Topic Maps Constraint Specification Languages: comparing AsTMa!, OSL, and XTche

Giovani Rubert Librelotto[1] and Renato Preigschadt[1]
and José Carlos Ramalho[2] and Pedro Rangel Henriques[2]

[1] UNIFRA, Centro Universitrio Franciscano, Santa Maria, RS, 97010-032, Brasil
{librelotto,rpa.renato}@gmail.com
[2] Universidade do Minho, Departamento de Informtica
4710-057, Braga, Portugal
{jcr, prh}@di.uminho.pt

Abstract. Topic maps are an ISO standard for the representation and interchange of knowledge, with an emphasis on the findability of information. A topic map can represent information using topics (representing any concept), associations (which represent the relationships between them), and occurrences (which represent relationships between topics and information resources relevant to them). They are thus similar to semantic networks and both concept and mind maps in many respects. According to Topic Map Data Model (TMDM) [GM05], Topic Maps are abstract structures that can encode knowledge and connect this encoded knowledge to relevant information resources.
In order to cope with a broad range of scenarios, a topic is a very wide concept. On one hand, this makes Topic Maps a convenient model for knowledge representation; but on the other hand, this can also put in risk the topic map consistency. A set of semantic constraints must be imposed to the topic map in order to grant its consistency.
Currently, we can find three approaches to constrain Topic Maps – AsTMa! [Bar03a], OSL [Gar04b], and XTche [LRH05] – that allow us to specify constraints and to validate the instances of a family of topic maps against that set of rules. With these resemblances it is easy to conclude that they are quite similar. However they differ in some fundamental concepts. These three Topic Maps constraint specification languages were hardly tested and benchmarked with a huge test suite. The most significant results will be discussed in this paper.
In this article, we will use that test suite and show, step-by-step, the way we handled several kinds of Topic Maps constraints in many different instances in order to answer questions like: Do they do the same job? Are there some kind of Topic Maps constraints that are easier to specify with one of them? Do you need different background to use the tools? Is it possible to use them in similar situations (the same topic maps instances)? May we use them to produce an equal result? How do AsTMa!, OSL, and XTche relate to Topic Maps Constraint Language (TMCL)? What kind of constraints each one of these three can not specify?
We will conclude this paper with a summary of the comparisons accomplished between those Topic Maps constraint languages over the use case proposed.

1 Introduction

Semantic Web is concerned with the arrangement on the Web of information in such way that its meaning can be understood by computers as easily as by people; that is, the web pages contain not only the concrete information to be shown, but also metadata that allows for its semantic interpretation. Such an organization of information offers new perspectives for the Web [PH03]:

- Greater efficiency and precision in the search for and comprehension of information by users, humans or machines;
- Automatic treatment of information;
- Transfer of simple tasks like search, selection, updating, and transaction from the user to the system.

Organization, standardization and automatic treatment of information are the key elements that allowed the transition from the first Web generation, which is first of all a vast collection of anarchic information, to the Semantic Web, which aims at treating decentralized, sharable, and exploitable knowledge.
The Semantic Web requires the cooperation of various disciplines: Ontologies, Artificial Intelligence, Agents, Formal Logic, Languages, Graph Theory and Topology, etc. Our working area is Ontologies for the Web, more exactly, ontologies represented by Topic Maps to be handled by web applications and browsers.
An ontology is a way of describing a shared common understanding, about the kind of objects and relationships which are being talked about, so that communication can happen between people and application systems [Rat03]. In other words, it is the terminology of a domain (it defines the universe of discourse).
As a real example consider the thesaurus used to search in a set of similar, but independent, websites. Ontologies can be used to:

- Create a structured core vocabulary, to be used and validated by a set of actors in a community;
- Define and use logical relationships and rules between the concepts, allowing an efficient use of intelligent agents;
- Develop, maintain, and publish knowledge (that changes rapidly) about an organization (the whole or a part), easily providing different views.

Topic Maps [Pep00] are a good solution to organize concepts, and the relationships between those concepts, because they follow a standard notation – ISO/IEC 13250 [BBN99] – for interchangeable knowledge representation. Topic Maps are composed of topics and associations giving rise to structured semantic network that gathers information concerned with certain domain. This hierarchical topic network can represent an ontology. A topic map is an organized set of topics (formal representation of subjects), with:

- several names for each topic (or subject of the index);
- pointers (occurrences) between topics and external documents (information resources) that are indexed;

– semantic relationships, whether they are hierarchical or not, between topics via associations;

It also has the capability of supporting multi-classification (a topic can belong to more than one class), and offers a filtering mechanism based on the concept of scope that is associated with names, occurrences, and associations.
According to H. Rath [Rat03], Topic Maps are very well suited to represent ontologies. Ontologies play a key role in many real-world knowledge representation applications, and namely the development of Semantic Web. The ability of Topic Maps to link resources anywhere, and to organize these resources according to a single ontology, will make Topic Maps a key component of the new generation of Web-aware knowledge management solutions.
On one hand, this section helps to understand our interest on Topic Maps in the actually important area of Semantic Web; on the other hand, the concepts so far introduced pointed out the indubitable need for mechanisms to guarantee the semantic correctness of Topic Maps.
Besides the simplicity and powerfulness of the topic/association-based model, there are two Topic Maps features that are important in the process of understanding and reasoning about a domain: the hierarchical structure that is represented in a map (defined by the relations is-a or contains); and the complementary topic network (made up of other links that connect topics that are not included in each other but just interact).
The facts above explain the importance of Topic Maps to describe knowledge in general; in particular their application to define ontologies is one of the growing up fields. So Topic Maps are nowadays widely used within XML environments: in archives, for cataloging purposes; or in web browsers, for conceptual navigation. To build reliable systems, like those referred, it is crucial to be sure about the validation of the underlying semantic network. When developing real topic maps, it is highly convenient to use a system to validate them; this is, to verify the correctness of an actual instance against the formal specification of the respective family of topic maps (according to the intention of its creator).

2 Topic Maps Constraint Language (TMCL)

Given a specification (modelling a data structure or an operation), a constraint is a logical expression that restricts the possible values that a variable in that specification can take.
For instance, the statement the book is on the table relates two objects. To add another object also related with the table, say a knife, one can specify another statement: the knife is on the table. If it is important to note that the relative position between the book and the knife is not arbitrary, we can use a constraint to express precisely that: the knife must be on the left of the book. In this case, the constraint restricts the possible values of variable position: values like on the right or on the top are not allowed. Now, giving the table configuration, it is possible to say if it is valid or not; this is, if the given configuration satisfies the constraint or not.

