the tmcl:constrained-statement association from ***c***, and has the following properties:

— ***rt***, which can be found by following associations of type tmcl:constrained-role,
— ***min***, which is the value of ***c***'s tmcl:card-min occurrence, if given, and 0, if not, and
— ***max***, which is the value of ***c***'s tmcl:card-max occurrence, if given, and undefined, if not.

For each association ***a*** of type ***at*** the set of its roles which are of type ***rt*** is referred to as ***R***. ***a*** violates ***c*** if the cardinality of ***R*** is lower than ***min*** or greater than ***max*** (unless ***max*** is undefined).

Global Validation Rule: for all association roles ***r***, ***r***'s type is referred to as ***rt***, the association it appears in is referred to as ***a***, and ***a***'s type is referred to as ***at***. ***r*** is invalid if there is no constraint ***c*** that is an instance of tmcl:association-role-constraint such that:

— following ***c***'s tmcl:constrained-statement association yields ***at***, and
— following ***c***'s tmcl:constrained-role association yields ***rt***.

7.15 Role Combination Constraint

The tmcl:role-combination-constraint constrains which combinations of topic types may appear together in associations of a certain type. The most common use case is for a hierarchical association like contained-in where a continent may contain countries, countries may contain provinces, and provinces may contain cities, but no other combinations (like a city contained directly in a continent) are allowed.

EXAMPLE The following states that the only allowed role player type combinations are (city, province), (province, country), (country, continent):

```
contained-in isa tmcl:association-type;
  has-role(containee, 1, 1);
  has-role(container, 1, 1);
  role-combination(containee, city, container, province);
  role-combination(containee, province, container, country);
  role-combination(containee, country, container, continent).
```

The CTM template for role combination constraint is defined as:

```
def role-combination($at, $rt, $tt, $ort, $ott)
  ?c isa tmcl:role-combination-constraint.
  tmcl:constrained-statement(tmcl:constraint: ?c, tmcl:constrained: $at)
  tmcl:constrained-role(tmcl:constraint: ?c, tmcl:constrained: $rt)
  tmcl:constrained-topic-type(tmcl:constraint: ?c, tmcl:constrained: $tt)
  tmcl:other-constrained-role(tmcl:constraint: ?c, tmcl:constrained: $ort)
  tmcl:other-constrained-topic-type(tmcl:constraint: ?c, tmcl:constrained: $ott)
end
```

Global Validation Rule: an association ***a*** of type ***at***, with a role ***r1*** of type ***rt1*** with player ***rp1*** which is a direct instance of ***tt1***, and another role ***r2*** of type ***rt2*** with player ***rp2*** which is a direct instance of ***tt2***, is invalid if there exists at least one ***c*** which is an instance of tmcl:role-combination-constraint and where following ***c***'s associations of type tmcl:constrained-statement produces ***at***, there does not exist any ***c2*** which is an instance of tmcl:role-combination-constraint and has the following properties:

— following ***c2***'s associations of type tmcl:constrained-statement produces ***at***,
— following ***c2***'s associations of type tmcl:constrained-role produces ***rt1***,
— following ***c2***'s associations of type tmcl:constrained-topic-type produces ***tt1***,
— following ***c2***'s associations of type tmcl:other-constrained-role produces ***rt2***, and
— following ***c2***'s associations of type tmcl:other-constrained-topic-type produces ***tt2***.

7.16 Occurrence Data Type Constraint

A tmcl:occurrence-datatype-constraint constrains the allowed datatype of an occurrence of a given type.

EXAMPLE The following states that all occurrences of type date-of-birth shall have the datatype xsd:dateTime:

```
date-of-birth isa tmcl:occurrence-type;
  has-datatype(xsd:dateTime).
```

The CTM template for the occurrence datatype constraint is defined as:

```
def has-datatype($ot, $dt)
  ?c isa tmcl:occurrence-datatype-constraint;
    tmcl:datatype: $dt.
  tmcl:constrained-statement(tmcl:constraint : ?c, tmcl:constrained : $ot)
end
```

Constraint Validation Rule for all constraints ***c*** of type tmcl:occurrence-datatype-constraint: ***c*** applies to an occurrence type ***t***, which can be found by following the tmcl:constrained-statement association from ***c***. It has a datatype ***d***, which is the value of ***c***'s tmcl:datatype occurrence.

