Extreme Markup Languages

Constraining Topic Maps

A TMCL declarative implementation

José Carlos Ramalho
Department of Informatics
University of Minho

Giovani Librelotto
Department of Informatics
University of Minho

Pedro Rangel Henriques
Department of Informatics
University of Minho

Keywords: Topic Maps; Semantic Validation; Knowledge Management; Semantic Web; XML Schema

Abstract

This paper describes the design of an XML language to formally specify constraints over Topic Maps (XTche). This language allows to express contextual conditions on classes of Topic Maps that are further processed by a XSLT based processor. With *XTche*, a topic map designer defines a set of restrictions that guarantee that a particular topic map is semantically valid.

Topic Maps tend to grow quite fast. Most times the designer has some restrictions in mind like: what kind of topics should be used for abstract concepts, what topics may link to resources and what topics can not, ... All these restrictions tend to blur when things get big or if the some member of the team changes. In these situatiations an automatic system able to validate the restrictions is desirable.

The constraining process presented in this paper is composed of a language and a processor. The language is based on XML Schema syntax (we have used the same syntax and concepts in a similar approach to RDFS). The processor is developed in XSLT language. XTche processor is very similar to the Schematron or XCSL processors: it's an high-level stylesheet that takes a *XTche* specification as input and generates a specific XSLT stylesheet. This stylesheet when applied to the Topic Map validates the constraints in the *XTche* specification.

In this paper we will show, in abstract terms and with concrete examples, how to specify Topic Maps schemas and constraints with *XTche*.

1 Introduction

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]**).

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.

Topic Maps are nowadays widely used within XML environments: in archives, for cataloging and indexing purposes; or in web browsers, for conceptual navigation or even in more complex applications like information systems integration.

To build reliable systems, like those referred, it is crucial to be sure about the complete correctness of the underlying semantic network.

Like in other fields, as formal language and document processing, it is wise to validate the syntax and semantics of a topic map before its use. This is precisely the focus of this paper: we propose *XTche*, a language to define Topic Maps Schema and Constraints. The validation process of a topic map based on a *XTche* specification will also be under the scope of the paper.

"Semantic Web, Ontology, and Topic Maps" is an overview about the basic concepts in the area of this work: Semantic Web, Ontology, and Topic Maps; it creates the context and motivation for our concern with the precise semantics of Topic Maps. A discussion about constraining Topic Maps is presented in "Constraints". "XTche - A Language for Topic Maps Schema and Constraints" describes *XTche*; before the introduction of *XTche* specific semantic constructors, we distinguish schema and contextual constraints. Then the automatic analysis of a *XTche* specification (in order to generate a concrete validator) is discussed. "Related Work" compares our proposal with related work and exemplifies the use of our constraint language. A synthesis of the paper and hints on future work are presented in the last part, "Conclusion".

2 Semantic Web, Ontology, and Topic Maps

Semantic Web is concerned with the arrangement of web based information systems 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 **[Mon04]**:

- 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 research areas: Ontologies, Artificial Intelligence, Agents, Formal Logic, Languages, Graph Theory and Topology, etc. Our research 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 **[Wri01]**. 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 by a set of actors in a community;
- Define and to 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 **[PH03]** 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 a 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 **[Wri01]**, 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.

3 Constraints

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 **[DGM01]**) 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.

4 XTche - A Language for Topic Maps Schema and Constraints

This section presents a language to define constraints on Topic Maps, named *XTche*, with this language is possible to specify constraints about a topic map family in order to guarantee the topic map semantic correctness. Before describing the language and its processor (a validator generator), we give the motivation behind its development, and discuss what a constraint is in this context.

As shown in “Semantic Web, Ontology, and Topic Maps”, 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 creator's ideas).

Adopting XTM format, the syntactic validation of a topic map is assured by any XML parser because XTM structure is defined by a DTD **[PM01]**. However, it is well known that structural validity does not mean the complete correctness - semantics should also be guaranteed.

Using XML Schema instead of DTD improves the validation process because some semantic requirements (domain, occurrence number, etc.) can be added to the structural specification. XML parsers can still deal with that task.

However other 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.

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) **[NM03]**. *XTche* language meets all the requirements in that list. For that purpose, *XTche* has a set of constructors to describe constraints in Topic Maps, as will be detailed in the next subsections. But the novelty of the proposal is that the language also permits the definition of the topic map structure in an XML Schema style; it is no more necessary a separate syntactic description. A *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.

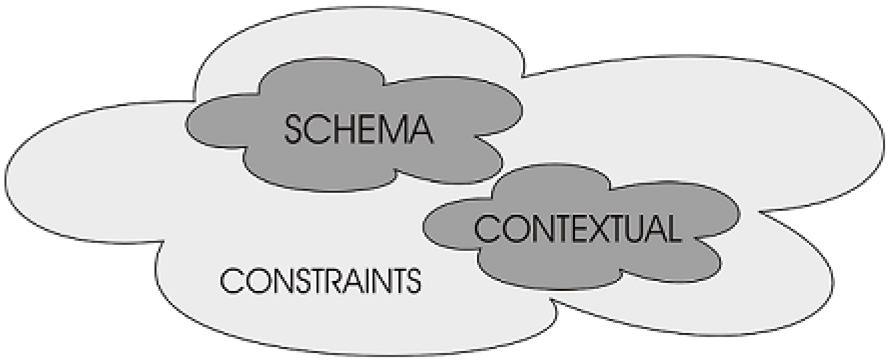
A Topic Map Schema defines all topic types, scopes, subject indicators, occurrence types, association types, association roles, and association players. So, it is possible to infer a topic map skeleton (written in XTM) from the schema; the user or an application (like *Oveia* **[LSRH04]**, a *Metamorphosis* **[LRH03]** module) must only fill it in (with data extracted from the information resources) to obtain the topic map instances. This functionality (skeleton derivation and syntactic validation) will not be more developed in this paper, as this paper is devoted to the semantic aspects.