Constraints can be applied to specifications in all domains. The set of valid sentences of a formal language can be restricted using contextual conditions over the grammar attributes. The proof process in logic programming can also be controlled adding constraints to the predicates. Also annotated documents can be coerced completing their type definition (DTDs or XML-Schema [DGM$^+$01]) with constraints; for this purpose there are some domain specific languages, like Schematron [Dod01] and XCSL [JLRH02].
These domain specific languages allow to describe the constraints required by each problem in a direct, clear and simple way; moreover they enable the derivation of a program to automatize the validation task. The derived semantic validator will verify every XML document, keeping silent when the constraints are satisfied, and reporting errors properly whenever the contextual conditions are broken.
The proposed Topic Maps Constraint System behaves like Schematron and XCSL. It means that the processor (generated according to a specification) checks the semantic validity of a topic map: if it is correct, the result is empty; on the other hand, every error detected is reported displaying an error message.
The language to define topic map constraints is called as Topic Map Constraint Language (TMCL). This language is currently on its way for standardization (ISO 19756 [MNB04]). The objective of TMCL is to allow formal specification of rules for topic map documents. TMCL has a similar purpose as schema languages for relational databases or XML applications. The constraint language is required to formalize application specific rules. Currently there are many different proposed constraint languages.

2.1 Case study – E-Sell Corporation

A list of requirements for the new language was recently established by the ISO Working Group – the ISO JTC1 SC34 Project for a Topic Map Constraint Language (TMCL) [NM03b]. As part of this document, there is a section that presents a case study for a language to constraint any topic map. This case study is about an e-Commerce application.
E-Commerce applications has been used widely across different businesses. Most companies are now using this application in order to do their transactions. Data of customers and products needs to be stored electronically to allow access over the internet.
In this use case, we will create a Topic Map document to store the information such about the customers and products. This may also include information on orders made by the customers.
From that we will design our vocabulary and type system (taxonomy) and formalize the design decisions we have taken. Adding application specific rules by defining full ontology in AsTMa!, XTche, and OSL, the constraining languages.
A trading company named E-Sell Corporation has many local offices, some of them large and others small. Each of this local offices maintain their own customer and order database. In order to provide unified access to all of these, E-Sell plans to use Topic Map framework. The contents of the various databases will

be mapped into Topic Map which will then be merged. Each of the local offices need to be given a template on how they should model their framework.

Product Database: The basic information of the products are the name of the product along with the product-id, the unit price, and also the quantity of products on stock. The first design decision, is to represent each product in a separate topic. Products may be categorized into different category, this categories are represented as classes and the products are instances of the class. These category classes can be: technology, beverage, etc. It is also understandable to store the name of the product as a basename.
The unit price of the product needs to be stored as well, and this can be added as inline occurrence. This information has to be represented as price or dollars. Same design for the quantity, where the quantity of the product in-stock is represented as inline occurrence as well. Every product may have a barcode that acts as a unique identifier. This can be represented as an occurrence. We may also add information on the product. Features of the product for example can be added as inline occurrence, which allow text to be added. The XTM code below shows a topic instance of technology called DVD.

```
<topic id="dvd">
  <instanceOf>
    <topicRef xlink:href="#technology"/>
  </instanceOf>
  <subjectIdentity>
    <subjectIndicatorRef xlink:href="http://products.com/dvd"/>
  </subjectIdentity>
  <baseName>
     <baseNameString>DVD</baseNameString>
  </baseName>
  <occurrence>
    <instanceOf>
      <topicRef xlink:href="#price"/>
    </instanceOf>
    <resourceData>10,00</resourceData>
  </occurrence>
  <occurrence>
    <instanceOf>
      <topicRef xlink:href="#unit"/>
    </instanceOf>
    <resourceData>30</resourceData>
  </occurrence>
  <occurrence>
    <instanceOf>
      <topicRef xlink:href="#comment"/>
    </instanceOf>
    <resourceData>A TV for all family.</resourceData>
  </occurrence>
</topic>
```

Customer Database: The basic information of the customers are the customer's name, the address, and the contact number. Customers need to have customer-ids to uniquely identify them.
The first design decision, is again to represent each customer in a separate topic. Customers can be divided into two different categories, one is an individual customer or a company customer. These categories are again modelled as classes,

and the customer is an instance of this class. It is also understandable to store the name of the customer as a basename. The address and email of each customer may be represented as inline occurrences. Customers may also have a customer-id to uniquely identify them, this can be represented as an occurrence.

```
<topic id="ronnie">
  <instanceOf>
    <topicRef xlink:href="#person"/>
  </instanceOf>
  <subjectIdentity>
    <subjectIndicatorRef xlink:href="http://alfa.di.uminho.pt/~ronnie"/>
  </subjectIdentity>
  <baseName>
    <baseNameString>Ronnie Alves</baseNameString>
  </baseName>
  <occurrence>
    <instanceOf>
      <topicRef xlink:href="#address"/>
    </instanceOf>
    <resourceData>Belm - Par</resourceData>
  </occurrence>
  <occurrence>
    <instanceOf>
      <topicRef xlink:href="#email"/>
    </instanceOf>
    <resourceRef xlink:href="mailto:ronnie@di.uminho.pt"/>
  </occurrence>
</topic>
```

If the customer is an instance of a company or a group or a firm, then additional information such as the contact person of the company is required.

```
<topic id="unifra">
  <instanceOf>
    <topicRef xlink:href="#company"/>
  </instanceOf>
  <baseName>
    <baseNameString>UNIFRA - Franciscan University Center</baseNameString>
  </baseName>
  <occurrence>
    <instanceOf>
      <topicRef xlink:href="#address"/>
    </instanceOf>
    <resourceData>Santa Maria, RS, Brazil</resourceData>
  </occurrence>
  <occurrence>
    <instanceOf>
      <topicRef xlink:href="#contact"/>
    </instanceOf>
    <resourceRef xlink:href="#ronnie"/>
  </occurrence>
  <occurrence>
    <instanceOf>
      <topicRef xlink:href="#phone_number"/>
    </instanceOf>
    <resourceData>+(351)1234-1234</resourceData>
  </occurrence>
  <occurrence>
    <instanceOf>
      <topicRef xlink:href="#fax"/>
    </instanceOf>
    <resourceData>+(351)-1234-1235</resourceData>
  </occurrence>
</topic>
```

Order Database: When a particular customer request for products to order, then this is represented as an order. Each order is represented as a topic. The order will contain information such as the order-id, the order-date, shipping-date, and the total price of the order. The order-id has to be unique because it then identifies each particular order. Order-date, shipping-date, and the total price are represented as inline occurrences. It does make sense to have the basename containing the order-id as well. In the next XTM, the order-id is represented as an occurrence.

```
<topic id="order05">
  <instanceOf>
    <topicRef xlink:href="#order"/>
  </instanceOf>
  <subjectIdentity>
    <subjectIndicatorRef xlink:href="http://order.com/05"/>
  </subjectIdentity>
  <baseName>
    <baseNameString>Order 05</baseNameString>
  </baseName>
  <occurrence>
    <resourceRef xlink:href="http://order.com/05"/>
  </occurrence>
  <occurrence>
    <instanceOf>
      <topicRef xlink:href="#date"/>
    </instanceOf>
    <resourceData>20/10/2006</resourceData>
  </occurrence>
  <occurrence>
    <instanceOf>
      <topicRef xlink:href="#ship-date"/>
    </instanceOf>
    <resourceData>24/10/2006</resourceData>
  </occurrence>
  <occurrence>
    <instanceOf>
      <topicRef xlink:href="#price"/>
    </instanceOf>
    <resourceData>90,00</resourceData>
  </occurrence>
</topic>
```

