ISO/IEC JTC 1/SC 34

Date: 2008-08-07

ISO/IEC FCD 19756

ISO/IEC JTC 1/SC 34/WG 3

Secretariat: JISC

Information technology — Topic Maps — Constraint Language (TMCL)

Élément introductif — Élément central — Élément complémentaire

Warning

This document is not an ISO International Standard. It is distributed for review and comment. It is subject to change without notice and may not be referred to as an International Standard.

Recipients of this draft are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

Document type: International Standard
Document subtype:
Document stage: (40) Enquiry
Document language: E

Contents

Page

Foreword

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

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

This International Standard defines a means to express and evaluate constraints on topic maps conforming to the Topic Map Data Model *[TMDM]*.

Topic Maps Constraint Language

1 Scope

This International Standard defines a data model for representing constraints on instances of the topic map data model and the formal semantics for the interpretation of different constraint types. This International Standard expresses constraints using topic map constructs and the interpretation of these constraints as TMQL. In addition, This International Standard defines a number of CTM templates to faciliate the construction of TMCL constraints.

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.

Unicode, *The Unicode Standard, Version 4.0*, The Unicode Consortium, Boston, MA, USA, Addison-Wesley , 2003, ISBN 0-321-18578-1

TMDM, *ISO 13250-2 Topic Maps — Data Model*, ISO, 2005, available at <http://www.isotopicmaps.org/sam/sam-model/>

TMQL, *ISO Topic Maps Query Language Working Draft*, ISO, 2005, available at <http://www.isotopicmaps.org/tmql/>

CTM, *ISO Compact Topic Map Notation Working Draft*, ISO, 2006, available at <http://www.isotopicmaps.org/ctm/>

XML 1.0, *Extensible Markup Language (XML) 1.0*, W3C, Third Edition, W3C Recommendation, 04 February 2004, available at <http://www.w3.org/TR/REC-xml/>

RFC3986, *RFC 3986 - Uniform Resource Identifiers (URI): Generic Syntax*, The Internet Society, 2005, available at <http://www.ietf.org/rfc/rfc3986.txt>

RFC3987, *RFC 3987 - Internationalized Resource Identifiers (IRIs)*, The Internet Society, 2005, available at <http://www.ietf.org/rfc/rfc3987.txt>

3 Notation and Conventions

3.1 Notation and Syntax

TMCL constraints are represented as topic map structures using the TMDM *[TMDM]*. Any syntax that can be used to create TMDM structures is a valid authoring syntax for TMCL constraints and therefor no special syntax is defined.

3.2 Formal Semantics

The formal semantics of TMCL constraints are defined using TMQL.

4 TMCL

TMCL defines constraint types and an interpretation for instances of those types. The interpretation indicates in an unambiguous fashion what it means for an instance of a given constraint type to be evaluated as true or false in the context of a TMDM instance.

The TMCL constraint types are defined in terms of the topic map data model. The formal interpretation of each constraint type is defined using TMQL.

All constraint types defined follow a common pattern. They are all defined as subtypes of the topic type called 'Constraint'. They all have an occurrence of type 'validation expression'. It is possible to define new constraint types that address specific domain requirements while still fitting into the overall TMCL validation framework. The constraint types defined in TMCL are intended for use in a entity constraint language fashion, such as ERM, UML etc. They are intended to be used to define the set of identities, occurrences, names and played association roles that a topic of a given type must have in order to been deemed valid e.g. topics of type person must have one unscoped name and be related to one other topic of type person who is their mother. The generalised constraint model, that of constraint and validation expression, is intended to facilitate map wide constraints that are not centered on one particular type. e.g. If a person works in department X then they must also be the author of at least 5 research papers.

4.1 Validation Semantics

A TMCL constraint type MUST have a validation expression occurrence. This occurrence contains the TMQL expression that formally defines what it means to evaluate instances of that constraint type in the context of a given topic map. The evaluation of this expression MUST return true or false. If the expression returns true, validation is said to have succeeded.

TMQL expressions that are contained in the validation expression occurrence may contain the unbound variable $THIS. $THIS is a placeholder for the constraint instance being validated.

For the purposes of evaluating the validation expression it is assumed that the constraint structure, the TMDM fragment representing the constraint, is virtually merged with the map being validated. However, the evaluating application must not permanently modify the topic map instance. This temporary, or virtual merging, is to allow the TMQL expression to acces both the constraint structure and the TMDM instance data at the same time.

If a given topic map is valid in respect to a constraint, then validation is said to have succeeded. More formally it can be said that :

```
Given:
  TopicMap: t
  Constraint  : c
Then:
  Validate(t, c) => true  | false
```

A schema is defined as a collection of constraints and if all constraints in that schema are valid then the schema can be said to be valid for a given topic map. More formally:

```
Given:
  TopicMap : t
  Schema : s

Then:
  Validate(t, s) =>
  ForAll Constraint c In s
    Validate(t, c)
  End
=> true | false
```

Constraints are independent from each other and can be evaluated in any order.

The following TMQL expressions are evaluated before each constraint is evaluated.

```
for $CARDMAX in $this / cardmax || MAX_INT
for $CARDMIN in $this / cardmin || 0
```

4.2 TMCL Syntax

TMCL does not define its own syntax. Each constraint type has a corresponding TMDM representation and thus any TMDM authoring syntax is considered a valid syntax for authoring TMCL constraints. To facilitate the authoring of TMCL from CTM this International Standard defines a number of templates. These templates exist in a resource 'http://www.topicmaps.org/tmcl/templates.ctm' that can be downloaded and included in any CTM file.

4.2.1 Including TMCL templates in a CTM document

To include the TMCL templates the following CTM import directive must be used.

```
%import http://www.topicmaps.org/tmcl/templates.ctm
```

4.2.2 TMCL namespace

The following namespace is defined in the template.ctm file.

```
%prefix tmcl http://psi.topicmaps.org/tmcl/
```

This International Standard also makes use of the Topic Map Data Model namespace.

```
%prefix tm http://psi.topicmaps.org/tmdm/
```

In addition this namespace is set to be the default namespace in the templates.ctm file.

```
%prefix http://psi.topicmaps.org/tmcl/
```

All following CTM fragments are assumed to be in the context of this namespace unless otherwise stated.

4.3 Metamodel

TMCL requires a metamodel in order to define constraint types using TMDM. This section defines the metamodel topics.

4.3.1 Topic Type

Instances of this topic are considered to be valid for use as topic types.

The topic type topic is defined as follows:

```
topictype isa topictype
```

4.3.2 Association Type

Instances of this topic are considered to be valid for use as association types.

The association type topic is defined as follows:

```
associationtype isa topictype
```

4.3.3 Occurrence Type

Instances of this topic are considered to be valid for use as occurrence types.

The occurrence type topic is defined as follows:

```
occurrencetype isa topictype
```

4.3.4 Role Type

Instances of this topic are considered to be valid for use as role types.

The role type topic is defined as follows:

```
roletype isa topictype
```

4.3.5 Name Type

Instances of this topic are considered to be valid for use as name types.