4.1 Schema Constraints and Contextual Constraints

XTche is designed to allow users to constrain any aspect of a topic map; for instance: topic names and scopes, association members, topics allowed as topic type, roles and players allowed in an association, instances of a topic (enumeration), association in which topics must participate, occurrences cardinality, etc.

These constraints can be divided in two parts: *schema constraints* and *contextual constraints*. The first subset defines the Topic Maps Schema (i.e., the structure of topics, associations, and occurrences); the second one is applied over particular conditions in a topic map. “Schema and Contextual Constraints” shows this classification.

Figure 1: Schema and Contextual Constraints



[Graphic entity *schemaXconstraints* — file:/C:/Documents and Settings/JCR/My Documents/Artigos/2005/extreme2005/ELM2005Rama020101.png]

An extensive list of Topic Maps constraints **[NM03]**, classified as *schema* or *contextual constraints*, is presented below:

1. Schema constraints:
   - Topic of type T must have a specified number of explicit names/occurrences/subject-indicators (cardinality);
   - Topic of type T must have as name/occurrence/subject-indicators a value matching a particular pattern;
   - Topic of type T must (not) have a name/occurrence with scope S;
   - Topic of type T must have a name/occurrence, that is instance of topic type T, in scope S;
   - Topic of type T must (not) have an occurrence that is of type O;
   - Topic T can (only/not) be used as an association role topic in association with

   association type A;
   - Topic of type T can (only/not) be used as an association player topic in association with association type A;
   - A list of topics are instances of topic type T;
   - Association with association type A must be in scope S;
   - Association with association type A has (only/at least/not) roles R1 and R2;
   - Association of type A must have (only/at least) two participating topics where one is of type T1 and the other is of type T2;
   - Association of type A must (not) have the role R being played by a topic of type T;
   - Association of type A has role R played by exactly two topics of type T (cardinality);
   - Association of type A has role R1 played by topic of type T1 and role R2 played by topic of type T1 or T2;
   - Association of type A must have dependencies between player types;
   - Occurrence of type O can (only/not) be a characteristic of topics of type T;
   - Occurrence of type O can (only/not) be used within scope S;
   - Occurrence of type O must have locators that match a URI pattern P;
2. Contextual constraints:
   - Topic T can (only/not) be used for typing other topics;
   - Topic T can (only/not) be used for typing subject indicator;
   - Topic T can (only/not) be used for typing basenames;
   - Topic T can (only/not) be used for typing occurrences;
   - Topic T can (only/not) be used for typing associations;
   - Topic T can (only/not) be used as an association scope;
   - Topic T can (only/not) be used as an association role topic;
   - Topic of type T can (only/not) be used for scoping occurrences;
   - Topic of type T can (only/not) be used for scoping base names;
   - Topic of type T can (only/not) be used for scoping associations;
   - Topic of type T can (only/not) be used as an association player topic;

Although all the concerns in the previous list are constraints, there is actually a slight difference in the way of dealing with the two subsets. So, the wish to have *XTche* expressing both - contextual constraints and schema constraints - has a direct influence in the design of the language and its processing. We will care about that in the following subsections.

4.2 An XML Schema-based language

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 (more details in the "XTche Processor and TM-Validator").

XTche also takes advantage of XML Schema data types to validate some constraints (see "Schema constraint specification").

XTche is an XML Schema-based language. All *XTche* specifications are XML Schema instances; but, obviously, not all XML Schema instances are *XTche* specifications.

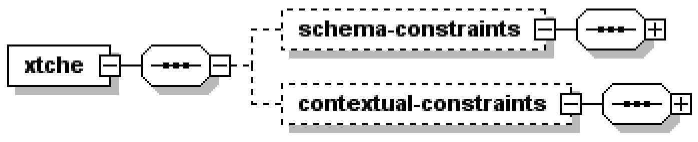
"XTche Skeleton" describes the skeleton for all the *XTche* specifications. That skeleton is a generic, but incomplete, XML Schema that must be fulfilled with particular constraints for each case, as detailed in "Schema constraint specification" and "Contextual constraint specification". To write those constraints, the basic schema language is extended by a set of domain specific attributes that are defined in a separated file (also presented in "XTche Skeleton") imported by the skeleton.

Like any other schema, before processing an *XTche* specification (in order to generate a *TM-Validator*), its correctness should be checked. However, an usual XML Schema-based parser is not enough to do that desired validation; we had to extend it with one more layer (to take care of the above referred domain specific attributes) as will be explained in "XTche-Specification Validation Processor".

4.2.1 XTche Skeleton

An *XTche* specification is a schema where the `<xtche>` element is the root. This element is composed of a sequence, where two elements are allowed: `<schema-constraints>` - that specifies the schema constraints - and `<contextual-constraints>` - that specifies the contextual constraints. Both 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.

Figure 2: *XTche* specification inicial structure



[Graphic entity *xtche-structure* — file:/C:/Documents and Settings/JCR/My Documents/Artigos/2005/extreme2005/ELM2005Rama020102.png]

The diagram presented in " XTche specification inicial structure" represents the code presented below and it corresponds to the generic skeleton referred above (that must be completed in each case). It begins with root specification, where the namespace `xtche` must be declared with the value `http//www.di.uminho.pt/~gepl/xtche`. After that, it is necessary to import the schema that specifies the *XTche* attributes, as discussed above in the introduction to this section. This schema is available at `http://www.di.uminho.pt/~gepl/xtche/xtche-schema.xsd`. Finally, a sequence of two non-required elements (contextual-constraints and schema) allows the definition of all the constraints necessary to validate the particular topic maps under definition.

```
<?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/20
    <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 2 -->
                            ...
                            <!-- 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 2 -->
                            ...
                            <!-- contextual constraint N -->
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

The specific XML Schema for XTche attributes is shown below. This schema (imported by the skeleton above) defines all the attributes required to qualify the elements in an XTche specification.