For each of this order, there are product lines. The product lines will contain the information or details on the order. This is represented as a relationship having the product-id and the number of quantity ordered, along with the order-id to identify which order does the order-line belongs to.

```
<association>
  <instanceOf>
    <topicRef xlink:href="#contains"/>
  </instanceOf>
  <member>
    <roleSpec>
      <topicRef xlink:href="#order"/>
    </roleSpec>
    <topicRef xlink:href="#order05"/>
  </member>
  <member>
    <roleSpec>
      <topicRef xlink:href="#product"/>
    </roleSpec>
    <topicRef xlink:href="#dvd"/>
  </member>
  <member>
    <roleSpec>
      <topicRef xlink:href="#quantity"/>
    </roleSpec>
    <topicRef xlink:href="#unit_5"/>
  </member>
</association>
```

The quantity is represented as a part of the relationship or association. Therefore the topic with the number of units should be available. This is just a temporary topic that is created, does not have to stay statically in the Topic Map framework. The order and the customer who ordered it are represented in a separate relationship. This association has at most one player of playing the role customer, and another player playing the role order. And since one customer can have more than one orders, this means that in the association, the role order can be played by more than one topic players. When the role order is played by more than one topic, this means that the customer has more than one orders.

```
<association>
  <instanceOf>
    <topicRef xlink:href="#is-making-order"/>
  </instanceOf>
  <member>
    <roleSpec>
      <topicRef xlink:href="#order"/>
    </roleSpec>
    <topicRef xlink:href="#order05"/>
  </member>
  <member>
    <roleSpec>
       <topicRef xlink:href="#customer"/>
    </roleSpec>
    <topicRef xlink:href="#unifra"/>
  </member>
</association>
```

The E-Sell's Ontology: The objective of the ontology is to define set of vocabularies along with its meaning that will be used within the framework. Rules or constraints also need to be defined to ensure the rigidity of the Topic Map framework that is used. This ensures that the information contained within the document are valid.
From the product class we derive subclasses which are the categories of products:

– beverage subclasses product
– technology subclasses product
– clothing subclasses product

In this case study, it is created some product subsubclasses. For instance:

– wine subclasses beverage
– radio subclasses technology
– television subclasses technology
– DVD subclasses technology

– phone subclasses technology

Another topic that needs to be covered as a class is the customer. From this class we derive subclasses which are the different customer categories:

– person subclasses customer
– company subclasses customer

This two customer subclasses has some own subclasses:

– Gustavo Arnold subclasses person
– Ronnie Alves subclasses person
– Giovani Librelotto subclasses person
– Marco Antonio subclasses person
– UNIFRA subclasses company

Talking about order subclasses, it is defined five instances which are associated with products and customer instances:

– Order 01 (subclasses order) was accomplished by Giovani Librelotto and it is composed of television and DVD;
– Order 02 subclasses order was accomplished by Gustavo Arnold and it is composed of phone and radio;
– Order 03 subclasses order was accomplished by Marco Antonio and it is composed of radio and television;
– Order 04 subclasses order was accomplished by Ronnie Alves and it is composed of DVD and television;
– Order 05 subclasses order was accomplished by UNIFRA and it is composed of DVD, radio, and television;

Figure 1 shows a graph that represents part of E-Sell's ontology on Vizigator [Gen04]. In this figure is presented the main topic types (order, product, and customer), the others topic types (person, company, technology, and beverage), and the topic instances (order 01, radio, Ronnie Alves, ...).
The links in that figure represents the relationship between topic type and topic instances (beverage and wine, for instance) or association between two (or more) topic of different types (for instance, order 05 is composed of DVD, radio, television, and wine).

3 XTche language

This section introduces a process for specifying constraints on topic maps with a constraint language called XTche, which is based on XML Schema syntax. This language allows to express contextual conditions on classes of Topic Maps. With XTche, a topic map designer defines a set of restrictions that enables to verify if a particular topic map is semantically valid.

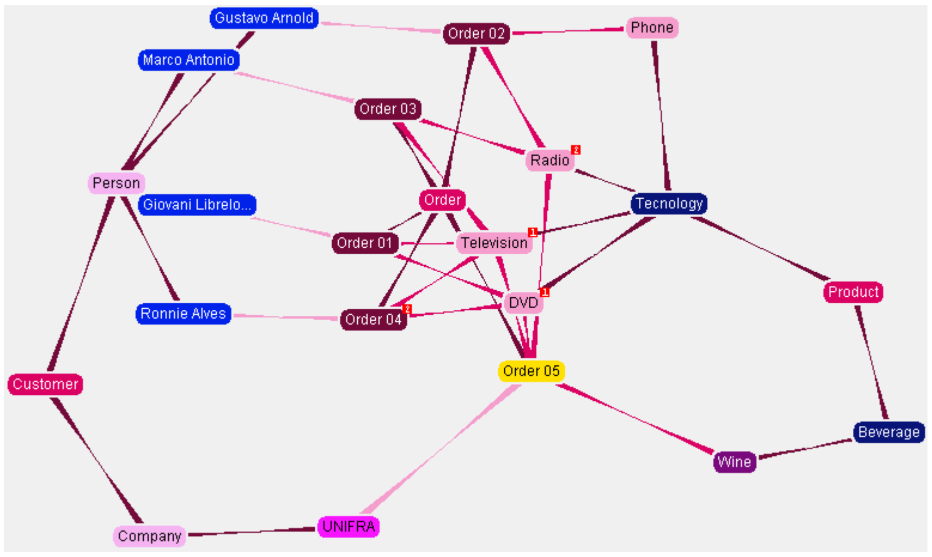

Fig. 1. The E-Sell Corporation's ontology

XTche is an XML Schema oriented language. This idea brings two benefits: on one hand it allows for the syntactic specification of Topic Maps (not only the constraints); and on the other hand it enables the use of an XML Schema editor (for instance, XMLSpy[3]) to provide a graphical interface and the basic syntactic checker [LRH05].

The constraining process is composed of a language and a processor. The language is based on XML Schema [DGM+01,HM01,Fal01] syntax. The processor is developed in XSLT language [Tid01,Tho03]. The XTche processor takes a XTche specification and it generates a particular XSLT stylesheet. This stylesheet can validate a specific topic map (or a set of them) according to the constraints in the XTche specification.