The name type topic is defined as follows:

```
nametype isa topictype
```

4.3.6 Constraint

This topic is used as the base type for all topics that are to be considered as constraint types.

The constraint type is defined as follows:

```
constraint isa topictype
```

4.3.7 Validation Expression

This topic is an occurrence type. It is used to type the occurrence on a constraint type that contains the TMQL expression used to evaluate constraint instances.

The validation expression occurrence type topic is defined as follows:

```
validation-expression isa occurrencetype
```

4.4 TMCL Constraint Types

The following sections contain the constraint types defined by TMCL. The model for each constraint type is defined using the TMCL model. The formal interpretation of the constraint is defined as an occurrence value in the form of TMQL.

4.4.1 Constraint

The *Constraint topic* is the root type of all constraint types in TMCL. It has one required occurrence of type validation-expression.

The constraint type is defined as:

```
constraint isa topictype
```

4.4.2 Topic Type Constraint

The *topic type constraint* provides a way to constrain that only topics explicitly defined as topic types can can play the role of type in the type instance relationship.

The topic type constraint is defined as:

```
topictype-constraint ako constraint
```

The value of the evaluation-expression occurrence and the evaluation semantics are defined as:

```
not (uniq ( // tm:topic >> types ) -- // tmcl:topicType)
```

There is no special CTM template for constructing instances of this constraint type. It can be created with the following line of CTM:

```
ttc isa topictype-constraint
```

4.4.3 Association Type Constraint

The *association type constraint* provides a way to constrain that only topics explicitly defined as association types are used to type associations.

The association type constraint is defined as:

```
associationtype-constraint ako constraint
```

The value of the evaluation-expression occurrence and the evaluation semantics are defined as:

```
not (uniq ( // tm:association >> types ) -- // tmcl:associationtype)
```

There is no special CTM template for constructing instances of this constraint type. It can be created with the following line of CTM:

```
atc isa associationtype-constraint
```

4.4.4 Association Role Type Constraint

The *association role type constraint* provides a way to constrain that only topics explicitly defined as association role types are used to type roles.

The association role type constraint is defined as:

```
roletype-constraint ako constraint
```

The value of the evaluation-expression occurrence and the evaluation semantics are defined as:

```
not (uniq ( // tm:role >> types ) -- // tmcl:roletype)
```

There is no special CTM template for constructing instances of this constraint type. It can be created with the following line of CTM:

```
rtc isa roletype-constraint
```

4.4.5 Occurrence Type Constraint

The *occurrence type constraint* provides a way to constrain that only topics explicitly defined as occurrence types are used to type occurrences.

The occurrence type constraint is defined as:

```
occurrencetype-constraint ako constraint
```

The value of the evaluation-expression occurrence and the evaluation semantics are defined as:

```
not (uniq ( // tm:occurrence >> types ) -- // tmcl:occurrenceType)
```

There is no special CTM template for constructing instances of this constraint type. It can be created with the following line of CTM:

```
otc isa occurrencetype-constraint
```

4.4.6 Name Type Constraint

The *name type constraint* provides a way to constrain that only topics explicitly defined as name types are used to type names.

The association role type constraint is defined as:

```
nametype-constraint ako constraint
```

The value of the evaluation-expression occurrence and the evaluation semantics are defined as:

```
not (uniq ( // tm:name >> types ) -- // tmcl:nameType)
```

There is no special CTM template for constructing instances of this constraint type. It can be created with the following line of CTM:

```
ntc isa nametype-constraint
```

4.4.7 Abstract Topic Type Constraint

The *abstract topic type constraint* provides a way to express that a given topic type is abstract and must therefor not have any direct instances. Direct instances are those instances connected to the topic itself and not infered via the transitive nature of the supersubtype association type.

Instances of this constraint type have the following structure. An association between the constraint instance and the topic type that is defined as being abstract. The association type is applies-to. The constraint topic plays the role of constraint-role, and the topic type plays the role of topictype-role.

The abstract topic type constraint is defined as:

```
abstract-topictype-constraint ako constraint
```

The following CTM shows a sample instance:

```
c isa abstract-topictype-constraint
applies-to(c : constraint-role, $topictype : topictype-role)
```

The value of the evaluation-expression occurrence and the evaluation semantics are defined as:

```
%taxonometrics off
for $atc in //abstract-topictype-constraint
where
   $atc <- constraint-role -> topictype-role >> instances
   return $atc
```

The following CTM template is defined:

```
def isAbstract($topictype)
   ?c isa abstract-topictype-constraint
   applies-to(?c : constraint-role, $topictype : topictype-role)
end
```

Example usage:

```
# create a topic type and use the isAbstract CTM template to define it as abstract
vehicle isa topictype
       isAbstract()

# create a substype of vehicle.
car iko vehicle
```

4.4.8 Exclusive Type Constraint

An *exclusive type constraint* provides a way to express that two topic types cannot have the same instance.

Instances of this type have the following structure. An association between each of the topic types that must have exclusive instances and the constraint instance. The association type is applies-to. The constraint topic plays the role of constraint-role, and the topic type plays the role of topictype-role.

The exclusive type constraint is defined as:

```
exclusive-instance ako constraint
```

The following CTM shows a sample instance

```
c isa exclusive-instance
atype isa topictype
btype isa topictype
applies-to(c : constraint-role, atype : topictype-role)
applies-to(c : constraint-role, btype : topictype-role)
```

The value of the evaluation-expression occurrence and the evaluation semantics are defined as:

```
for $atc in // exclusive-constraint
       for $tt in $atc <- constraint-role -> topictype-role
       where
              uniq ($tt >> instances >> types) -- $tt << supertypes
              return $atc
```

The following CTM template is defined:

```
def exclusive-with($atype, $btype)
   ?c isa exclusive-instance
   applies-to(?c : constraint-role, $atype : topictype-role)
   applies-to(?c : constraint-role, $btype : topictype-role)
end
```

Example usage:

```
vehicle isa topictype

person isa topictype
    exclusive-with(vehicle)
```

4.4.9 Subject Identifier Constraint

A *subject identifier constraint* provides a way to constrain the subject identifiers of a topic of a given type.

Instances of this type have the following structure. An association between the constraint topic and the topic type to which the constraint applies. The association type is applies-to. The constraint topic plays the role of constraint-role, and the topic type plays the role of topictype-role. In addition, the constraint may have the following occurrences;

- card-min indicating the minimum number of subject identifiers a valid instance must have
- card-max indicating the maximum number of subject identifiers a valid instance must have
- regexp, a regular expression that must be valid in terms of the string representation of the subject identifier IRI.