```
<xs:schema targetNamespace="http://www.di.uminho.pt/~gepl/xtche" elementFormDefa
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:xtche="http://www.di.uminho.pt
    <!-- ============================================================ -->
    <!-- XTche attributes for schema constraints -->
    <!-- ============================================================ -->
    <xs:attribute name="topic"/>
    <xs:attribute name="topicType"/>
    <xs:attribute name="subjectIndicator"/>
    <xs:attribute name="scope"/>
    <xs:attribute name="baseName"/>
    <xs:attribute name="baseNamePattern"/>
    <xs:attribute name="baseNameScope"/>
    <xs:attribute name="occurrence"/>
    <xs:attribute name="occurrenceType"/>
    <xs:attribute name="occurrenceScope"/>
    <xs:attribute name="associationType"/>
    <xs:attribute name="associationScope"/>
    <xs:attribute name="associationRole"/>
    <xs:attribute name="associationPlayer"/>
    <!-- ============================================================ -->
    <!-- XTche attributes for contextual constraints -->
    <!-- ============================================================ -->
    <xs:attribute name="topicType-Exclusive"/>
    <xs:attribute name="topicType-Forbidden"/>
    <xs:attribute name="baseNameType-Exclusive"/>
    <xs:attribute name="baseNameType-Forbidden"/>
    <xs:attribute name="baseNameScope-Exclusive"/>
    <xs:attribute name="baseNameScope-Forbidden"/>
    <xs:attribute name="subjectIndicator-Exclusive"/>
    <xs:attribute name="subjectIndicator-Forbidden"/>
    <xs:attribute name="occurrenceType-Exclusive"/>
    <xs:attribute name="occurrenceType-Forbidden"/>
    <xs:attribute name="occurrenceScope-Exclusive"/>
    <xs:attribute name="occurrenceScope-Forbidden"/>
    <xs:attribute name="associationType-Exclusive"/>
    <xs:attribute name="associationType-Forbidden"/>
    <xs:attribute name="associationScope-Exclusive"/>
    <xs:attribute name="associationScope-Forbidden"/>
    <xs:attribute name="associationRole-Exclusive"/>
    <xs:attribute name="associationRole-Forbidden"/>
    <xs:attribute name="associationPlayer-Exclusive"/>
    <xs:attribute name="associationPlayer-Forbidden"/>
    <!-- ============================================================ -->
    <!-- XTche elements -->
    <!-- ============================================================ -->
    <xs:element name="baseName"/>
    <xs:element name="occurrence"/>
    <xs:element name="subjectIndicator"/>
</xs:schema>
```

Those two XML Schemas (the first one incomplete) are all that is necessary to learn the general structure of the new *XTche* language to define the schema of Topic Maps. To use it, the topic map designer shall also know how to write the constraints he wants to be satisfied by each particular topic map instance. However, before explaining both the schema and contextual constraints, let us just talk about the *XTche* validation that will guarantee that a particular specification is a well-formed XML-Schema and a valid *XTche* description.

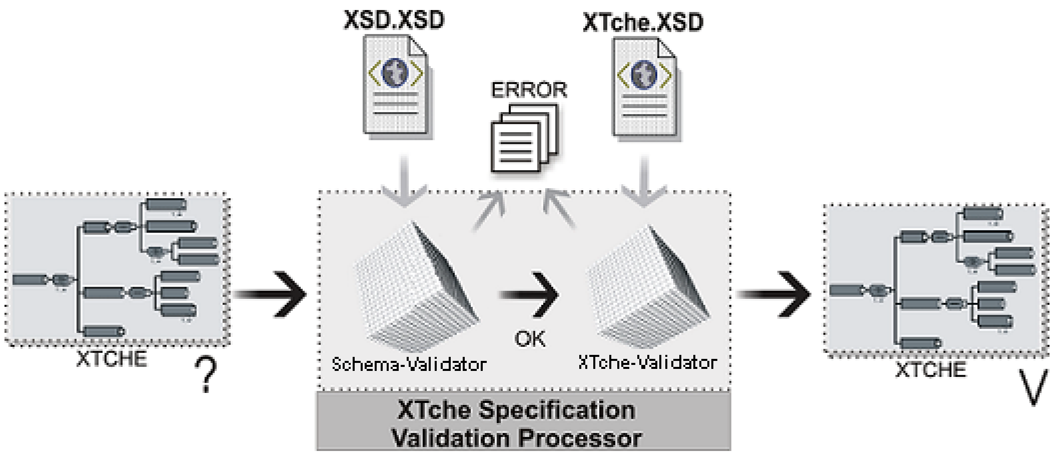
4.2.2 XTche-Specification Validation Processor

XTche-Specification Validation Processor (*XTche-SpecVP*) checks the structure of a *XTche* specification in agreement with the standard schema for XML Schema language and the specific schema for *XTche* language, presented in last subsection.

It is possible to make an analogy between an *XTche* specification and an XML document: an *XTche* instance should be a well-formed XML document[1](in this case a well-formed XML Schema), but it also needs to be valid according to *XTche* schema. So, its correctness is assured by *XTche-SpecVP* that performs separately those two verifications.

"XTche-Specification Validation Processor" depicts *XTche-SpecVP* behavior. Initially it verifies if the source *XTche* specification is a valid XML Schema (any XML parser is able to do this simple task); if no errors are found, the processor executes the second step that consists on the verification of its compliance against the rules defined below. Errors are reported as they occur. The *XTche* specification is correct if no errors are reported.

Figure 3: XTche-Specification Validation Processor



[Graphic entity *XTcheSpecVP* — file:/C:/Documents and Settings/JCR/My

Documents/Artigos/2005/extreme2005/ELM2005Rama020103.png]

Rules verified by *XTche-SpecVP* in the second phase:

1. in Schema Conditions:
   1. the first level of `<schema-constraints>` subelements must have only one of these attributes: `associationType`, `topicType`, `occurrenceType`, or `baseNameType`;
   2. elements with `associationType` attribute must have subelements with only one of these attributes: `associationScope`, `associationRole`, `associationRole-Exclusive`, `associationPlayer`, or `associationPlayer-Exclusive`;
   3. elements with `associationType` attribute can have a `<xs:any>` subelement (its namespace must have the value `##any`);
   4. elements, that have attributes with `Exclusive` suffix, do not have subelements;
   5. elements with `associationScope` or `associationPlayer` attributes do not have any subelements;
   6. elements with `associationRole` attribute can have subelements with `associationPlayer` attribute;
   7. elements with `associationRole` attribute can also have `minOccurs` and `maxOccurs` attributes;
   8. elements with `associationRole` attribute can have a `<xs:any>` subelement (its namespace must have the value `##any`);
   9. elements with `associationPlayer` attribute can also have `minOccurs` and `maxOccurs` attributes;
   10. elements with `topicType` attribute must have subelements with only one of these attributes: `baseName`, `baseNameScope`, `occurrenceType`, `occurrenceScope`, `subjectIndicator`, or `topic`;
   11. elements, that have one of these attributes - `baseNameScope`, `occurrenceScope` - do not have subelements;
   12. elements with `occurrenceType`, `occurrenceScope`, attributes can also have `minOccurs` and `maxOccurs` attributes;
   13. elements with `topicType` attribute can have a `<xs:any>` subelement (this namespace must have the value `##any`) if all its subelements have `topic` attribute;
   14. elements with `occurrenceType` attribute can have subelements with `occurrenceScope` attribute.

1. in Contextual Conditions:
   1. attributes with `Forbidden` suffix must only be found in subelements children of `<contextual-conditions>`;
   2. an attribute with `Exclusive` suffix must be unique in its element;
   3. an element can have more than one attribute with `Forbidden` suffix, but all its attributes must have this suffix;

4.2.3 Schema constraint specification

The schema constraint specification follows closely XTM schema. Each schema specification is a subelement of `<schema-constraints>`, the first subelement of `<xtche>`, as shown in the skeleton previously presented. It has several elements structured according to XTM schema.

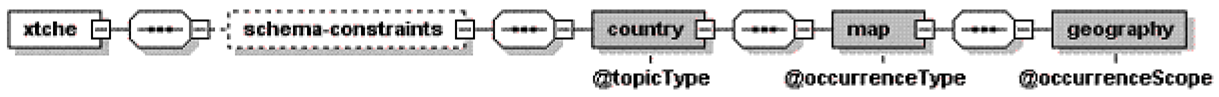
For instance: to specify that topics of type *country* must have occurrence of type *map* in the scope *geography*, we should write the following code:

```
<xs:element name="country">
```

```
       <xs:complexType>
          <xs:sequence>
             <xs:element name="map">
                <xs:complexType>
                   <xs:sequence>
                      <xs:element name="geography">
                         <xs:complexType>
                            <xs:attribute ref="xtche:occurrenceScope"/>
                         </xs:complexType>
                      </xs:element>
                   </xs:sequence>
                   <xs:attribute ref="xtche:occurrenceType"/>
                </xs:complexType>
             </xs:element>
           </xs:sequence>
           <xs:attribute ref="xtche:topicType"/>
        </xs:complexType>
     </xs:element>
```

"An XTche specification" shows the diagrammatic view.

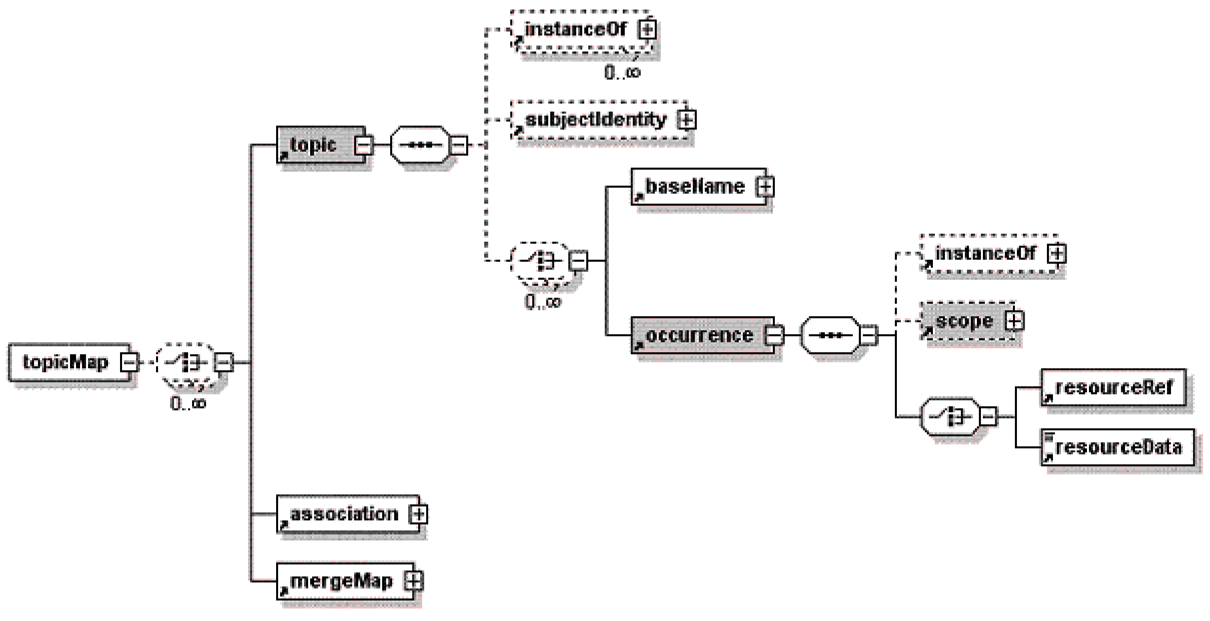
Figure 4: An XTche specification



[Graphic entity *ex-schema-constraint1* — file:/C:/Documents and Settings/JCR/My Documents/Artigos/2005/extreme2005/ELM2005Rama020104.png]

To compare the XTche specification in "An XTche specification" and XTM structure, "XTM schema" exhibits a part of that schema, where the path to occurrence scope is in contrast.

Figure 5: XTM schema



[Graphic entity *xtm-schema-scope1* — file:/C:/Documents and Settings/JCR/My Documents/Artigos/2005/extreme2005/ELM2005Rama020105.png]

As shown in “An XTche specification” one schema constraint is a sequence of concrete topics (country, map, and geography) each one qualified by an associated XTche attribute. A similar description in XTM (“XTM schema”) uses generic element names (`topic`, `occurrence`, and `scope`) and defines the concrete data via attributes associated to those elements (see code below). This systematic correspondence justifies a previous statement that the XTM code can be inferred from the XTche specification. However, the first contains more semantic information.