As the manual checking of large topic maps (frequent in real cases) is impossible, it is mandatory to provide an automatic validator.

XTche language meets all the TMCL requirements [NMB04]; for that purpose, XTche has a set of constructors to describe constraints in Topic Maps. But the novelty of the proposal is that the language also permits the definition of the topic map structure in an XML Schema style. An XTche specification merges the schema (defining the structure and the basic semantics) with constraints (describing the contextual semantics) for all the topic maps in that family.

[3] http://www.altova.com

3.1 XTche syntax

XTche uses XML Schema instead of DTD to improve the validation process because some semantic requirements (domain, occurrence number, etc.) can be added to the structural specification. Still XML parsers will deal with that task. However main topic maps semantic requirements remain unspecified. So, a specification language that allows us to define the schema and constraints of a family of Topic Maps is necessary.
Like XTM, XTche specifications can be too verbose; this way a graphical tool to support XTche specifications authoring is desirable. To overcome this problem, XTche syntax follows the XML Schema syntax; so, any XTche constraint specification can be written in a diagrammatic style with a common XML Schema editor. It is up to the designer to decide how to edit the constraints and schemas: either in a XML Schema visual editor (that outputs the respective textual description), or in an XML text file according to XTche schema.
The XTche specification (in textual format) is taken as input by XTche Processor that analyzes and checks it, and generates a Topic Map validator (TM-Validator) as output.
XTche also takes advantage of XML Schema data types to validate some constraints [LRH05].
An XTche specification is a schema where the <xtche> element is the root. This element is composed of a sequence, where three elements are allowed: <schema-constraints>; <contextual-constraints>, and <existence-constraints> as shows the Figure 2. Both these subelements are optional; it means that a specification can only have one kind of constraints. These subelements are composed of a sequence, where each subelement represents a particular constraint.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema elementFormDefault="qualified" attributeFormDefault="unqualified"
xmlns:xtche="http://www.di.uminho.pt/~gepl/xtche" xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:import namespace="http://www.di.uminho.pt/~gepl/xtche"
    schemaLocation="http://www.di.uminho.pt/~gepl/xtche/xtche-schema.xsd"/>
    <xs:element name="xtche">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="schema-constraints" minOccurs="0">
                    <xs:complexType>
                        <xs:sequence>
                            <!-- schema constraint 1 -->
                            ...
                            <!-- schema constraint N -->
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
                <xs:element name="contextual-constraints" minOccurs="0">
                    <xs:complexType>
                        <xs:sequence>
                            <!-- contextual constraint 1 -->
                            ...
                            <!-- contextual constraint N -->
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
                <xs:element name="existence-constraints" minOccurs="0">
                    <xs:complexType>
                        <xs:sequence>
```

```
                            <!-- contextual constraint 1 -->
                            ...
                            <!-- contextual constraint N -->
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

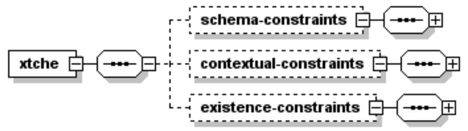

Fig. 2. XTche Skeleton

XTche Skeleton above describes the structure for all the XTche specifications. This skeleton is a generic, but incomplete, XML Schema that must be fulfilled with particular constraints for each case. To write those constraints, the basic schema language is extended by a set of domain specific attributes that are defined in a separated file imported by the skeleton [LRH05].

Schema Constraints: For instance, to specify that a person must have at least one basename, the XTche specification is (Figure 3:

```
<xs:element name="schema-constraints">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="customer">
        <xs:complexType>
          <xs:sequence>
            <xs:element ref="xtche:baseName"/>
            <xs:any namespace="##any"/>
          </xs:sequence>
          <xs:attribute ref="xtche:topicType"/>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

In a schema constraint, <xs:attribute name="xtche:topicType"/> represents a topic type. The element <xs:any namespace="##any"/> allows any other topic characteristics to this topic (it is similar to the brackets "[" and "]" in the AsTMa! language).

Contextual Constraints: To define a contextual constraint, we can explain by this example (Figure 4:

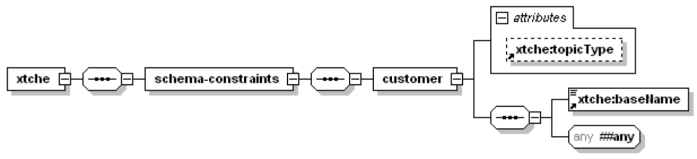

Fig. 3. An XTche schema constraint

```
<xs:element name="contextual-constraints">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="wine">
        <xs:complexType>
          <xs:attribute ref="xtche:topicType-Exclusive"/>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

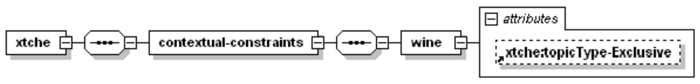

Fig. 4. An XTche contextual constraint

In XTche contextual constraints, the <xs:attribute name="xtche:topicType-Exclusive"/> means that this topic can only be used as topic types.

Existence Constraints: In terms of existence constraints, a topic map is valid if at least a topic or association is in agreement with each constraint of this kind. For instance, the constraint below makes necessary a topic map contain an association instance of is-making-order that has the topic unifra playing the role customer (Figure 5.

```
<xs:element name="existence-constraints">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="is-making-order">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="customer">
              <xs:complexType>
                <xs:sequence>
                  <xs:element name="unifra">
                    <xs:complexType>
                      <xs:attribute ref="xtche:topic"/>
                    </xs:complexType>
                  </xs:element>
                </xs:sequence>
                <xs:attribute ref="xtche:associationRole"/>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
          <xs:attribute ref="xtche:associationType"/>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

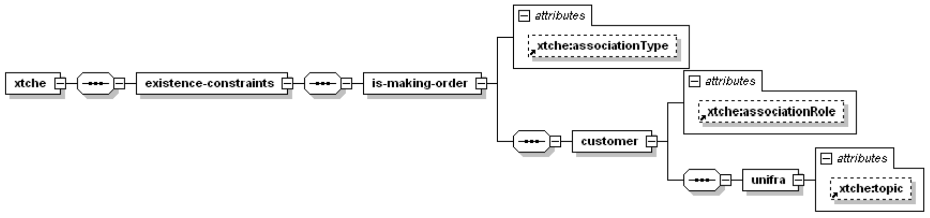

Fig. 5. An XTche existence constraint

4 AsTMa! language

This section introduces another language for constraint topic maps, called AsTMa! defined by Robert Barta [Bar03a], with a proposed objective of validating topic maps against a given set of rules. AsTMa! is a member of AsTMa* family [Bar03b] (which includes AsTMa= for authoring TM, and AsTMa? for querying TM) and exposes some features of TMCL [MNB04].
AsTMa! meets only some of the requirements of the TMCL, because it has written early than the final version of the TMCL.
As the XTche the constraining process is composed of a language and a processor. The language is based on a Perl language, and the processor is written in Perl. At this time the AsTMa! Language is no longer maintained, because the author is working on a completely new distribution [Bar05]. So for this article we assume the AsTMa! Language definition for evaluate expressions.

4.1 AsTMa! Syntax

AsTMa! syntax is developed to be written by human on text mode, not demanding an graphical tool, but it need a knowledge of a new language syntax.
The AsTMa! Syntax allows logical operators, like AND, OR, and regular expressions. Above we show some of the particularities of the AsTMa! syntax.

Brackets "[" and "]": Defines the scope of a constraining, when is used in pairs of "[" expression "]" tells that a topic map has to be at least the expression, and when is used in pairs of "]" expression "[" tells that a topic map has to be only the expression. Example:

```
forall [ * (person)
          bn: $name ]
=> exists ] (is-making-order) constraint
     customer: $name
     order: * [
```

This example means that every person must have at least one basename, and must have an association of type is-making-order with only an person (represented by the variable $name) playing the role of an customer and an order with any content.

Wildcards and regular expression: Allows the use of wildcards and regular expressions allowing constraints more complex, limiting some fields of the topic map. The rule for write regular expressions is similar to language Perl, that is starting and finish the regular expression which an slash (/). The example below shows the use of wildcards on the id of the topic person, and force the order to have the string "order" (the /i tells us who "order" is case insensitive) in an association instance of is-making-order.

```
forall [ * (person)
          bn: $name ]
=> exists ] (is-making-order) constraint
     customer: $name
     order: /order/i [
```

Conditional Constraints: Allows constraints on topics with must play a role on some other topic or association, that is constraint dependencies on topics and associations. In a set with wildcards and regular expressions makes an powerful mechanism to constraint complex relationships in an topic map. The next example is showing the conditional constraints because for each order must have an association contains with an order of same name as the basename of order.

```
forall [ * (order)
          bn: $name ]
=> exists [ (contains) constraint
          order: $name
          product: *
]
```

Boolean Constraints: Adds logical operators on AsTMa! language for constraints topic maps, allowing for example the deny all the rule, or some parts of it. In the next example the verification is for each order must have an association of is-making-order and an association of type contains with the player order including the basename of topic order.

```
forall [ * (order)
          bn: $name          ]
=> exists [ (is-making-order)
          order: $name ]
and exists [ (contains)
          order: $name ]
```

5 OSL language

According to the Ontopia Schema Language specification [Gar04a], OSL has been designed to have a minimal number of features and a minimum of expressive power. So the language specifies only a minimal set of features that will be available on TMCL. Basically the OSL language constraints only the structure of a Topic Map.
A topic map schema in the Ontopia Schema Language consists of a set of topic and association class definitions. These class definitions constrain the structure of the instances of the classes, and so control the form information may take in a topic map that uses the schema [Gar04b].
As the languages above the constraints process is composed by an language and a processor. The language is based on XTM [PH03] and the processor is written in Java, available for running standalone and with a plug-in of Ontopia Omnigator [Ont02].

5.1 OSL characteristics

OSL syntax is similar to XTM language, having some elements and functionality exactly as the XTM, like <association>, <instanceOf>, <occurrence>, <baseName>, <scope>, etc.
The language have the ability to determining who topic has an capacity of play some roles in associations; there is an example below:

```
<topic>
    <instanceOf>
      <internalTopicRef href="#order"/>
    </instanceOf>
    <baseName> min="1" max="1">
      <scope></scope>
    </baseName>
    <occurrence min="1" max="1">
      <instanceOf>
         <internalTopicRef href="#date"/>
      </instanceOf>
    </occurrence>
    <occurrence min="1" max="1">
      <instanceOf>
         <internalTopicRef href="#ship-date"/>
      </instanceOf>
    </occurrence>
    <occurrence min="1" max="1">
      <instanceOf>
         <internalTopicRef href="#price"/>
      </instanceOf>
    </occurrence>
    <playing>
      <instanceOf>
        <internalTopicRef href="#order"/>
      </instanceOf>
      <in>
        <instanceOf>
          <internalTopicRef href="#is-making-order"/>
        </instanceOf>
      </in>
    </playing>
    <playing>
```

```
      <instanceOf>
        <internalTopicRef href="#order"/>
      </instanceOf>
      <in>
        <instanceOf>
       <internalTopicRef href="#contains"/>
        </instanceOf>
      </in>
    </playing>
</topic>
```

In example above, the topic of type order may play a role of order in a contains and is-making-order association. That topic map also is an instance of order topic, have an one basename without scope, have three types of occurrences, be respectively date, ship-date and price (at least and at most one occurrence of them).

6 Comparing the Topic Maps Constraint Languages

In this section we will compare the three languages viewed previously in this paper. So on we discuss some advantages of one or other particularities of the constraints languages.

6.1 Do you need different background to use the languages?

Yes.
To use the XTche language, the topic map designer needs to have solid knowledge about XML, XML Schema, XSLT, and XPath [CD99]. All XTche specifications are in XML Schema format, so the designer requires to use a visual tool to write the constraints. The constraint can be written in any text editor, of course, but the its complexity is almost the same that to write a common XML Schema.
XTche processor is written in XSLT language. So, if the designer wants to modify anything in the processor, he needs to programm this XSLT code, that has several XPath instructions.
To specify AsTMa! constraints, the same designer requires to know the AsTMa! particular syntactic. To run the AsTMa! processor is necessary to have installed the Perl and Prolog compilers in his machine.
In OSL case, the language is XTM-based, so the designer needs specifies this kind of constraints in agreement with XTM elements. The OSL tool is standalone and requires Java language. Another way to execute OSL verifications is running it on Omnigator [Ont02].

6.2 Do they do the same job?

Not really. To illustrate this subject, we will present a few comparisons among these languages.

Validating generic topic map structure: In the first example, XTche, OSL, and AsTMa! languages virtually do the same job. These three languages allow to verify if a topic map (or a family of topic maps) has some inconsistency in agreement with a set of rules that verifies about its structure.
For instance, the association is-making-order represent each product line. This creates a relationship between a particular product and an order, along with the quantity of the product ordered. In the topic map language, it means that an association of type is-making-order must have three association roles: product, order, and quantity. The code below shows the AsTMa! specification:

```
forall $a [ (is-making-order) ]
=> exists $a ] (is-making-order)
         product: *
         quantity: *
         order: * [
```

XTche language defines this constraint like this (Figure 6):

```
  <xs:element name="schema-constraints">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="is-making-order">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="product">
                <xs:complexType>
                  <xs:attribute ref="xtche:associationRole"/>
                </xs:complexType>
              </xs:element>
              <xs:element name="quantity">
                <xs:complexType>
                  <xs:attribute ref="xtche:associationRole"/>
                </xs:complexType>
              </xs:element>
              <xs:element name="order">
                <xs:complexType>
                  <xs:attribute ref="xtche:associationRole"/>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
            <xs:attribute ref="xtche:associationType"/>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
```

In OSL language, the associated specification is:

```
<association>
    <instanceOf>
      <internalTopicRef href="#is-making-order"/>
    </instanceOf>
    <role min="1" max="1">
      <instanceOf>
          <internalTopicRef href="#product"/>
      </instanceOf>
      <player>
          <internalTopicRef href="#product"/>
      </player>
    </role>
    <role min="1" max="1">
```

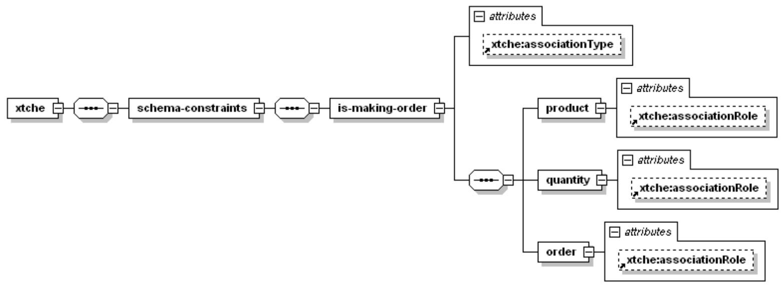

Fig. 6. XTche specification for an association structure

```
      <instanceOf>
          <internalTopicRef href="#order"/>
      </instanceOf>
      <player>
          <internalTopicRef href="#order"/>
      </player>
    </role>
    <role min="1" max="1">
      <instanceOf>
          <internalTopicRef href="#quantity"/>
      </instanceOf>
      <player>
          <any/>
      </player>
    </role>
</association>
```

Validating specific topic map structure: In the second example, if the constraint is also about the association is-making-order where we need to be sure of a topic instance of product plays the role product and a topic instance of order plays the role order. The code below introduces the AsTMa! function called *exists*:

```
 forall $a [ (is-making-order) ]
 => exists $a ] (is-making-order)
         product: $p
         quantity: *
         order: $o [
    and
    exists [ $p (product) ]
    and
    exists [ $o (order) ]
```

In XTche, the relative specification is shown in the Figure 7 and presented below:

```
  <xs:element name="schema-constraints">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="is-making-order">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="product">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="product">
                      <xs:complexType>
                        <xs:attribute ref="xtche:associationPlayer"/>
                        <xs:attribute ref="xtche:topicType"/>
                      </xs:complexType>
                    </xs:element>
                  </xs:sequence>
                  <xs:attribute ref="xtche:associationRole"/>
                </xs:complexType>
              </xs:element>
              <xs:element name="quantity">
                <xs:complexType>
                  <xs:attribute ref="xtche:associationRole"/>
                </xs:complexType>
              </xs:element>
              <xs:element name="order">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="order">
                      <xs:complexType>
                        <xs:attribute ref="xtche:associationPlayer"/>
                        <xs:attribute ref="xtche:topicType"/>
                      </xs:complexType>
                    </xs:element>
                  </xs:sequence>
                  <xs:attribute ref="xtche:associationRole"/>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
            <xs:attribute ref="xtche:associationType"/>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
```

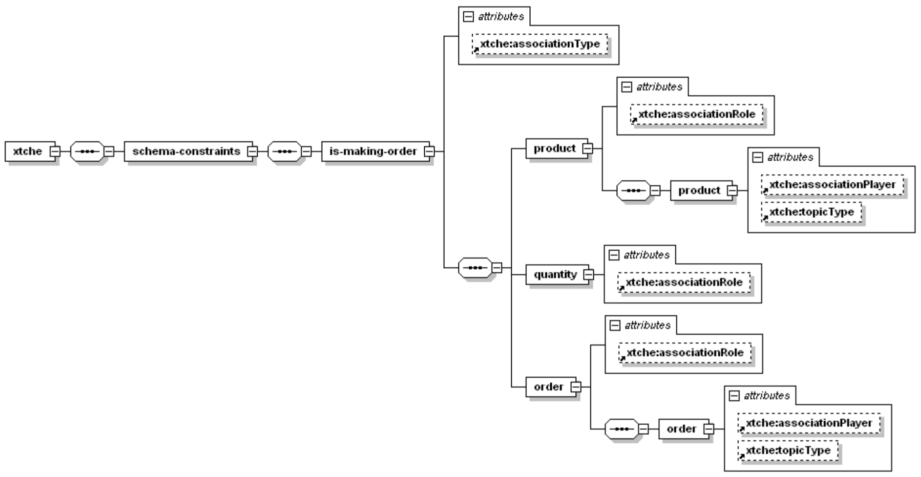

Fig. 7. An XTche specification to an association type with specific roles and players

Unfortunately, OSL language does not allow to specify this kind of constraint.

Data types: According TMDM [GM05], Topic Maps do have a concept of data and data types, but there is no commitment to any set of primitives such as XML Schema (XSD) [DGM+01]. That may be a good move, since XSD is – like any other set – quite arbitrary. Useful, but arbitrary. So, if a topic map designer wants to validate an age occurrence as a number, it is necessary to use a constraint language.
The only way to constrain text in AsTMa! was by using regular expressions. For instance, to allow to invoke "boolean test functions", such as:

```
in (age): ?is_age()
```

The AsTMa! validator would call this function (implemented externally). According your creator, if AsTMa! would have to be recreated, this issue would have to be addressed.
XTche takes advantage of XML Schema data types to validate some constraints. The code below (represented by Figure 8) shows the specification for person topic type that has an age occurrence of type integer. Any XSD data type can be used in XTche specification.

```
<xs:element name="schema-constraints">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="person">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="age">
              <xs:complexType>
                <xs:simpleContent>
                  <xs:extension base="xs:integer">
                    <xs:attribute ref="xtche:occurrenceType"/>
                  </xs:extension>
                </xs:simpleContent>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
          <xs:attribute ref="xtche:topicType"/>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

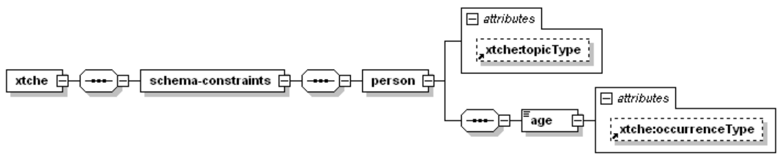

Fig. 8. Data type in XTche language

In the Figure 8, it is not possible to see the data type specification. But the age element has three horizontal lines that means this element has a type. In this case, <xs:extension base="xs:integer"> is the type of that element.

In terms of data type, OSL does not provide any kind of data type.