The subject identifier constraint is defined as:

```
subjectidentifier-constraint ako constraint
```

The following CTM shows a sample instance

```
atype isa topictype

c isa subjectidentifier-constraint
    card-min: 0
    card-max: 1
    regexp: """*"""

applies-to(c : constraint-role, atype : topictype-role)
```

The value of the evaluation-expression occurrence and the evaluation semantics are defined as:

```
for $CARDMAX in $this / cardmax || MAX_INT
for $CARDMIN in $this / cardmin || 0

for $sic in // subjectidentifier-constraint
    for $instance in $sic <- constraint-role -> topictype-role >> instances
        where
            $sic / cardmin & count($instance ~) < $sic / cardmin
            |
            $sic / cardmax & count($instance ~) > $sic / cardmax
            |
            $sic / regexp  &
                not every $si in $instance~
                    satifies
                        $si =~ $sic / regexp
        return $sic
```

The following CTM template is defined:

```
def has-subjectidentifier($topictype, $cardmin, $cardmax, $expression)
       ?c isa subjectidentifier-constraint
            card-min: $cardmin
            card-max: $cardmax
            reg-exp:  $expression
       applies-to(?c : constraint-role, $topictype : topictype-role)
end
```

Example usage:

```
# define topic type with subject identifier constraint
person isa topictype
       has-subjectidentifier(1, 1, "*")
```

4.4.10 Subject Locator Constraint

A *subject locator constraint* provides a way to constrain the subject locators of a topic of a given type.

Instances of this type have the following structure. An association between the constraint topic and the topic type to which the constraint applies. The association type is applies-to. The constraint topic plays the role of constraint-role, and the topic type plays the role of topictype-role. In addition, the constraint may have the following occurrences;

- card-min indicating the minimum number of subject locators a valid instance must have
- card-max indicating the maximum number of subject locators a valid instance must have
- regexp, a regular expression that must be valid in terms of the string representation of the subject locator IRI.

The subject locator constraint is defined as:

```
subjectlocator-constraint ako constraint
```

The following CTM shows a sample instance

```
  atype isa topictype

  c isa subjectlocator-constraint
       card-min: 0
       card-max: 1
       regexp: """*"""

  applies-to(c : constraint-role, atype : topictype-role)
```

The value of the evaluation-expression occurrence and the evaluation semantics are defined as:

```
for $CARDMAX in $this / cardmax || MAX_INT
for $CARDMIN in $this / cardmin || 0
for $slc in // subjectlocator-constraint
    for $instance in $slc <- constraint-role -> topictype-role >> instances
        where
            $slc / cardmin & count($instance ~) < $slc / cardmin
            |
            $slc / cardmax & count($instance ~) > $slc / cardmax
            |
            $slc / regexp  &
                not every $sl in $instance~
                    satifies
                        $sl =~ $slc / regexp
        return $slc
```

The following CTM template is defined:

```
def has-subjectlocator($topictype, $cardmin, $cardmax, $expression)
    ?c isa subjectlocator-constraint
       card-min: $cardmin
       card-max: $cardmax
       reg-exp:  $expression
    applies-to(?c : constraint-role, $topictype : topictype-role)
end
```

Example usage:

```
# define topic type with subject locator constraint
docuemnt isa topictype
    has-subjectloctor(1, 1, "file://*")
```

4.4.11 Topic Name Constraint

A *topic name constraint* provides a way to constrain the type and cardinality of topic names for instances of a given topic type.

Instances of this type have the following structure.

- An association between the constraint topic and the topic type to which the constraint applies. The association type is applies-to. The constraint topic plays the role of constraint-role, and the topic type plays the role of topictype-role

- An association between the constraint topic and a name type. The association type is applies-to. The constraint topic plays the role of constraint-role, and the name type plays the role of nametype-role. This association is to indicate the type of name to which the constraint applies.

In addition, the constraint may have the following occurrences;

- card-min indicating the minimum number of names a valid instance must have

- card-max indicating the maximum number of names a valid instance must have

- regexp, a regular expression that must match the string value of the name value property.

The topicname-constraint constraint is defined as:

```
topicname-constraint ako constraint
```

The following CTM shows a sample instance

```
  aname isa nametype
  atype isa topictype

  c isa topicname-constraint
        card-min: 0
        card-max: 1
        regularexpr: """*"""

  applies-to(c : constraint-role, atype : topictype-role)
  applies-to(c : constraint-role, aname : nametype-role)
```

The value of the evaluation-expression occurrence and the evaluation semantics are defined as:

```
for $tnc in // topicname-constraint
    for $instance in $tnc <- constraint-role -> topictype-role >> instances
        for $nametype in $tnc <- constraint-role -> nametype-role
        where
            $tnc / cardmin & count($instance / $nametype) < $tnc / cardmin
            |
            $tnc / cardmax & count($instance / $nametype) > $tnc / cardmax
            |
            $tnc / regexp  &
                not every $name in $instance / $nametype
                    satifies
                        $name =~ $tnc / regexp
        return $tnc
```

The following CTM template is defined:

```
def has-name($topictype, $nametype, $cardmin, $cardmax, $expression)
   ?c isa topicname-constraint
        card-min: $cardmin
        card-max: $cardmax
        regularexpr: $expression
   applies-to(?c : constraint-role, $topictype : topictype-role)
   applies-to(?c : constraint-role, $nametype : nametype-role)
end
```

Example usage:

```
person isa topictype
    has-name(shortname, 1, 1, """*""")
```

4.4.12 Name Type Scope Constraint

A *name type scope constraint* provides a way to constrain the type and number of scoping topics required on a name of a given type.

Instances of this type have the following structure.

- An association between the constraint topic and the name type to which the constraint applies. The association type is applies-to. The constraint topic plays the role of constraint-role, and the name type plays the role of nametype-role
- An association between the constraint topic and a scope type. The association type is applies-to. The constraint topic plays the role of constraint-role, and the scope type plays the role of scopetype-role.

In addition, the constraint may have the following occurrences;

- card-min indicating the minimum number of scoping topics that must be present
- card-max indicating the maximum number of scoping topics that must be present

The name type scope constraint is defined as:

```
nametypescope-constraint ako constraint
```

The following CTM shows a sample instance

```
 st isa scopetype
 nt isa nametype

 c isa nametypescope-constraint
           card-min: 0
           card-max: 1

 applies-to(c : constraint-role, nt : nametype-role)
 applies-to(c : constraint-role, st : scopetype-role)
```

The value of the evaluation-expression occurrence and the evaluation semantics are defined as:

```
// $nametype >> scope >> types == $nametype
for $ntsc in // nametypescope-constraint
      for $nametype in $ntsc <- constraint-role -> nametype-role
      for $name in $nametype >> instances

where
      $ntsc / cardmax & count ($name @ [. == $nametype]) > $ntsc / cardmax
      |
      $ntsc / cardmin & count ($name @ [. == $nametype]) < $ntsc / cardmin
      return $ntsc
```

The following CTM template is defined:

```
def has-name-scope($nametype, $scopetype, $cardmin, $cardmax)
   ?c isa nametypescope-constraint
          card-min: $cardmin
          card-max: $cardmax
   applies-to(?c : constraint-role, $nametype : nametype-role)
   applies-to(?c : constraint-role, $scopetype : scopetype-role)
end
```

Example usage:

```
language isa scopetype

displayname isa nametype
      has-scope(language, 0, 1)
```

4.4.13 Occurrence Type Scope Constraint

A *occurrence type scope constraint* provides a way to constrain the type and number of scoping topics required on an occurrence of a given type.