```
    <topic id="xxx">
        <instanceOf>
            <topicRef xlink:href="#country"/>
        </instanceOf>
        <occurrence>
            <instanceOf>
                <topicRef xlink:href="#map"/>
            </instanceOf>
            <scope>
                <topicRef xlink:href="#geography"/>
            </scope>
        </occurrence>
    </topic>
```

Now a more sophisticated XTche example inspired in the *E-Commerce Application*, subsection 6.1 of **[NM03]**, is described. The relationship defined by the association of type `is-making-order` has two association roles: `customer` and `order`. The role `order` must be played by, at least, one topic of type `order`, and the role `customer` played by one player, which must be a topic of type `customer` or `employee`. To specify this kind of constraint, the code must be written as follows.

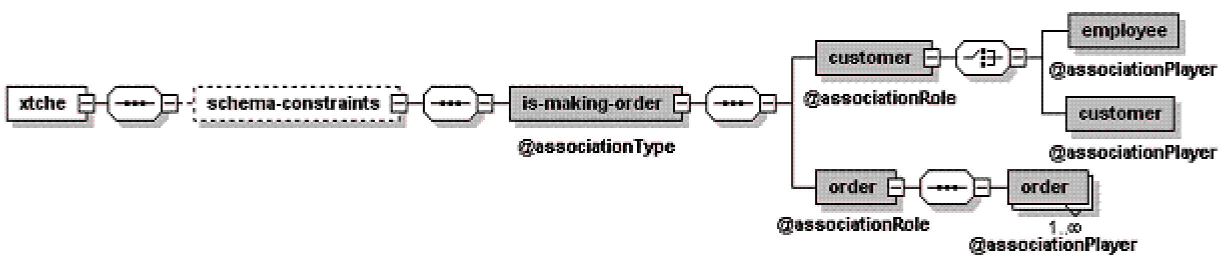
```
<xs:element name="is-making-order">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="customer">
        <xs:complexType>
          <xs:choice>
```

```
            <xs:element name="employee">
              <xs:complexType>
                <xs:attribute ref="xtche:associationPlayer"/>
              </xs:complexType>
            </xs:element>
            <xs:element name="customer">
              <xs:complexType>
                <xs:attribute ref="xtche:associationPlayer"/>
              </xs:complexType>
            </xs:element>
          </xs:choice>
          <xs:attribute ref="xtche:associationRole"/>
        </xs:complexType>
      </xs:element>
      <xs:element name="order">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="order" maxOccurs="unbounded">
              <xs:complexType>
                <xs:attribute ref="xtche:associationPlayer"/>
              </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>
```

The element `<xs:choice>` inside the role `customer` defines the two alternative association players that can be found playing that role in the topic map. The attribute `maxOccurs`, associated with player `order`, defines the cardinality (in this case, one or more) of the players allowed in role `order`.

This XTche specification above can be depicted by the diagram shown in “An XTche specification”.

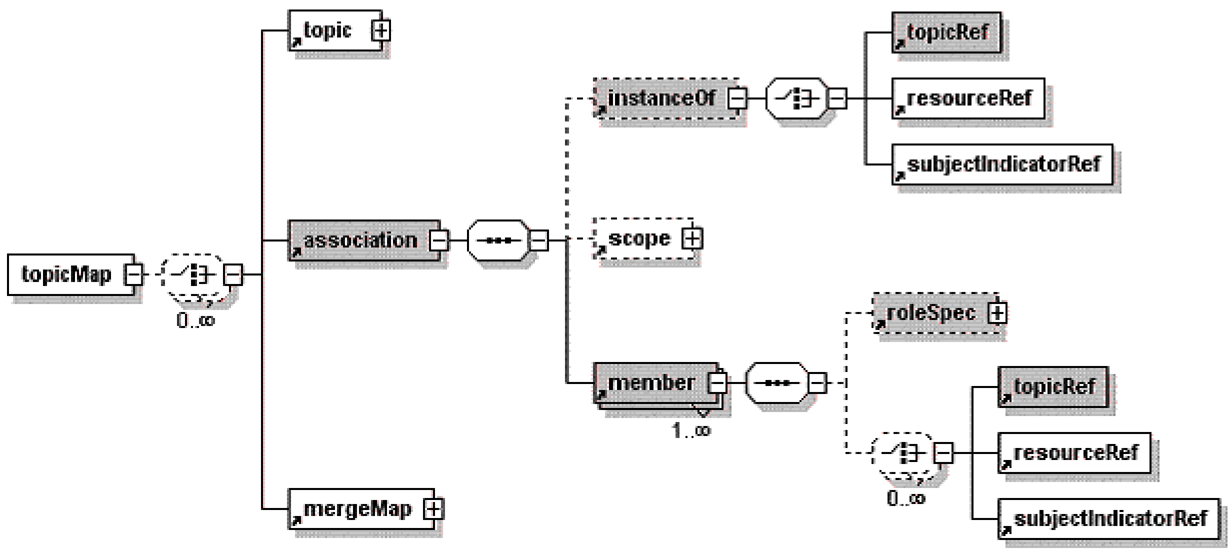
Figure 6: An XTche specification



[Graphic entity *ex-schema-constraint2* — file:/C:/Documents and Settings/JCR/My Documents/Artigos/2005/extreme2005/ELM2005Rama020106.png]

For comparison, “XTM schema” shows the classic XTM structure for that association.

Figure 7: XTM schema



[Graphic entity *xtm-schema-scope2* — file:/C:/Documents and Settings/JCR/My Documents/Artigos/2005/extreme2005/ELM2005Rama020107.png]

Though, the XTche specification represented in "An XTche specification" can validate the intended constraints on Topic Maps containing the following code:

```
<topic id="xxx">
    <instanceOf>
        <topicRef xlink:href="#customer"/>
    </instanceOf>
</topic>
<topic id="yyy">
    <instanceOf>
        <topicRef xlink:href="#order"/>
    </instanceOf>
</topic>

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

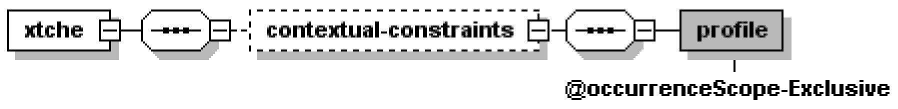
Once again, we think that the observation of both diagrams makes clear the difference between a XTche specification and a XTM specification enhancing the advantage of XTche.

4.2.4 Contextual constraint specification

Contextual constraints appear in the *XTche* specification as subelements of `<contextual-constraints>`, the second subelement of <xtche>, as explained in “XTche Skeleton” (see the skeleton included). They do not have more subelements; they only have attributes.

For instance, to create a topic `profile` and say that *it can be used for scoping occurrences and nothing else*, all that we have to do is to add a `<profile>` subelement with an `@occurrenceScope-Exclusive` attribute, as shown in “A contextual constraint specification example”.

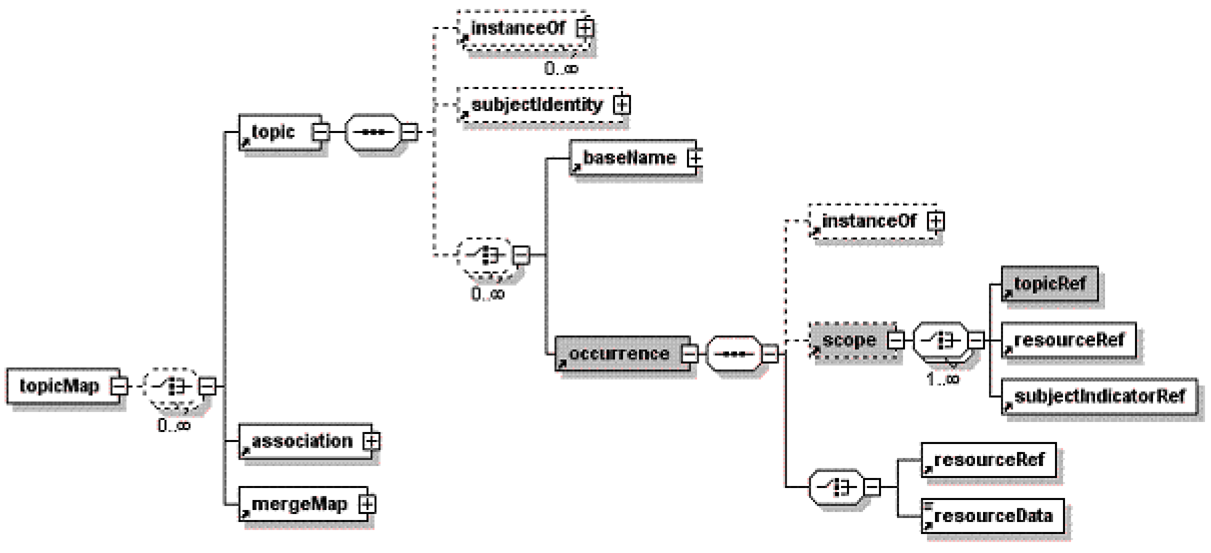
Figure 8: A contextual constraint specification example



[Graphic entity *ex-contextual-constraint1* — file:/C:/Documents and Settings/JCR/My Documents/Artigos/2005/extreme2005/ELM2005Rama020108.png]

Such a restriction can not be made explicitly in XTM; this is why we call this family of constraints *contextual*, to distinguish it from those that can be included in XTM (called *schema-constraints*). This way, to validate the above stated restriction, the *TM-Validator* needs to check if the topic `profile` is only used as a `topicRef` element at the end of `//occurrence/scope` path, as shown in “XTM schema”.

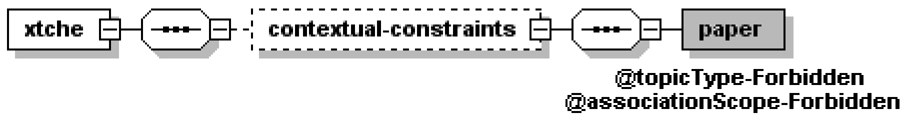
Figure 9: XTM schema



[Graphic entity *occurScope-exclusive* — file:/C:/Documents and Settings/JCR/My Documents/Artigos/2005/extreme2005/ELM2005Rama020109.png]

As a second example, consider that we wish to create a topic `paper` and state that *it can not be used for typing other topics or associations*. In *XTche* language, we simply need to add a `<paper>` subelement with the attributes `@topicType-Forbidden` and `@associationType-Forbidden`, as shown in “Another contextual constraint specification example”.