Any occurrence ***o*** of type ***t*** has a datatype known as ***d2***, a value known as ***v***, and violates constraint ***c*** if:

— ***d2*** is not ***d*** or XML Schema validly substitutable for it,
— ***v*** is not XML Schema datatype valid according to ***d***, and
— ***v*** is not XML Schema datatype valid according to ***d2***.

7.17 Unique Value Constraint

The tmcl:unique-value-constraint constraint constrains all names or occurrences of a given type to have different values.

EXAMPLE The following states that all occurrences of type email shall have different values:

```
email isa tmcl:occurrence-type;
  has-unique-value().
```

The CTM template for the unique value constraint is defined as:

```
def has-unique-value($st)
  ?c isa tmcl:unique-value-constraint.
  tmcl:constrained-statement(tmcl:constraint : ?c, tmcl:constrained : $st)
end
```

Constraint Validation Rule for all constraints ***c*** of type tmcl:unique-value-constraint: ***c*** applies to a name or occurrence type ***st***, which can be found by following associations of type tmcl:constrained-statement from ***c***. Any two different names or occurrences ***s1*** and ***s2*** of type ***st*** violate ***c*** if their values are equal.

7.18 Regular Expression Constraint

The tmcl:regular-expression-constraint constrains all values of a given name or occurrence type shall match a given regular expression.

EXAMPLE The following states that phone-number occurrences shall follow a particular syntax:

```
phone-number isa tmcl:occurrence-type;
  matches-regexp("(+47\s)?\d\d\s\d\d\s\d\d\s\d\d").
```

The CTM template for the regular expression constraint is defined as:

```
def matches-regexp($st, $regexp)
  ?c isa tmcl:regular-expression-constraint;
    tmcl:regexp: $regexp.

  tmcl:constrained-statement(tmcl:constraint: ?c, tmcl:constrained: $st)
end
```

Constraint Validation Rule for all constraints ***c*** of type tmcl:regular-expression-constraint: ***c*** applies to a name or occurrence type ***t***, which can be found by following associations of type tmcl:constrained-statement from ***c***. It has a regular expression ***r***, which is the value of ***c***'s tmcl:regexp occurrence.

Any name or occurrence of type ***t*** whose value does not match the regular expression ***r*** violates ***c***.

8 Additional TMCL Templates

This clause defines additional CTM templates, which are strictly speaking redundant, but which aid authoring by simplifying a number of common use cases.

The following template is used to define a binary association type and two role types:

```
def binary-association($at, $rt1, $rt2)
  $at isa tmcl:association-type;
    has-role($rt1, 1, 1);
    has-role($rt2, 1, 1).

  $rt1 isa tmcl:role-type.
  $rt2 isa tmcl:role-type.
end
```

The following template is used to define a binary symmetric association type and one role type:

```
def symmetric-association($at, $rt)
  $at isa tmcl:association-type;
    has-role($rt, 2, 2).

  $rt isa tmcl:role-type.
end
```

9 User-defined Constraints

9.1 General

In addition to the predefined constraints provided by TMCL it is possible for users to define their own constraints in order to define more detailed limitations which the predefined constraints cannot express. These constraints shall all be instances of some subtype of tmcl:user-defined-constraint. Each constraint type shall provide a TMQL expression which defines the interpretation of the constraint.

tmcl:validation-expression is an occurrence type used on user-defined constraints, which contains a TMQL expression as defined in ISO/IEC 18048. Its exact use and meaning depends on the kind of user-defined constraint it is used with, as defined below.

9.2 Denial Constraint

A topic of type tmcl:denial-constraint represents a user-defined constraint whose tmcl:validation-expression shall return nothing in order to be satisfied. That is, if the TMQL query in the tmcl:validation-expression returns a non-empty result the constraint is violated.

EXAMPLE The following constraint states that a person's date-of-death shall be equal to or later than the person's date-of-birth:

```
c isa tmcl:denial-constraint;
  tmcl:validation-expression "
    // person [ . / date-of-birth <= . / date-of-death ]
  ".
```

Constraint Validation Rule for all constraints ***c*** of type tmcl:denial-constraint: ***c*** has an attached query ***q***, which is the value of its occurrence of type tmcl:validation-expression. The query ***q*** is run against the topic map being validated, and if the query result is non-empty ***c*** is violated.