Instances of this type have the following structure.

- An association between the constraint topic and the occurrence type to which the constraint applies. The association type is applies-to. The constraint topic plays the role of constraint-role, and the name type plays the role of occurrencetype-role
- An association between the constraint topic and a scope type. The association type is applies-to. The constraint topic plays the role of constraint-role, and the scope type plays the role of scopetype-role.

In addition, the constraint may have the following occurrences;

- card-min indicating the minimum number of scoping topics that must be present
- card-max indicating the maximum number of scoping topics that must be present

The occurrence type scope constraint constraint is defined as:

```
occurrencetypescope-constraint ako constraint
```

The following CTM shows a sample instance

```
ot isa occurrencetype
st isa scopetype

c isa occurrencetypescope-constraint
      card-min: 0
      card-max: 1

applies-to(c : constraint-role, ot : occurrencetype-role)
applies-to(c : constraint-role, st : scopetype-role)
```

The value of the evaluation-expression occurrence and the evaluation semantics are defined as:

```
// $occurrencetype >> scope >> types == $occurrencetype
for $otsc in // occurrencetypescope-constraint
      for $occurrencetype in $ntsc <- constraint-role -> occurrencetype-role
      for $occurrence in $occurrencetype >> instances

where
      $otsc / cardmax & count ($occurrence @ [. == $occurrencetype]) > $otsc / cardmax
      |
      $otsc / cardmin & count ($occurrence @ [. == $occurrencetype]) < $otsc / cardmin
      return $ntsc
```

The following CTM template is defined:

```
def has-occurrence-scope($occurrencetype, $scopetype, $cardmin, $cardmax)
   ?c isa occurrencetypescope-constraint
            card-min: $cardmin
            card-max: $cardmax
   applies-to(?c : constraint-role, $occurrencetype : occurrencetype-role)
   applies-to(?c : constraint-role, $scopetype : scopetype-role)
end
```

Example usage:

```
securitylevel isa scopetype
documentlink isa occurrencetype
       has-occurrence-scope(securitylevel, 1, 1)
```

4.4.14 Association Type Scope Constraint

An *association type scope constraint* provides a way to constrain the type and number of scoping topics required on an association of a given type.

Instances of this type have the following structure.

An association between the constraint topic and the association type to which the constraint applies. The association type is applies-to. The constraint topic plays the role of constraint-role, and the name type plays the role of associationtype-role

An association between the constraint topic and a scope type. The association type is applies-to. The constraint topic plays the role of constraint-role, and the scope type plays the role of scopetype-role.

In addition, the constraint may have the following occurrences;

— card-min indicating the minimum number of scoping topics that must be present

— card-max indicating the maximum number of scoping topics that must be present

The association type scope constraint is defined as:

```
associationtypescope-constraint ako constraint
```

The following CTM shows a sample instance

```
at isa associationtype
st isa scopetype

c isa associationtypescope-constraint
       card-min: 0
       card-max: 1

applies-to(c : constraint-role, at : associationtype-role)
applies-to(c : constraint-role, st : scopetype-role)
```

The value of the evaluation-expression occurrence and the evaluation semantics are defined as:

```
// $associationtype >> scope >> types == $associationtype
for $atsc in // associationtypescope-constraint
      for $associationtype in $atsc <- constraint-role -> associationtype-role
      for $association in $associationtype >> instances

where
      $atsc / cardmax & count ($association @ [. == $associationtype]) > $atsc / cardmax
      |
      $atsc / cardmin & count ($association @ [. == $associationtype]) < $atsc / cardmin
      return $atsc
```

The following CTM template is defined:

```
def has-association-scope($associationtype, $scopetype, $cardmin, $cardmax)
   ?c isa associationtypescope-constraint
           card-min: $cardmin
           card-max: $cardmax
   applies-to(?c : constraint-role, $associationtype : associationtype-role)
   applies-to(?c : constraint-role, $scopetype : scopetype-role)
end
```

Example usage:

```
timeperiod isa scopetype
personworksforcompany isa associationtype
      has-association-scope(timeperiod, 1, 1)
```

4.4.15 Occurrence Data Type Constraint

An *occurrence data type constraint* provides a way to constrain the allowed datatype of an occurrence of a given type.

Instances of this type have the following structure. An association between the constraint topic and the occurrence type to which the constraint applies. The association type is applies-to. The constraint topic plays the role of constraint-role, and the topic type plays the role of occurrencetype-role. In addition, the constraint may have the following occurrences;

— datatype an IRI that must match the datatype property of an occurrence of the specified type

The occurrence data type constraint is defined as:

```
occurrencedatatype-constraint ako constraint
```

The following CTM shows a sample instance

```
ot isa occurrencetype
c isa occurrencedatatype-constraint
      datatype: IRI
applies-to(c : constraint-role, ot : occurrencetype-role)
```

The value of the evaluation-expression occurrence and the evaluation semantics are defined as:

```
for $odtc in // occurrencedatatype-constraint
    for $occurrencetype in $odtc <- constraint-role -> occurrencetype-role
        for $occurrence in $occurrencetype >> instances
        where
            not (datatype($occurrence) == $odtc / datatype)
        return $odtc
```

The following CTM template is defined:

```
def has-datatype($occurrencetype, $datatype)
   ?c isa occurrencedatatype-constraint
         datatype: $datatype
   applies-to(?c : constraint-role, $occurrencetype : occurrencetype-role)
end
```

Example usage:

```
age isa occurrencetype
    has-datatype(http://www.xmlschema.com/types/integer)
```

4.4.16 Topic Occurrence Constraint

A *topic occurrence constraint* defines a way to constrain the type, cardinality and value of occurrences connected to a topic of a given type.

Instances of this type have the following structure.

- An association between the constraint topic and the topic type to which the constraint applies. The association type is applies-to. The constraint topic plays the role of constraint-role, and the topic type plays the role of topictype-role

- An association between the constraint topic and an occurrence type. The association type is applies-to. The constraint topic plays the role of constraint-role, and the occurrence type plays the role of occurrencetype-role. This association is to indicate the type of occurrence to which the constraint applies.

In addition, the constraint may have the following occurrences;

- card-min indicating the minimum number of occurrences a valid instance must have

- card-max indicating the maximum number of occurrences a valid instance must have

- regexp a regular expression that must match the string value of the occurrence value property.