6.3 Are there some kind of Topic Maps constraints that are easier to specify with one of them?

The topic maps constraints about topics and association structures are easier to specify in both languages. For instance, the constraint “customer must have a contact number either a phone or fax number” is specified in AsTMa! like this:

```
forall $c [ * (customer) ]
=> exists $c [ in (phone): * ]
    or
   exists $c [ in (fax): * ]
```

According XTche language, the respective specification is below and a diagrammatic view of this specification is in Figure 9.

```
<xs:element name="schema-constraints">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="customer">
        <xs:complexType>
          <xs:choice>
            <xs:element name="phone">
              <xs:complexType>
                <xs:attribute ref="xtche:occurrenceType"/>
              </xs:complexType>
            </xs:element>
            <xs:element name="fax">
              <xs:complexType>
                <xs:attribute ref="xtche:occurrenceType"/>
              </xs:complexType>
            </xs:element>
          </xs:choice>
          <xs:attribute ref="xtche:topicType"/>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

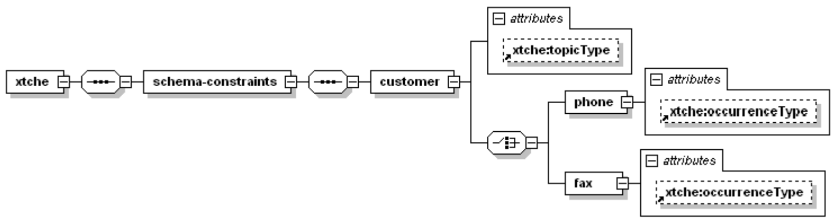

Fig. 9. An XTche topic type structure

In other hand, OSL correspondent code is presented below. However, this language have a limitation: it does not work with boolean operations. So the constraint “either a phone or fax number” is not supported.