9.3 Requirement Constraint

A topic of type tmcl:requirement-constraint represents a user-defined constraint whose tmcl:validation-expression shall return at least one result row in order to be satisfied. That is, if the TMQL query in the tmcl:validation-expression returns an empty result the constraint is violated. As such it is the mirror image of tmcl:denial-constraint.

EXAMPLE The following constraint states that the topic map shall contain at least one person topic:

```
c isa tmcl:requirement-constraint;
  tmcl:validation-expression "
    // person
  ".
```

Constraint Validation Rule for all constraints ***c*** of type tmcl:requirement-constraint: ***c*** has an attached query ***q***, which is the value of its occurrence of type tmcl:validation-expression. The query ***q*** is run against the topic map being validated, and if the query result is empty ***c*** is violated.

10 Schema Documentation

10.1 General

TMCL provides a number of constructs which are not constraints or declarations, but which provide useful documentation to users. These constructs are defined in this clause.

10.2 The Schema Topic

The tmcl:schema topic type is provided in order to provide a standard mechanism for making assertions about schemas in topic maps. Each topic of this type represents a TMCL schema.

The tmcl:belongs-to-schema association type is used to connect individual constraints and declarations with the schema. This is useful in situations where a single topic map contains more than one schema, in order to track what comes from which schema.

EXAMPLE 1 The following has an example of a schema containing a topic type and a constraint on that topic type:

```
example-schema isa tmcl:schema;
  - "Example schema".
```

```
creature isa tmcl:topic-type;
  - "Creature";
  belongs-to(example-schema).

c isa tmcl:abstract-constraint;
  belongs-to(example-schema).

tmcl:constrained-topic-type(tmcl:constraint : c, tmcl:constrained : creature)
```

The CTM template for the belongs to schema association is defined as:

```
def belongs-to($construct, $schema)
  tmcl:belongs-to-schema(tmcl:container: $schema, tmcl:containee: $construct)
end
```

The tmcl:version occurrence type is used to attach some identifier of the schema's version to the schema topic.

EXAMPLE 2 The following is an example of using tmcl:version:

```
example-schema isa tmcl:schema;
  - "Example schema";
  tmcl:version: "1.0".
```

The tmcl:schema-resource occurrence type is used to refer from the schema topic to a resource that contains some serialization of the schema.

EXAMPLE 3 The following is an example of using tmcl:schema-resource:

```
example-schema isa tmcl:schema;
  - "Example schema";
  tmcl:schema-resource: http://example.org/example.ctm .
```

The tmcl:includes-schema association type is used to reference another schema used in the current schema, in order to import all its declarations and constraints. The importing of declarations and constraints is done by the TMCL processor.

EXAMPLE 4 The following is an example of importing another schema:

```
example isa tmcl:schema;
  - "Example schema";
  tmcl:schema-resource: http://example.org/example.ctm .

sub isa tmcl:schema;
  - "Subschema";
  tmcl:schema-resource: http://example.org/sub.ctm .

tmcl:includes-schema(tmcl:container: example, tmcl:containee: sub)
```

10.3 Documentation Occurrences

Three occurrence types which may be attached to any TMCL construct to document it are also provided, as described in this clause.

The tmcl:description occurrence type is used to attach a textual description of a TMCL construct to it inside the topic map.

The tmcl:comment occurrence type is used to attach any textual information to a TMCL construct inside the topic map.

The tmcl:see-also occurrence type is used to attach a to a TMCL construct a reference to any kind of external information about that construct.

EXAMPLE The following shows example use of all three occurrence types:

```
creature isa tmcl:topic-type;
  tmcl:description: "A creature is any kind of living organism.";
  tmcl:comment: "Not sure if we need this. Should review.";
  tmcl:see-also: http://en.wikipedia.org/wiki/Creature .
```

10.4 The Topic Map Topic Type

The tmcl:topic-map topic type is is provided as a mechanism for making assertions about topic maps. Each instance of this topic type represents an individual topic map.