The topic name constraint is defined as:

```
topicname-constraint ako constraint
```

The following CTM shows a sample instance

```
ot isa occurrencetype
tt isa topictype

c isa topicname-constraint
        card-min: 0
        card-max: 1
        regularexpr: """*"""

applies-to(c : constraint-role, tt : topictype-role)
applies-to(c : constraint-role, ot : occurrencetype-role)
```

The value of the evaluation-expression occurrence and the evaluation semantics are defined as:

```
for $toc in // topicname-constraint
    for $instance in $toc <- constraint-role -> occurrencetype-role >> instances
        for $occurrencetype in $tnc <- constraint-role -> occurrencetype-role
        where
            $toc / cardmin & count($instance / $occurrencetype) < $toc / cardmin
            |
            $toc / cardmax & count($instance / $occurrencetype) > $toc / cardmax
            |
            $toc / regexp  &
                not every $occurrence in $instance / $occurrencetype
                    satifies
                        $occurrence =~ $toc / regexp
            return $toc
```

The following CTM template is defined:

```
def has-occurrence($topictype, $occurrencetype, $cardmin, $cardmax, $expression)
  ?c isa topicoccurrence-constraint
       card-min: $cardmin
       card-max: $cardmax
       regexp: $expression

    applies-to(?c : constraint-role, $topictype : topictype-role)
  applies-to(?c : constraint-role, $occurrencetype : occurrencetype-role)
end
```

Example usage:

```
article isa topictype
    has-occurrence(summary, 0, 1, """*""")
```

4.4.17 Association Role Constraint

A *association role constraint* defines how many and of what type the roles in associations of given a type must be.

Instances of this type have the following structure.

- An association between the constraint topic and the association type to which the constraint applies. The association type is applies-to. The constraint topic plays the role of constraint-role, and the topic type plays the role of associationtype-role

- An association between the constraint topic and a role type. The association type is applies-to. The constraint topic plays the role of constraint-role, and the role type plays the role of roletype-role. This association is to indicate the type of role to which the constraint applies.

In addition, the constraint may have the following occurrences;

- card-min indicating the minimum number of times the role can appear in the association
- card-max indicating the maximum number of times the role can appear in the association

The association role constraint is defined as:

```
associationrole-constraint ako constraint
```

The following CTM shows a sample instance

```
at isa associationtype
rt isa roletype

c isa associationrole-constraint
     card-min: $cardmin
     card-max: $cardmax

applies-to(c : constraint-role, at : associationtype-role)
applies-to(c : constraint-role, rt : roletype-role)
```

The value of the evaluation-expression occurrence and the evaluation semantics are defined as:

```
for $asrc in // associationrole-constraint
        for $associationtype in $arsc <- constraint-role -> associationtype-role
                for $roletype in $arsc <- constraint-role -> associationroletype-role
                        for $association in $associationtype >> instances
                        where
                                 $asrc / cardmin & count ($association >> roles [. == $roletype]) < $arsc / cardmin
                                 $asrc / cardmax & count ($association >> roles [. == $roletype]) > $arsc / cardmax
                         return $asrc
```

The following CTM template is defined:

```
def has-role($assoctype, $roletype, $cardmin, $cardmax)
    ?c isa associationrole-constraint
             card-min: $cardmin
             card-max: $cardmax
    applies-to(?c : constraint-role, $assoctype : assoctype-role)
    applies-to(?c : constraint-role, $roletype : roletype-role)
end
```

Example usage:

```
employee isa roletype
employer isa roletype

worksfor isa associationtype
```

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

4.4.18 Role Player Constraint

A *role player constraint* defines a way to constrain the type of allowed role players of a given role with a given type in an association of a given type..

Instances of this type have the following structure.

- An association between the constraint topic and the topic type to which the constraint applies. The association type is applies-to. The constraint topic plays the role of constraint-role, and the topic type plays the role of topictype-role
- An association between the constraint topic and an association type. The association type is applies-to. The constraint topic plays the role of constraint-role, and the association type plays the role of associationtype-role.
- An association between the constraint topic and a role type. The association type is applies-to. The constraint topic plays the role of constraint-role, and the role type plays the role of roletype-role.

In addition, the constraint may have the following occurrences;

- card-min indicating the minimum number of times a topic can play the a role with the given type in the given an association of the given type
- card-max indicating the maximum number of times a topic can play the a role with the given type in the given an association of the given type

The role player constraint is defined as:

```
roleplayer-constraint ako constraint
```

The following CTM shows a sample instance

```
at isa associationtype
tt isa topictype
rt isa roletype

c isa roleplayer-constraint
   card-min: 0
   card-max: 1
      applies-to(c : constraint-role, at : assoctype-role)
      applies-to(c : constraint-role, rt : roletype-role)
      applies-to(c : constraint-role, tt : topictype-role)
```

The value of the evaluation-expression occurrence and the evaluation semantics are defined as:

```
for $rpc in // roleplayercardinality-constraint
      for $associationtype in $rpc <- constraint-role -> associationtype-role
            for $roletype in $rpc <- constraint-role -> associationroletype-role
                  for $topictype in $rpcc <- constraint-role -> topictype-role
                  where
                        $rpc / cardmax & count
                              ($topictype >> instances <- $roletype [^$associationtype]) > $rpcc / cardmax
                        |
                        $rpc / cardmin & count
```

```
                    ($topictype >> instances <- $roletype [^$associationtype]) > $rpcc / cardmin
          return $rpc
```

The following CTM template is defined:

```
def plays-role($assoctype, $roletype, $topictype, $cardmin, $cardmax)
   ?c isa roleplayer-constraint
      card-min: $cardmin
      card-max: $cardmax
            applies-to(?c : constraint-role, $assoctype : assoctype-role)
            applies-to(?c : constraint-role, $roletype  : roletype-role)
            applies-to(?c : constraint-role, $topictype : topictype-role)
end
```

Example usage:

```
employee isa roletype
employer isa roletype

worksfor isa associationtype
      has-role(employee, 1, 1)
      has-role(employer, 1, 1)

person isa topictype
      plays-role(employee, worksfor, 1)

company isa topictype
      plays-role(employer, worksfor, MAX_INT)
```

4.4.19 Other Role Constraint

A *other role constraint* defines a way to constrain the allowed type of role player given the existence of another role player type in an association..

Instances of this type have the following structure.

- An association between the constraint topic and one topic type to which the constraint applies. The association type is applies-to. The constraint topic plays the role of constraint-role, and the topic type plays the role of topictype-role

- An association between the constraint topic and an association type. The association type is applies-to. The constraint topic plays the role of constraint-role, and the association type plays the role of associationtype-role.

- An association between the constraint topic and one role type. The association type is applies-to. The constraint topic plays the role of constraint-role, and the role type plays the role of roletype-role.

- An association between the constraint topic and the other topic type to which the constraint applies. The association type is applies-to. The constraint topic plays the role of constraint-role, and the topic type plays the role of othertopictype-role

- An association between the constraint topic and the other role type. The association type is applies-to. The constraint topic plays the role of constraint-role, and the role type plays the role of otherroletype-role.