```
<topic>
    <instanceOf>
      <internalTopicRef href="#customer"/>
    </instanceOf>
    <occurrence min="0" max="1">
      <instanceOf>
        <internalTopicRef href="#phone"/>
      </instanceOf>
    </occurrence>
    <occurrence min="0" max="1">
      <instanceOf>
        <internalTopicRef href="#fax"/>
      </instanceOf>
    </occurrence>
</topic>
```

The code above defines a topic instance of customer that has zero or one phone occurrence and zero or one fax occurrence. But, according this OSL specification, there is no way to impede that a topic instance of customer has both occurrences.

6.4 Is it possible to use them in similar situations (the same topic maps instances)?

It is possible to use them in several similar situations but is important to care about the topic map format. XTche language only process topic maps in XTM format [PM01]. There is a small project in the XTche context to create a processor that converts other topic maps formats – LTM (Linear Topic Map) [Gar02] and HyTM (HyTime Topic Maps) [NBB03] – to XTM.
In the same way, AsTMa! language processes topic maps according to the AsTMa= format.
Talking about OSL, this language is part of Omnigator tool [Ont02]. So, almost all kind of topic map format can be validated according a set of OSL rules. Ontopia enable the navigation over the following topic map formats: XTM, RDF (Resource Description Framework) [LS99], LTM, and HyTM.

6.5 May we use them to produce an equal result?

Maybe.
The answer is *Yes* if the topic map designer wants to validate the topic map schema because these three languages confirm the validity of a topic map instance across a set of rules. The answer is *No* if the topic maps designer wants to validate the topic map with particular constraints, like existence, boolean, and conditional constraints. In this case, XTche and AsTMa! has constructors to specify that; OSL has not.
For example, the constraint "for all topic that has the topic type customer, it must have a basename (for customer name), an occurrence (for address), an subject identifier (for customer id), and optionally additional occurrence (for email address)" [NMB04] can be constrained in XTche, AsTMa!, and OSL. The result for both languages is a list of the topics that are not conformed with this rule. If all the topics conforms this rule, the result is the topic map validation confirmed. So, for this case: *Yes*, we may use them to produce an equal result.

However another constraint example: “for all association with association type is-making-order, it must have the association roles customer and order played by the topic that is of type customer and order respectively” [NMB04] can be validated by AsTMa! and XTche languages, and can not be validated by OSL language. Thus, for this case: *No*, we may not use them to produce an equal result.

6.6 How do AsTMa!, OSL, and XTche relate to Topic Maps Constraint Language (TMCL)?

According to the ISO, “the Topic Map Constraint Language (TMCL) is a formal language for defining schemas and constraints on topic map models. Specifically, TMCL is to constrain topic map models as defined by the Data Model for Topic Maps. The constraint language will provide a formal constraint language, related operational semantics, and a syntax” [NM03a].
XTche and AsTMa! languages are based on a draft version of the TMCL, so they are able to specify any kind of constraint suggested by TMCL requirement. However OSL was not designed on the basis of TMCL requirements; it is intended only for validating the topic maps structures.

6.7 What kind of constraints each one of these three languages can not specify?

AsTMA! and XTche specifies a full draft version of TMCL, and have constructors to make complex conditional, boolean and existential constraints. In other way, OSL does not have relationship with TMCL, and it was defined to make just simple validations in a topic map. So the language does not have boolean, existential, and conditional operators, becoming an real alternative only in simple and small projects.
For instance, OSL can not specify the following constraint: “for all association with association type is-making-order, it must have the association roles customer and order played by the topic that is of type customer and order respectively”.

7 Conclusion

Topic Maps are a standard for organizing and representing knowledge about a specific domain. They allow us to specify subjects and relationships between subjects. Steve Pepper [Pep00] defines subject as the term used for the real world thing that the topic itself stands in for. A topic, in its most generic sense, can be anything whatsoever - a person/object, an entity/organization, a concept – regardless of whether it actually exists or is a mental abstraction [Rat03].
This paper showed a comparison among the three TMCL-based languages – AsTMa!, OSL, and XTche – over several kinds of Topic Maps constraints in many different instances. We started with our strong motivation to check a

topic map for syntactic and semantic correctness - as a notation to describe an ontology that supports a sophisticated computer system (like the applications in the area of Semantic Web or archiving) its validation is crucial!
Then we assumed XTM and TMCL as starting points and we used our background in compilers and XML validation to come up with our proposal. The main conclusion is that XTche and AsTMa! comply with all requirements stated for TMCL whereas OSL just includes topic maps structure validation.
We succeeded in applying this approach into a case study – E-Commerce Application (subsection 6.1 of TMCL Requirements [NM03b]) – virtually representative of all possible cases. It means that: on one hand, we were able to describe the constraints required by each problem in a direct, clear and simple way; on the other hand, the Topic Maps semantic validator could process every document successfully, that is, keeping silent when the constraints are satisfied, and detecting/reporting errors, whenever the contextual conditions are broken.
Doing a comparison among these languages, some advantages of XTche emerge:

1. XTche has a XML Schema-based language, a well-known format;
2. XTche allows the use of an XML Schema graphic editor, like XMLSpy. In a diagrammatic view, it is easy to check visually the correctness of the specification.
3. XTche gathers in one specification both the structure and the semantic descriptions, and it realizes a fully declarative approach requiring no procedural knowledge for users.

The main problem about XTche is the size of this code. If a topic map designer does not have a XML Schema editor, the specification is too complex comparing with the other languages. This XTche problem is a AsTMa! advantage: the size of AsTMa! constraints is small, very similar to the regular expressions.
In a related work, Eric Freese [Fre02] says that it should be possible to use the DAML+OIL language to provide a constraint and validation mechanism for topic map information. The cited paper discusses how to describe validation and consistency of the information contained in Topic Maps using DAML+OIL and RDF, showing how to extend XTM and how to define PSIs and class hierarchies, as well as to assign properties to topics.
Talking about the constraints covered by these languages, XTche and AsTMa! have more mechanisms to check the validity of Topic Maps than the OSL language and Eric Freese proposal.

References

[Bar03a] Robert Barta. AsTMa! Bond University, TR., 2003. `http://astma.it.bond.edu.au/constraining.xsp`.

[Bar03b] Robert Barta. AsTMa* Language Family. Bond University, TR., 2003. `http://astma.it.bond.edu.au/astma-family.dbk`.

[Bar05] Robert Barta. CPanel Readme. `http://search.cpan.org/src/DRRHO/XTM-0.37/README`, 2005.

[BBN99] Michel Biezunsky, Martin Bryan, and Steve Newcomb. ISO/IEC 13250 - Topic Maps. ISO/IEC JTC 1/SC34, December 1999. `http://www.y12.doe.gov/sgml/sc34/document/0129.pdf`.

[CD99] James Clark and Steve DeRose. XML Path Language (XPath) - Version 1.0. `http://www.w3.org/TR/xpath`, November 1999.

[DGM+01] Jon Duckett, Oliver Griffin, Stephen Mohr, Francis Norton, Nikola Ozu, Ian Stokes-Rees, Jeni Tennison, Kevin Williams, and Kurt Cagle. *Professional XML Schemas*. Wrox Press, 2001.

[Dod01] Leigh Dodds. Schematron: Validating XML Using XSLT. In *XSLT UK Conference*. Keble College, Oxford, England, 2001.

[Fal01] David C. Fallside. XML Schema part 0: Primer. `http://www.w3.org/TR/xmlschema-0`, May 2001.

[Fre02] Eric Freese. Using DAML+OIL as a Constraint Language for Topic Maps. In *XML Conference and Exposition 2002*. IDEAlliance, 2002. `http://www.idealliance.org/papers/xml02/dx_xml02/papers/05-03-03/05-03-03.html`.

[Gar02] Lars Marius Garshol. LTM – The Linear Topic Map Notation. Ontopia, 2002. `http://www.ontopia.net/topicmaps/ltm.html`.

[Gar04a] Lars Marius Garshol. The Ontopia Schema Language – Reference Specification. `http://www.ontopia.net/omnigator/docs/schema/spec.html`, 2004.

[Gar04b] Lars Marius Garshol. The Ontopia Schema Language – Tutorial. `http://www.ontopia.net/omnigator/docs/schema/tutorial.html`, 2004.

[Gen04] Pamela Gennusa. Ontopia's Vizigator(tm) - Now you see it! In *XML 2004 Conference and Exposition*, Washington D.C., U.S.A, 2004. IDEAlliance. `http://www.idealliance.org/proceedings/xml04/papers/311/311.html`.

[GM05] Lars Marius Garshol and Graham Moore. Topic Maps – Data Model. In *ISO/IEC JTC 1/SC34*. `http://www.isotopicmaps.org/sam/sam-model/`, January 2005.

[HM01] Elliote Rusty Harold and W. Scott Means. *XML in a Nutshell*. O'Reilly & Associates, 2001.

[JLRH02] Marta H. Jacinto, Giovani R. Librelotto, Jos C. Ramalho, and Pedro R. Henriques. Constraint Specification Languages: comparing XCSL, Schematron and XML-Schemas. In *XML Europe 2002*, Barcelona, Spain, 2002.

[LRH05] Giovani Rubert Librelotto, Jos Carlos Ramalho, and Pedro Rangel Henriques. Constraining topic maps: a TMCL declarative implementation. In *Extreme Markup Languages 2005: Proceedings*. IDEAlliance, 2005. `http://www.mulberrytech.com/Extreme/Proceedings/html/2005/Ramalho01/EML2005Ramalho01.html`.

[LS99] Ora Lassila and Ralph R. Swick. Resource Description Framework (RDF) Model and Syntax Specification. World Wide Web Consortium, February 1999. `http://www.w3.org/TR/REC-rdf-syntax`.

[MNB04] Graham Moore, Mary Nishikawa, and Dmitry Bogachev. Topic Map Constraint Language. ISO/IEC JTC 1/SC 34, 2004. `http://www.jtc1sc34.org/repository/0549.htm`.

[NBB03] Steven R. Newcomb, Michel Biezunski, and Martin Bryan. The HyTime Topic Maps (HyTM) Syntax 1.0. ISO/IEC JTC 1/SC34 N0391, 2003. `http://www.jtc1sc34.org/repository/0391.htm`.

[NM03a] Mary Nishikawa and Graham Moore. Topic Map Constraint Language (TMCL). ISO/IEC, 2003. `http://www.isotopicmaps.org/tmcl/`.

[NM03b] Mary Nishikawa and Graham Moore. Topic Map Constraint Language (TMCL) Requirements and Use Cases. ISO/IEC JTC 1/SC34 N0405rev, 2003. `http://www.isotopicmaps.org/tmcl/requirements.html`.

[NMB04] Mary Nishikawa, Graham Moore, and Dmitry Bogachev. Topic Map Constraint Language (TMCL) Requirements and Use Cases. ISO/IEC JTC 1/SC34 N0548, 2004. `http://www.jtc1sc34.org/repository/0548.htm`.

[Ont02] Ontopia. The Ontopia Omnigator, 2002. `http://www.ontopia.net/omnigator/`.

[Pep00] Steve Pepper. The TAO of Topic Maps - finding the way in the age of infoglut. Ontopia, 2000. `http://www.ontopia.net/topicmaps/materials/tao.html`.

[PH03] Jack Park and Sam Hunting. *XML Topic Maps: Creating and Using Topic Maps for the Web*, volume ISBN 0-201-74960-2. Addison Wesley, 2003.

[PM01] Steve Pepper and Graham Moore. XML Topic Maps (XTM) 1.0 - Annex D: XTM 1.0 Document Type Declaration (Normative). TopicMaps.Org Specification, August 2001. `http://www.topicmaps.org/xtm/1.0/#dtd`.

[Rat03] H. Holger Rath. White Paper: The Topic Maps Handbook. Empolis, 2003.

[Tho03] Henry Thompson. The Extensible Stylesheet Language (XSL) Family. `http://www.w3.org/Style/XSL/`, 2003.

[Tid01] Doug Tidwell. *XSLT*. O'Reilly, 2001.