The tmcl:uses-schema association type connects a tmcl:topic-map, playing the role of tmcl:user, with a tmcl:schema topic, playing the role of tmcl:used, representing a schema used by that topic map. This association is for documentation purposes only, and does not affect the validation of the topic map in any way.

EXAMPLE The following states that the topic map uses the TMCL meta-schema:

```
~ topicmap

tmcl:uses-schema(tmcl:user: topicmap, tmcl:used: meta-schema)

meta-schema isa tmcl:schema .
```

11 Conformance

There are two levels of conformance defined by this International Standard. In order to meet the requirements for Level One conformance, a TMCL processor shall be able to:

— validate TMCL schemas and reject all schemas which are not valid according to the requirements in Clause 5, and
— validate topic maps against valid TMCL schemas as defined in Clause 4, but without necessarily evaluating user-defined constraints, and report accurately whether or not each topic map is valid according to the schema.

NOTE 1 This implies that level one processors are not required to support TMQL.

Further, Level One processors shall recognize the validly substitutable relationships between and be able to validate lexical values against the following datatypes:

— xsd:anyURI,
— xsd:decimal,
— xsd:integer,
— xsd:date,
— xsd:dateTime,
— xsd:string, and
— iso:ctm-integer.

In order to satisfy Level Two conformance processors shall conform to Level One and in addition also evaluate user-defined constraints.

NOTE 2 This implies that level two processors are required to support TMQL.

Annex A
(normative)