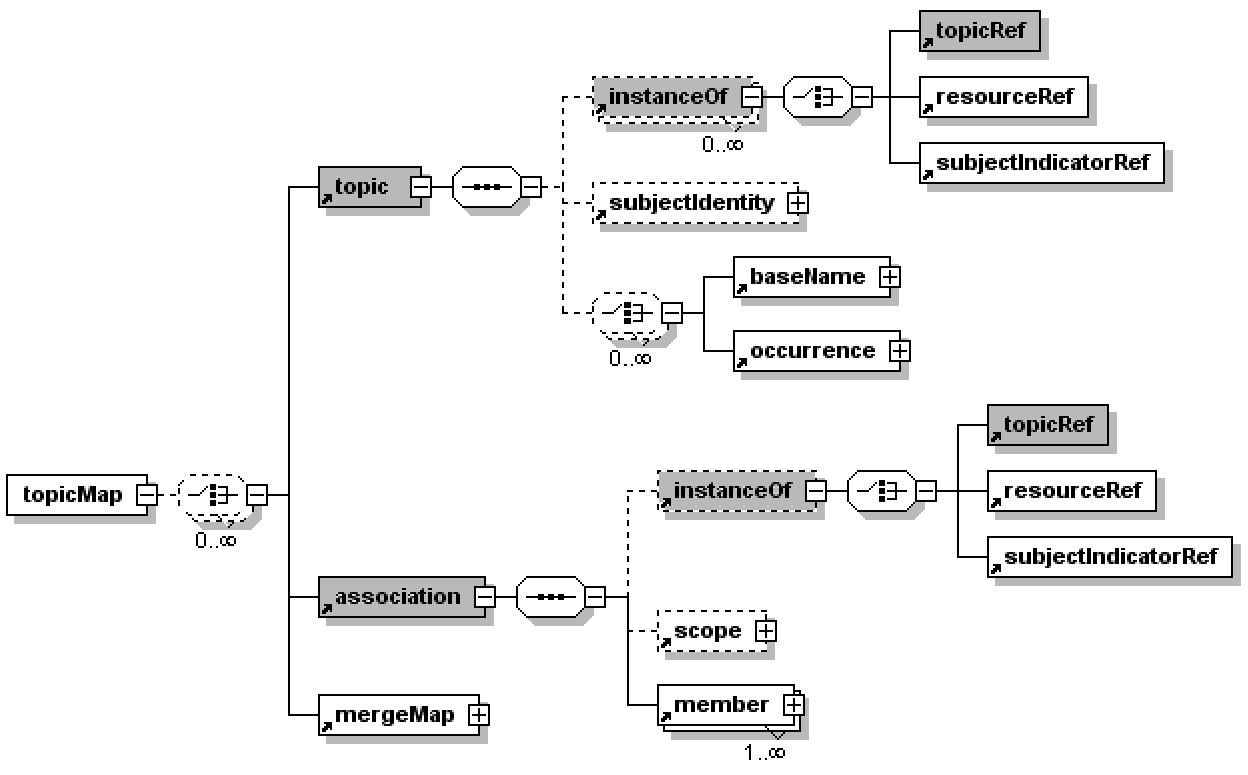
Figure 10: Another contextual constraint specification example



[Graphic entity *ex-contextual-constraint2* — file:/C:/Documents and Settings/JCR/My Documents/Artigos/2005/extreme2005/ELM2005Rama020110.png]

"XTM schema" shows the places where the topic `paper` can not be found, according to the constraint described in "Another contextual constraint specification example", but unfortunately in XTM there is no explicit systematic way to impose that; the designer should pay attention and avoid its use in the undesirable places. However the *TM-Validator* will ckeck the two contexts. If it finds a reference to the topic paper in one of these two places it will issue an error message.

Figure 11: XTM schema



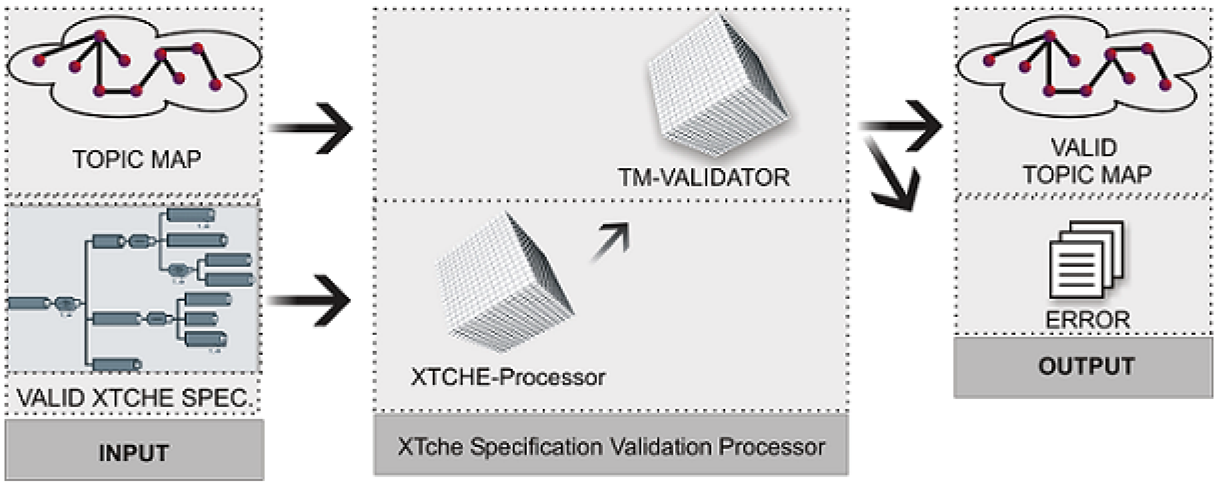
[Graphic entity *topic-type-forbidden* — file:/C:/Documents and Settings/JCR/My Documents/Artigos/2005/extreme2005/ELM2005Rama020111.png]

4.2.5 XTche Processor and TM-Validator

Each *XTche* specification - listing all the conditions (involving topics and associations) that must be checked - *specifies a specific topic map validation process* (*TM-Validator*), enabling the systematic codification (in XSL) of this verification task. We understood that in those circunstances, it was possible to generate automatically this *TM-Validator*. For that purpose, we developed another XSL stylesheet that translates an *XTche* specification into the *TM-Validator* XSL code.

The *XTche processor* is the *TM-Validator* generator; it behaves precisely like a compiler generator and it is the core of our architecture, as can be seen in “ XTche Architecture”. It takes *a valid topic map schema and constraint specification* (an XML instance, written according to the *XTche* schema), verified by the *XTche-SpecVP* introduced in “XTche-Specification Validation Processor”, and generates an XSL stylesheet (the TM-Validator) that will process an input topic map and will generate an *ok/error message* (an ok message states that the topic map is valid according to the *XTche* specification).