In addition, the constraint may have the following occurrences;

- card-min indicating the minimum number of times the other role can occur

- card-max indicating the maximum number of times the other role can occur

The other role constraint is defined as:

```
otherrole-constraint ako constraint
```

The following CTM shows a sample instance

```
at isa associationtype
tt1 isa topictype
tt2 isa topictype

rt1 isa roletype
rt2 isa roletype

c isa otherrole-constraint
     card-min: $cardmin
     card-max: $cardmax

applies-to(c : constraint-role, at : assoctype-role)

applies-to(c : constraint-role, rt1 : roletype-role)
applies-to(c : constraint-role, tt1 : topictype-role)

applies-to(c : constraint-role, rt2 : otherroletype-role)
applies-to(c : constraint-role, tt2 : othertopictype-role)
```

The value of the evaluation-expression occurrence and the evaluation semantics are defined as:

```
for $associationtype in $orc <- constraint-role -> associationtype-role
    for $roletype in $orc <- constraint-role -> associationroletype-role
        for $topictype in $orc <- constraint-role -> topictype-role
            for $othertopictype in $orc <- constraint-role -> othertopictype-role
                for $otherroletype in $orc <- constraint-role -> otherroletype-role
                    for $association in $associationtype >> instances
                    where
                        $association -> $roletype isa $topictype
                        &
                        (
                        count ($association -> $otherroletype isa $othertopictype) > $CARDMAX
                        |
                        count  ($association -> $otherroletype isa $othertopictype) < $CARDMIN
                        )
                    return false
```

The following CTM template is defined:

```
def other-role($roletype, $assoctype, $topictype, $otherroletype, $othertopictype, $cardmin, $cardmax)
   ?c isa otherrole-constraint
         card-min: $cardmin
         card-max: $cardmax

      applies-to(?c : constraint-role, $assoctype : assoctype-role)

   applies-to(?c : constraint-role, $roletype : roletype-role)
      applies-to(?c : constraint-role, $topictype : topictype-role)
```

```
        applies-to(?c : constraint-role, $otherroletype : otherroletype-role)
        applies-to(?c : constraint-role, $othertopictype : topictype-role)
    end
```

Example usage:

```
locatedin isa associationtype

city isa topictype
country isa topictype
continent isa topictype
region isa topictype

containee isa roletype

container isa roletype
  other-role(locatedin, country, containee, city, 1, *)
```

4.4.20 Unique Occurrence Constraint

A *unique occurrence constraint* defines a way to constrain that occurrences of a given type with the same value can only be present once.

Instances of this type have the following structure.

— An association between the constraint topic and the topic type to which the constraint applies. The association type is applies-to. The constraint topic plays the role of constraint-role, and the topic type plays the role of topictype-role

— An association between the constraint topic and an occurrence type. The association type is applies-to. The constraint topic plays the role of constraint-role, and the occurrence type plays the role of occurrencetype-role. This association is to indicate the type of occurrence to which the constraint applies.

In addition, the constraint may have the following occurrences;

— card-min indicating the minimum number of names a valid instance must have

— card-max indicating the maximum number of names a valid instance must have

— regexp a regular expression that must match the string value of the occurrence value property.

The unique occurrence constraint is defined as:

```
uniqueoccurrence-constraint ako constraint
```

The following CTM shows a sample instance

```
tt isa topictype
ot isa occurrencetype

c isa uniqueoccurrence-constraint

applies-to(?c : constraint-role, tt : topictype-role)
applies-to(?c : constraint-role, ot : occurrencetype-role)
```

The value of the evaluation-expression occurrence and the evaluation semantics are defined as:

```
for $uoc in // uniqueoccurrence-constraint
        for $occurrencetype in $uoc <- constraint-role -> occurrencetype-role
            for $topictype in $uoc <- constraint-role -> topictype-role
                for $occurrence in $occurrencetype
                where
                    not (count ($occurrence \ $occurrencetype [^$topictype]) == 1)
                return $uoc
```

The following CTM template is defined:

```
def unique-occurrence($topictype, $occurrencetype)
  ?c isa uniqueoccurrence-constraint
    applies-to(?c : constraint-role, $topictype : topictype-role)
  applies-to(?c : constraint-role, $occurrencetype : occurrencetype-role)
end
```

Example usage:

```
emailaddress isa occurrencetype
person isa topictype
    unique-occurrence(emailaddress)
```

5 Additional TMCL Templates

The following templates are defined to aid authoring:

```
def BinaryAssociation($assoctype, $r1type, $r2type, $r1playertype, $r2playertype)
 AssociationRoleConstraint($assoctype, $r1type, 1, 1)
 AssociationRoleConstraint($assoctype, $r2type, 1, 1)

 RolePlayerConstraint($assoctype, $r1type, $r1playertype)
 RolePlayerConstraint($assoctype, $r2type, $r2playertype)
end

def associated-to($r1type, $r2type, $assoctype, $r1playertype, $r2playertype)
 AssociationRoleConstraint($assoctype, $r1type, 1, 1)
 AssociationRoleConstraint($assoctype, $r2type, 1, 1)

 RolePlayerConstraint($assoctype, $r1type, $r1playertype)
 RolePlayerConstraint($assoctype, $r2type, $r2playertype)
end
```

6 TMCL Extensions

The TMCL data model in its most basic form supports any TMQL expression that returns true or false to be used as a constraint predicate.

A new constraint type can be created by creating a subtype of constraint and ensuring that the new type has a validation-expression occurrence. When evaluating this constraint type the value of the validation-expression occurrence will be evaluated as TMQL as described in the validation semantics clause above.

7 TMCL Conformance

There are two levels of conformance defined by this International Standard.

7.1 TMCL Level One Conformance

Level one conformance includes all processors that can validate a topic map against a schema where that schema is composed only of constraint types defined in This International Standard.

7.2 TMCL Level Two Conformance

Level two conformance is defined as processors that can validate a topic map against a schema where that schema uses any type of constraint, providing the constraint meets the basic criteria of having a TMQL expression occurrence value.