TMCL Templates CTM

The CTM defined below is the same that can be located online at http://www.isotopicmaps.org/tmcl/templates.ctm

```
##########################################################################
# TMCL - Topic Maps Constraint Language
# This file defines the set of templates defined in the TMCL standard
# This document is located at http://www.isotopicmaps.org/tmcl/templates.ctm
# and is normative.
# Final version of 2011-01-04.
##########################################################################

# prefix declarations
%prefix tmcl http://psi.topicmaps.org/tmcl/
%prefix tmdm http://psi.topicmaps.org/iso13250/model/


# Overlap Declaration
def overlaps($tt1, $tt2)
  ?c isa tmcl:overlap-declaration.
  tmcl:overlaps(tmcl:allows : ?c, tmcl:allowed : $tt1)
  tmcl:overlaps(tmcl:allows : ?c, tmcl:allowed : $tt2)
end

# Abstract Topic Type Constraint
def is-abstract($tt)
  ?c isa tmcl:abstract-constraint.
  tmcl:constrained-topic-type(tmcl:constraint : ?c, tmcl:constrained : $tt)
end

# Subject Identifier Constraint
def has-subject-identifier($tt, $min, $max, $regexp)
  ?c isa tmcl:subject-identifier-constraint;
    tmcl:card-min: $min;
    tmcl:card-max: $max;
    tmcl:regexp: $regexp.
  tmcl:constrained-topic-type(tmcl:constraint : ?c, tmcl:constrained : $tt)
end

# Subject Locator Constraint
def has-subject-locator($tt, $min, $max, $regexp)
  ?c isa tmcl:subject-locator-constraint;
    tmcl:card-min: $min;
    tmcl:card-max: $max;
    tmcl:regexp: $regexp.
  tmcl:constrained-topic-type(tmcl:constraint : ?c, tmcl:constrained : $tt)
end

# Item Identifier Constraint
def has-item-identifier($tt, $min, $max, $regexp)
  ?c isa tmcl:item-identifier-constraint;
    tmcl:card-min: $min;
    tmcl:card-max: $max;
    tmcl:regexp: $regexp.
  tmcl:constrained-construct(tmcl:constraint : ?c, tmcl:constrained : $tt)
end

# Topic Name Constraint
def has-name($tt, $nt, $min, $max)
  ?c isa tmcl:topic-name-constraint;
    tmcl:card-min: $min;
    tmcl:card-max: $max.

  tmcl:constrained-topic-type(tmcl:constraint : ?c, tmcl:constrained : $tt)
  tmcl:constrained-statement(tmcl:constraint : ?c, tmcl:constrained : $nt)
end

# Variant Name Constraint
def has-variant($tt, $nt, $t, $min, $max)
 ?c isa tmcl:variant-name-constraint;
   tmcl:card-min: $min;
   tmcl:card-max: $max.

 tmcl:constrained-topic-type(tmcl:constraint : ?c, tmcl:constrained : $tt)
 tmcl:constrained-statement(tmcl:constraint : ?c, tmcl:constrained : $nt)
 tmcl:constrained-scope-topic(tmcl:constraint : ?c, tmcl:constrained : $t)
end

# Topic Occurrence Constraint
def has-occurrence($tt, $ot, $min, $max)
  ?c isa tmcl:topic-occurrence-constraint;
    tmcl:card-min: $min;
    tmcl:card-max: $max.

  tmcl:constrained-topic-type(tmcl:constraint : ?c, tmcl:constrained : $tt)
  tmcl:constrained-statement(tmcl:constraint : ?c, tmcl:constrained : $ot)
end

# Topic Role Constraint
def plays-role($tt, $rt, $at, $min, $max)
  ?c isa tmcl:topic-role-constraint;
    tmcl:card-min: $min;
    tmcl:card-max: $max.

  tmcl:constrained-topic-type(tmcl:constraint : ?c, tmcl:constrained : $tt)
  tmcl:constrained-statement(tmcl:constraint : ?c, tmcl:constrained : $at)
  tmcl:constrained-role(tmcl:constraint : ?c, tmcl:constrained : $rt)
end

# Scope Constraint
def has-scope($st, $tt, $min, $max)
  ?c isa tmcl:scope-constraint;
    tmcl:card-min: $min;
    tmcl:card-max: $max.

  tmcl:constrained-statement(tmcl:constraint : ?c, tmcl:constrained : $st)
  tmcl:constrained-scope(tmcl:constraint : ?c, tmcl:constrained : $tt)
end

# Scope Required Constraint
def requires-scope($tt, $st, $t, $min, $max)
  ?c isa tmcl:scope-required-constraint;
  tmcl:card-min: $min;