Figure 12: XTche Architecture



[Graphic entity *XTche* — file:/C:/Documents and Settings/JCR/My Documents/Artigos/2005/extreme2005/ELM2005Rama020112.png]

Both XSL stylesheets (the generator and the validator) are interpreted by a standard XSL processor like Saxon[2], what in our opinion is one of the benefits of the proposal.

During the development of this generator we found some problems that had a strong impact in the final algorithm. The most important was the ambiguity in constraint selection; until now, we have just said that an *XTche* specification is composed of a set of constraints; we did not say that these constraints are disjoint in terms of context; in some cases there is a certain overlap between the contexts of different conditions; this overlap will cause an error when transposed to XSL; XSL processors can only match one context at a time. The solution we have adopted to overcome this problem was to run each constraint in a different mode (in XSL each mode corresponds to a different traversal of the document tree).

5 Related Work

AsTMa! **[Bar03]** is another Topic Maps constraint language, and Robert Barta also proposes a mechanism to validate a topic map document against a given set of rules. This language uses *AsTMa=* **[Bar04]**, the authoring language, and extends it with several new language constructors, and logic operators (like NOT, AND and OR), simple logical quantifiers and regular expressions. *AsTMa!* exposes some features of a future TMCL.

The topic declaration below defines a topic with an id (`pele`) which matches that in the constraint, the type also matches (`person`) and so does the basename (*Pelé*). Additional topic characteristics such as the inline and occurrence characteristic (with occurrence type `profile`) does not affect the matching, and therefore the constraint is satisfied as long as the minimal requirements are met.

```
        pele (person)
bn: Pelé
in: Pelé is the best soccer player of all-time
oc (profile): http://www.time.com/time/time100/heroes/profile/pele01.html
```

To verify if every `person` has at least an `URL`, it is necessary to write a sentence like the one shown in the code fragment below.

```
forall $r [ * (person) ]
   => exists $r [ oc : ?is_url ] is-reified-by report-has-URL-S
```

In order to define that an `>is-written-by` association type relates two members, one is a `person`, and the other is a `paper`, the *AsTMa!* code below must be written:

```
forall [ (is-written-by) ]
   => exists ] (is-written-by)
                user  : $p
                thing : $t [
      and
      exists [ $p (person) ]
      and
      exists [ $t (paper) ] is-reified-by person-writes-papers-S
```

In another 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.

Comparing *XTche* with the other known approaches, some advantages of *XTche* emerge: *XTche* has a XML Schema-based language, a well-known format. In addition, *XTche* allows the use of an XML Schema graphic editor, like XMLSpy. With the diagrammatic view, it is easy to check visually the correctness of the specification. Moreover, *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.

Talking about the constraints covered by these languages, *XTche* and *AsTMa!* have more mechanisms to check the validity of Topic Maps than the Eric Freese proposal.

6 Conclusion

In this paper we introduced a Topic Maps Validation System - *XTche Constraint language and its processor*. 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. *XTche* complies with all requirements stated for TMCL but it 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), eliminating the need for two separated specifications; and on the other hand it enables the use of an XML Schema editor (for instance, XMLSpy) to provide a graphical interface and the basic syntactic checker (the first stage of the *XTche-SpecVP*).

We succeeded in applying this approach to some case studies - *E-Commerce Application* (subsection 6.1 of **[Wri01]**) and a *personal video library management system* - 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.

Notes

1. The concept of being well-formed was introduced as a requirement of XML, to deal with the situation where a schema (DTD, XML Schema, or RelaxNG) is not available.

2. http://saxon.sourceforge.net/

Bibliography

[Bar03] *AsTMa!*, R. Barta, Bond University, TR., 2003, available at: http://astma.it.bond.edu.au/constraining.xsp.

[Bar04] *AsTMa= Language Definition*, R. Barta, Bond University, TR., 2004, available at: http://astma.it.bond.edu.au/astma=-spec-xtm.dbk

[BBN99] *ISO/IEC 13250 - Topic Maps.*, December, 1999, available at: http://www.y12.doe.gov/sgml/sc34/document/0129.pdf

[DGM01] *Professional XML Schemas*,, Wrox Press, 2001,

[Dod01] *Schematron: Validating XML Using XSLT*,, 2001,

[Fre02] *Using DAML+OIL as a Constraint Language for Topic Maps.*, 2002, available at: http://www.idealliance.org/papers/xml02/dx xml02/papers/05-03-03/05-03-03.html

[JLRH02] *Constraint Specification Languages: comparing XCSL, Schematron and XML-Schemas*,, XML Europe 2002, 2002,

[LRH03] *Ontology driven Websites with Topic Maps*,, The International Conference on Web Engineering, 2003,

[LSRH04] *Using the Ontology Paradigm to Integrate Information Systems*,, International Conference on Knowledge Engineering and Decision Support, (497–504), 2004,

[Mon04] *Questions and Answers*, March, 2004, available at: http://www.mondeca.com/english/faqs.htm

[NM03] *Topic Map Constraint Language (TMCL) Requirements and Use Cases*, 2003, available at: http://www.ontopia.net/topicmaps/materials/tao.html

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

[PH03] *XML Topic Maps: Creating and Using Topic Maps for the Web,*, Addison-Wesley, 2003,

[PM01] *XML Topic Maps (XTM) 1.0 - Annex D: XTM 1.0 Document Type Declaration (Normative)*, August, 2001, available at:
http://www.topicmaps.org/xtm/1.0/#dtd

[Rat03] *White Paper: The Topic Maps Handbook*, 2003, available at:
http://www.empolis.com/downloads/empolis_TopicMaps_Whitepaper20030206.pdf

[Wri01] *Topic Maps and Knowledge Representation*, February 2001, available at:
http://www.ontopia.net/topicmaps/materials/kr-tm.html