8 TMCL Templates CTM

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

```
################################################################################
# TMCL - Topic Map Constraint Language
# This file defines the set of templates defined by TMCL document no: N458
# This document is located at http://www.topicmaps.org/tmcl/templates.ctm
# and should be considered normative.
################################################################################

# the TMCL prefix declaration.
%prefix tmcl http://psi.topicmaps.org/tmcl/

# the TMDM prefic declaration
%prefix tm http://psi.topicmaps.org/tmdm/model

# use tmcl as the default prefix
%prefix http://psi.topicmaps.org/tmcl/

################################################################################
# TMCL meta-model topics
################################################################################

topictype isa topictype

occurrencetype isa topictype

associationtype isa topictype

roletype isa topictype

nametype isa topictype

scopetype isa topictype

# role types used to include the topictype metatypes in associations
topictype-role isa roletype

occurrencetype-role isa roletype
```

```
associationtype-role isa roletype

roletype-role isa roletype

otherroletype-role isa roletype

nametype-role isa roletype

scopetype-role isa roletype

constraint-role isa roletype

########################################################################
# TMCL model topics
########################################################################

# the constraint topic is the common supertype of all constraint types
# defined by TMCL.
constraint isa topictype

# this occurrence type is used to type a single occurrence on each
# constraint type. This occurrence holds the TMQL value used to
# evaluate constraint instances for validity.
validation-expression isa occurrencetype

# the association type used to bind different components into a
# constraint
applies-to isa associationtype

# the card min facet is used on many constraint types
card-min isa occurrencetype

# the card max facet is used on many constraint types
card-max isa occurrencetype

# the reg exp facet is used on many constraint types
regexp isa occurrencetype

# the datatype facet is used on many constraint types
datatype isa occurrencetype

########################################################################
# TMCL constraint types
########################################################################

#TopicType constraint
topictype-constraint ako constraint
            validation-expression """
                       not (uniq ( // tm:topic >> types ) -- // tmcl:topictype)
                        """

########################################################################

#AssociationType constraint
associationtype-constraint ako constraint
               validation-expression """
                       not (uniq ( // tm:association >> types ) -- // tmcl:associationtype)
                            """

########################################################################

#AssociationRoleType constraint
roletype-constraint ako constraint
           validation-expression """
                       not (uniq ( // tm:roletype >> types ) -- // tmcl:roletype)
```

```
                              """

##########################################################################

# OccurrenceType constraint
occurrencetype-constraint ako constraint
                  validation-expression """
                            not (uniq ( // tm:occurrence >> types ) -- // tmcl:occurrencetype)
                                                              """

##########################################################################

#NameType constraint
nametype-constraint ako constraint
              validation-expression """
                            not (uniq ( // tm:name >> types ) -- // tmcl:nametype)
                                                           """

##########################################################################

#Abstract topic type constraint
abstract-topictype-constraint ako constraint
                     validation-expression """
                                 %taxonometrics off
                                 for $atc in //abstract-topictype-constraint
                                 where
                                      $atc <- constraint-role -> topictype-role >> instances
                                      return $atc
                                   """

#Abstract topic type constraint construction template
def is-abstract($topictype)
   ?c isa abstract-topictype-constraint
   applies-to(?c : constraint-role, $topictype : topictype-role)
end

##########################################################################

# Exclusive Instance type and evaluation semantics
exclusive-instance ako constraint
               validation-expression """
                            for $atc in // exclusive-constraint
                                 for $tt in $atc <- constraint-role -> topictype-role
                                 where
                                      uniq ($tt >> instances >> types) -- $tt << supertypes
                                      return $atc
                                           """

# constraint construction template
def exclusive-with($atype, $btype)
    ?c isa exclusive-instance
    applies-to(?c : constraint-role, $atype : topictype-role)
    applies-to(?c : constraint-role, $btype : topictype-role)
end

##########################################################################

# subject identifier constraint
subjectidentifier-constraint ako constraint
                    validation_expression
                    """
        for $CARDMAX in $this / cardmax || MAX_INT
                for $CARDMIN in $this / cardmin || 0

                for $sic in // subjectidentifier-constraint
                     for $instance in $sic <- constraint-role -> topictype-role >> instances
                    where
                        $sic / cardmin & count($instance ~) < $sic / cardmin
                        |
                        $sic / cardmax & count($instance ~) > $sic / cardmax
                        |
                        $sic / regexp  &
                            not every $si in $instance~
                                satifies
                                    $si =~ $sic / regexp
                    return $sic
                            """

# constraint construction template
def has-subjectidentifier($topictype, $cardmin, $cardmax, $expression)
    ?c isa subjectidentifier-constraint
       card-min: $cardmin
       card-max: $cardmax
       reg-exp:  $expression
    applies-to(?c : constraint-role, $topictype : topictype-role)
end

########################################################################

subjectlocator-constraint ako constraint
              validation_expression
              """
      for $CARDMAX in $this / cardmax || MAX_INT
            for $CARDMIN in $this / cardmin || 0

            for $slc in // subjectlocator-constraint
                for $instance in $slc <- constraint-role -> topictype-role >> instances
                    where
                        $slc / cardmin & count($instance ~) < $slc / cardmin
                        |
                        $slc / cardmax & count($instance ~) > $slc / cardmax
                        |
                        $slc / regexp  &
                            not every $sl in $instance~
                                satifies
                                    $sl =~ $sic / regexp
                    return $slc
                            """

# constraint construction template
def has-subjectlocator($topictype, $cardmin, $cardmax, $expression)
  ?c isa subjectlocator-constraint
       card-min: $cardmin
       card-max: $cardmax
     reg-exp:  $expression

    applies-to(?c : constraint-role, $topictype : topictype-role)
end

########################################################################

# topicname constraint
topicname-constraint ako constraint
          validation-expression: """
                for $tnc in // topicname-constraint
                    for $instance in $tnc <- constraint-role -> topictype-role >> instances
                        for $nametype in $tnc <- constraint-role -> nametype-role
                        where
                            $tnc / cardmin & count($instance / $nametype) < $tnc / cardmin
                            |
                            $tnc / cardmax & count($instance / $nametype) > $tnc / cardmax
                            |
                                        $tnc / regexp  &
                                            not every $name in $instance / $nametype
                                                satifies
                                                    $name =~ $tnc / regexp
                                    return $tnc
                """

# constraint construction template
def has-name($topictype, $nametype, $cardmin, $cardmax, $expression)
   ?c isa topicname-constraint
        card-min: $cardmin
        card-max: $cardmax
        regularexpr: $expression
   applies-to(?c : constraint-role, $topictype : topictype-role)
   applies-to(?c : constraint-role, $nametype : nametype-role)
end

########################################################################

# nametypescope constraint
nametypescope-constraint ako constraint
            validation-expression

            """
                // $nametype >> scope >> types == $nametype
                for $ntsc in // nametypescope-constraint
                    for $nametype in $ntsc <- constraint-role -> nametype-role
                    for $name in $nametype >> instances

                where
                    $ntsc / cardmax & count ($name @ [. == $nametype]) > $ntsc / cardmax
                    |