  tmcl:card-max: $max.

  tmcl:constrained-topic-type(tmcl:constraint : ?c, tmcl:constrained : $tt)
  tmcl:constrained-statement(tmcl:constraint : ?c, tmcl:constrained : $st)
  tmcl:constrained-scope-topic(tmcl:constraint : ?c, tmcl:constrained : $t)
end

# Reifier Constraint
def must-have-reifier($st, $tt)
  ?c isa tmcl:reifier-constraint;
    tmcl:card-min: 1;
    tmcl:card-max: 1.
  tmcl:constrained-statement(tmcl:constraint: ?c, tmcl:constrained: $st)
  tmcl:allowed-reifier(tmcl:allows: ?c, tmcl:allowed: $tt)
end

def cannot-have-reifier($st)
  ?c isa tmcl:reifier-constraint;
    tmcl:card-min: 0;
    tmcl:card-max: 0.
  tmcl:constrained-statement(tmcl:constraint: ?c, tmcl:constrained: $st)
  tmcl:allowed-reifier(tmcl:allows: ?c, tmcl:allowed: tmdm:subject)
end

def may-have-reifier($st, $tt)
  ?c isa tmcl:reifier-constraint;
    tmcl:card-min: 0;
    tmcl:card-max: 1.
  tmcl:constrained-statement(tmcl:constraint: ?c, tmcl:constrained: $st)
  tmcl:allowed-reifier(tmcl:allows: ?c, tmcl:allowed: $tt)
end

# Topic Reifies Constraint
def must-reify($tt, $st)
  ?c isa tmcl:topic-reifies-constraint;
    tmcl:card-min: 1;
    tmcl:card-max: 1.
  tmcl:constrained-topic-type(tmcl:constraint: ?c, tmcl:constrained: $tt)
  tmcl:constrained-statement(tmcl:constraint: ?c, tmcl:constrained: $st)
end

def cannot-reify($tt)
  ?c isa tmcl:topic-reifies-constraint;
    tmcl:card-min: 0;
    tmcl:card-max: 0.
  tmcl:constrained-topic-type(tmcl:constraint: ?c, tmcl:constrained: $tt)
end

def may-reify($tt, $st)
  ?c isa tmcl:topic-reifies-constraint;
    tmcl:card-min: 0;
    tmcl:card-max: 1.
  tmcl:constrained-topic-type(tmcl:constraint: ?c, tmcl:constrained: $tt)
  tmcl:constrained-statement(tmcl:constraint: ?c, tmcl:constrained: $st)
end

# Association Role Constraint
def has-role($at, $rt, $min, $max)
  ?c isa tmcl:association-role-constraint;
```

```
    tmcl:card-min: $min;
    tmcl:card-max: $max.

  tmcl:constrained-statement(tmcl:constraint : ?c, tmcl:constrained : $at)
  tmcl:constrained-role(tmcl:constraint : ?c, tmcl:constrained : $rt)
end

# Role Combination Constraint
def role-combination($at, $rt, $tt, $ort, $ott)
  ?c isa tmcl:role-combination-constraint.
  tmcl:constrained-statement(tmcl:constraint: ?c, tmcl:constrained: $at)
  tmcl:constrained-role(tmcl:constraint: ?c, tmcl:constrained: $rt)
  tmcl:constrained-topic-type(tmcl:constraint: ?c, tmcl:constrained: $tt)
  tmcl:other-constrained-role(tmcl:constraint: ?c, tmcl:constrained: $ort)
  tmcl:other-constrained-topic-type(tmcl:constraint: ?c, tmcl:constrained: $ott)
end

# Occurrence Data Type Constraint
def has-datatype($ot, $dt)
  ?c isa tmcl:occurrence-datatype-constraint;
    tmcl:datatype: $dt.
  tmcl:constrained-statement(tmcl:constraint : ?c, tmcl:constrained : $ot)
end

# Unique Value Constraint
def has-unique-value($st)
  ?c isa tmcl:unique-value-constraint.
  tmcl:constrained-statement(tmcl:constraint : ?c, tmcl:constrained : $st)
end

# Regular Expression Constraint
def matches-regexp($st, $regexp)
  ?c isa tmcl:regular-expression-constraint;
    tmcl:regexp: $regexp.

  tmcl:constrained-statement(tmcl:constraint: ?c, tmcl:constrained: $st)
end

# Additional TMCL Templates
def binary-association($at, $rt1, $rt2)
  $at isa tmcl:association-type;
    has-role($rt1, 1, 1);
    has-role($rt2, 1, 1).

  $rt1 isa tmcl:role-type.
  $rt2 isa tmcl:role-type.
end

# Additional TMCL Templates
def symmetric-association($at, $rt)
  $at isa tmcl:association-type;
    has-role($rt, 2, 2).

  $rt isa tmcl:role-type.
end

# The Schema Topic
def belongs-to($construct, $schema)
  tmcl:belongs-to-schema(tmcl:container: $schema, tmcl:containee: $construct)
end
```