                    $ntsc / cardmin & count ($name @ [. == $nametype]) < $ntsc / cardmin
                    return $ntsc
            """

# constraint construction template
def has-name-scope($nametype, $scopetype, $cardmin, $cardmax)
   ?c isa nametypescope-constraint
         card-min: $cardmin
         card-max: $cardmax
   applies-to(?c : constraint-role, $nametype : nametype-role)
   applies-to(?c : constraint-role, $scopetype : scopetype-role)
end

########################################################################

# occurrence constraint
occurrencetypescope-constraint ako constraint
            validation-expression

            """
           // $occurrencetype >> scope >> types == $occurrencetype
           for $otsc in // occurrencetypescope-constraint
                for $occurrencetype in $otsc <- constraint-role -> occurrencetype-role
                for $occurrence in $occurrencetype >> instances

           where
                $otsc / cardmax & count ($occurrence @ [. == $occurrencetype]) > $otsc / cardmax
                |
                $otsc / cardmin & count ($occurrence @ [. == $occurrencetype]) < $otsc / cardmin
                return $otsc
            """

# constraint construction template
def has-occurrence-scope($occurrencetype, $scopetype, $cardmin, $cardmax)
```

```
  ?c isa occurrencetypescope-constraint
        card-min: $cardmin
        card-max: $cardmax
  applies-to(?c : constraint-role, $occurrencetype : occurrencetype-role)
  applies-to(?c : constraint-role, $scopetype : scopetype-role)
end

###########################################################################

associationtypescope-constraint ako constraint
                validation-expression
                """
  // $associationtype >> scope >> types == $associationtype
    for $atsc in // associationtypescope-constraint
        for $associationtype in $atsc <- constraint-role -> associationtype-role
        for $association in $associationtype >> instances

    where
        $atsc / cardmax & count ($association @ [. == $associationtype]) > $atsc / cardmax
        |
        $atsc / cardmin & count ($association @ [. == $associationtype]) < $atsc / cardmin
        return $atsc

                """

def has-association-scope($associationtype, $scopetype, $cardmin, $cardmax)
  ?c isa associationtypescope-constraint
        card-min: $cardmin
        card-max: $cardmax
  applies-to(?c : constraint-role, $associationtype : associationtype-role)
  applies-to(?c : constraint-role, $scopetype : scopetype-role)
end

###########################################################################

occurrencedatatype-constraint ako constraint
               validation-expression """

        for $odtc in // occurrencedatatype-constraint
            for $occurrencetype in $odtc <- constraint-role -> occurrencetype-role
                for $occurrence in $occurrencetype >> instances
                where
                    not (datatype($occurrence) == $odtc / datatypeuri)
                return $odtc
               """

def has-datatype($occurrencetype, $datatype)
  ?c isa occurrencedatatype-constraint
        datatype: $datatype
  applies-to(?c : constraint-role, $occurrencetype : occurrencetype-role)
end

###########################################################################

topicoccurrence-constraint ako constraint
              validation-expression
              """
    for $toc in // topicname-constraint
        for $instance in $toc <- constraint-role -> occurrencetype-role >> instances
            for $occurrencetype in $tnc <- constraint-role -> occurrencetype-role
            where
                $toc / cardmin & count($instance / $occurrencetype) < $toc / cardmin
                |
                $toc / cardmax & count($instance / $occurrencetype) > $toc / cardmax
                |
                $toc / regexp  &
```

```
                      not every $occurrence in $instance / $occurrencetype
                          satifies
                              $occurrence =~ $toc / regexp
              return $toc
                """

def has-occurrence($topictype, $occurrencetype, $cardmin, $cardmax, $expression)
  ?c isa topicoccurrence-constraint
       card-min: $cardmin
       card-max: $cardmax
       regexp: $expression

    applies-to(?c : constraint-role, $topictype : topictype-role)
  applies-to(?c : constraint-role, $occurrencetype : occurrencetype-role)
end

#################################################################################

associationrole-constraint ako constraint
                validation-expression
                 """
for $asrc in // associationrole-constraint
for $associationtype in $arsc <- constraint-role -> associationtype-role
    for $roletype in $arsc <- constraint-role -> associationroletype-role
        for $association in $associationtype >> instances
        where
             $asrc / cardmin & count ($association >> roles [. == $roletype]) < $arsc / cardmin
             $asrc / cardmax & count ($association >> roles [. == $roletype]) > $arsc / cardmax
         return $asrc
                 """

def has-role($assoctype, $roletype, $cardmin, $cardmax)
  ?c isa associationrole-constraint
       card-min: $cardmin
       card-max: $cardmax
  applies-to(?c : constraint-role, $assoctype : associationtype-role)
  applies-to(?c : constraint-role, $roletype : roletype-role)
end


#################################################################################

roleplayer-constraint ako constraint
           validation_expression
"""
    for $rpc in // roleplayercardinality-constraint
        for $associationtype in $rpc <- constraint-role -> associationtype-role
            for $roletype in $rpc <- constraint-role -> associationroletype-role
                for $topictype in $rpcc <- constraint-role -> topictype-role
                where
                    $rpc / cardmax & count ($topictype >> instances <- $roletype [^$associationtype]) > $rpcc / cardmax
                    |
                    $rpc / cardmin & count ($topictype >> instances <- $roletype [^$associationtype]) > $rpcc / cardmin
                return $rpc
"""


def plays-role($assoctype, $roletype, $topictype, $cardmin, $cardmax)
  ?c isa roleplayer-constraint
    card-min: $cardmin
    card-max: $cardmax
        applies-to(?c : constraint-role, $assoctype : assoctype-role)
        applies-to(?c : constraint-role, $roletype : roletype-role)
        applies-to(?c : constraint-role, $topictype : topictype-role)
end
```

```
##########################################################################

otherrole-constraint ako constraint
          validation-expression
                        """"
    for $associationtype in $orc <- constraint-role -> associationtype-role
        for $roletype in $orc <- constraint-role -> associationroletype-role
            for $topictype in $orc <- constraint-role -> topictype-role
                for $othertopictype in $orc <- constraint-role -> othertopictype-role
                    for $otherroletype in $orc <- constraint-role -> otherroletype-role
                        for $association in $associationtype >> instances
                        where
                            $association -> $roletype isa $topictype
                            &
                            (
                            count ($association -> $otherroletype isa $othertopictype) > $CARDMAX
                            |
                            count  ($association -> $otherroletype isa $othertopictype) < $CARDMIN
                            )
                        return false
                        """"

def other-role($roletype, $assoctype, $topictype, $otherroletype, $othertopictype, $cardmin, $cardmax)
  ?c isa otherrole-constraint
       card-min: $cardmin
       card-max: $cardmax

    applies-to(?c : constraint-role, $assoctype : assoctype-role)

  applies-to(?c : constraint-role, $roletype : roletype-role)
    applies-to(?c : constraint-role, $topictype : topictype-role)

    applies-to(?c : constraint-role, $otherroletype : otherroletype-role)
    applies-to(?c : constraint-role, $othertopictype : othertopictype-role)
end

##########################################################################

uniqueoccurrence-constraint ako constraint
              validation-expression:
            """"
    for $uoc in // uniqueoccurrence-constraint
            for $occurrencetype in $uoc <- constraint-role -> occurrencetype-role
                for $topictype in $uoc <- constraint-role -> topictype-role
                    for $occurrence in $occurrencetype
                    where
                        not (count ($occurrence \ $occurrencetype [^$topictype]) == 1)
                    return $uoc
      """"

def UniqueOccurrenceConstraint($topictype, $occurrencetype)
  ?c isa uniqueoccurrence-constraint
    applies-to(?c : constraint-role, $topictype : topictype-role)
  applies-to(?c : constraint-role, $occurrencetype : occurrencetype-role)
end

##########################################################################

def BinaryAssociation($assoctype, $r1type, $r2type, $r1playertype, $r2playertype)
 AssociationRoleConstraint($assoctype, $r1type, 1, 1)
 AssociationRoleConstraint($assoctype, $r2type, 1, 1)

 RolePlayerConstraint($assoctype, $r1type, $r1playertype)
 RolePlayerConstraint($assoctype, $r2type, $r2playertype)
end
```

```
def associated-to($r1type, $r2type, $assoctype, $r1playertype, $r2playertype)
  AssociationRoleConstraint($assoctype, $r1type, 1, 1)
  AssociationRoleConstraint($assoctype, $r2type, 1, 1)

  RolePlayerConstraint($assoctype, $r1type, $r1playertype)
  RolePlayerConstraint($assoctype, $r2type, $r2playertype)
end

################################################################################
```

Bibliography

[1] *TMCL Requirements*, ISO, 2004

[2] *TMCL Use Cases*, ISO, 2004

[3] *TMQL Requirements*, ISO, 2003