(Blank page)

Annex B
(normative)

TMCL meta-schema

This annex contains a TMCL schema for TMCL in CTM syntax, defining the allowed structure for TMCL schemas.

```
##########################################################################
# TMCL - Topic Maps Constraint Language
# This file is the TMCL schema that TMCL schemas must follow.
# This document is located at http://www.isotopicmaps.org/tmcl/schema.ctm
# and is normative.
##########################################################################

%include http://www.isotopicmaps.org/tmcl/templates.ctm
%prefix tmcl http://psi.topicmaps.org/tmcl/
%prefix tmdm http://psi.topicmaps.org/iso13250/model/
%prefix xsd http://www.w3.org/2001/XMLSchema#
%prefix iso http://psi.topicmaps.org/iso13250/

# THE SCHEMA ITSELF
tmcl-schema isa tmcl:schema;
  - "TMCL schema";
  tmcl:version: "2011-01-04";
  tmcl:description: "A meta-schema for TMCL, describing the allowed structure
                  of TMCL schemas.";
  tmcl:see-also: http://www.isotopicmaps.org/tmcl/tmcl.html;
  tmcl:schema-resource: http://www.isotopicmaps.org/tmcl/2011-01-04/schema.ctm .

# TOPIC TYPE
tmcl:topic-type isa tmcl:topic-type;
  - "Topic type";
  has-name(tmdm:topic-name, 0, *);
  has-occurrence(tmcl:description, 0, 1);
  has-occurrence(tmcl:comment, 0, *);
  has-occurrence(tmcl:see-also, 0, *);
  plays-role(tmcl:constrained, tmcl:constrained-topic-type, 0, *);
  plays-role(tmcl:containee, tmcl:belongs-to, 0, *);
  plays-role(tmcl:allowed, tmcl:overlaps, 0, *);
  plays-role(tmcl:constrained, tmcl:other-constrained-topic-type, 0, *);
  plays-role(tmcl:constrained, tmcl:constrained-scope, 0, *);
  plays-role(tmcl:allowed, tmcl:allowed-reifier, 0, *);
  overlaps(tmcl:name-type);
  overlaps(tmcl:occurrence-type);
  overlaps(tmcl:association-type);
  overlaps(tmcl:role-type).

# NAME TYPE
tmcl:name-type isa tmcl:topic-type;
  - "Name type";
  has-name(tmdm:topic-name, 0, *);
  has-occurrence(tmcl:description, 0, 1);
  has-occurrence(tmcl:comment, 0, *);
  has-occurrence(tmcl:see-also, 0, *);
  plays-role(tmcl:constrained, tmcl:constrained-statement, 0, *);
  plays-role(tmcl:constrained, tmcl:constrained-construct, 0, *);
  plays-role(tmcl:containee, tmcl:belongs-to, 0, *).

# OCCURRENCE TYPE
tmcl:occurrence-type isa tmcl:topic-type;
  - "Occurrence type";
  has-name(tmdm:topic-name, 0, *);
  has-occurrence(tmcl:description, 0, 1);
  has-occurrence(tmcl:comment, 0, *);
  has-occurrence(tmcl:see-also, 0, *);
  plays-role(tmcl:constrained, tmcl:constrained-statement, 0, *);
  plays-role(tmcl:constrained, tmcl:constrained-construct, 0, *);
  plays-role(tmcl:containee, tmcl:belongs-to, 0, *).

# ASSOCIATION TYPE
tmcl:association-type isa tmcl:topic-type;
  - "Association type";
  has-name(tmdm:topic-name, 0, *);
  has-occurrence(tmcl:description, 0, 1);
  has-occurrence(tmcl:comment, 0, *);
  has-occurrence(tmcl:see-also, 0, *);
  plays-role(tmcl:constrained, tmcl:constrained-statement, 0, *);
  plays-role(tmcl:constrained, tmcl:constrained-construct, 0, *);
  plays-role(tmcl:containee, tmcl:belongs-to, 0, *).

# ROLE TYPE
tmcl:role-type isa tmcl:topic-type;
  - "Role type";
  has-name(tmdm:topic-name, 0, *);
  has-occurrence(tmcl:description, 0, 1);
  has-occurrence(tmcl:comment, 0, *);
  has-occurrence(tmcl:see-also, 0, *);
  plays-role(tmcl:constrained, tmcl:constrained-construct, 0, *);
  plays-role(tmcl:constrained, tmcl:constrained-role, 0, *);
  plays-role(tmcl:constrained, tmcl:other-constrained-role, 0, *);
  plays-role(tmcl:containee, tmcl:belongs-to, 0, *).

# OVERLAP DECLARATION
tmcl:overlap-declaration isa tmcl:topic-type;
  - "Overlap declaration";
  has-occurrence(tmcl:description, 0, 1);
  has-occurrence(tmcl:comment, 0, *);
  has-occurrence(tmcl:see-also, 0, *);
  plays-role(tmcl:allows, tmcl:overlaps, 2, *);
  plays-role(tmcl:containee, tmcl:belongs-to, 0, 1).

# CONSTRAINT
tmcl:constraint isa tmcl:topic-type; isa tmcl:role-type;
  - "Constraint";
  is-abstract();
  has-occurrence(tmcl:description, 0, 1);
  has-occurrence(tmcl:comment, 0, *);
  has-occurrence(tmcl:see-also, 0, *);
  plays-role(tmcl:containee, tmcl:belongs-to, 0, 1).

# CARD-MIN and CARD-MAX
tmcl:card-min isa tmcl:occurrence-type;
  - "Minimum cardinality";
  has-datatype(xsd:integer).
```