

A Rule-Based Conversion of XML Document Type Definition to a Conceptual Schema

Ronaldo dos Santos Mello^{1,2}
Carlos Alberto Heuser¹
Silvana Castano³

¹Federal University of Rio Grande do Sul - Informatics Institute
Cx. Postal 15064 - Porto Alegre - RS - Brazil. 91501-970
e-mail: {ronaldo,heuser}@inf.ufrgs.br

²Federal University of Santa Catarina - Informatics and Statistics Department
Cx. Postal 476 - Florianopolis - SC - Brazil. 88040-900
e-mail: ronaldo@inf.ufsc.br

³University of Milano - Information Science Department
Via Comelico, 39 - Milano - Italy. 20135
e-mail: castano@dsi.unimi.it

Abstract

XML raises as a standard for semi-structured and structured data representation and exchange over the Web. Currently, Web-based Information Systems require semantic integration mechanisms for XML data to obtain a global view of heterogeneous XML sources of a given domain. This paper describes a process for converting XML DTDs to conceptual schemata in XCM (XML Conceptual Model), a conceptual model based on ORM and ER models. This process is part of a bottom-up approach for integration of XML sources that takes a set of DTDs and generates an ontology that acts as an unified vocabulary of XML sources for querying purposes. XCM simplifies the integration activity because provides a semantically rich representation of XML sources. The core of the process is a set of *conversion rules* that consider the DTD structure and heuristics related to default semantic interpretations on such structure in order to generate the corresponding XCM concepts.

1 Introduction

XML (*eXtensible Markup Language*) raises as a standard for semi-structured and structured data representation and exchange over the *Web*. XML documents present data organized in a tag structure that may be in accordance to a schematic representation described by a DTD (*Data Type Description*). A DTD may be considered a logical model for XML because describe a hierarchical organization for XML data representation.

Conceptual models provide a high-level abstraction for information concerning to an application domain, being a useful tool for tasks like database design, reverse en-

engineering and semantic integration. Semantic integration techniques, in particular, allow the definition of a global view of heterogeneous data schemata of a same domain. Such global view acts further as a basis for an integrated service of data manipulation. Integrating conceptual schemata instead of logical schemata is preferable because conceptual models have a more clear semantics are not bounded to any specific data structure.

Integration mechanisms for XML schemata are now required by Information Systems in order to provide an unified view of several heterogeneous XML sources over the Web. Canonic conceptual representations of these sources simplifies this task. This paper focuses on describing a semi-automatic process for converting a DTD to a conceptual schema in XCM (*XML Conceptual Model*). XCM is the canonic conceptual model of an approach for integration of XML sources, under development at the *Informatics Institute of Federal University of Rio Grande do Sul*, Brazil. This process is strongly based on a *set of rules* that takes into account DTD elements and attributes, syntactical constructs (*sequences* and *choices*) and default semantic interpretations to generate the corresponding XCM concepts. Final adjustments in the XCM schema are accomplished by an human expert in order to provide a semantically correct conceptual abstraction of the XML data.

The paper is organized as follows. ...

2 Overall Context

2.1 XML and DTD

XML (*eXtensible Markup Language*) organizes character data through markup tags. An unit of information in XML is called *element*, which is composed by a *start tag* with the form `<element_name>`, a *content* (character data and/or a group of component elements) and an *end tag* with the form `<\element_name>`. An element may have *attributes* that are described into the start tag. An example of XML data is: `<author> <name> Ronaldo dos Santos Mello <\name> <eMail> ronaldo@inf.ufrgs.br <\eMail> <\author>`, that defines an element *author* composed by *name* and *eMail*.

The schema of XML data may be defined through a DTD (*Document Type Description*). A DTD describes a set of nested elements and their optional attributes. An example of DTD to the domain of *Conferences* is shown in figure 1 (a).

An element content model in a DTD may be defined to have only character data (`#PCDATA` type) - a *simple element*; or to have only *component elements* or a mix of character data and component elements - a *composite element*. An element with components may be defined through two DTD syntactical constructs: *sequences* and *choices*. A sequence (e_1, \dots, e_n) defines n component elements and the order that they must appear. A choice ($e_1 / \dots / e_m$) defines m alternatives of component elements. A component element may be another embedded sequence or choice definition - a *complex component element*; just the name of the component element - a *simple component element*; or a `#PCDATA` type - a *lexical component*. A simple component element that occurs several times as a component element is a *repeated simple component element*.

The *regular expression operators* '?', '*' and '+' indicates allowed occurrences of component elements. An element content model or a component element followed by a regular expression operator '*' or '+' is called a *repeatable element*.

Attributes may be associated to an element. Each attribute has a name, a datatype and optional constraints that restrict the permitted values to an enumeration or a fixed value, or defines it as a required element property.

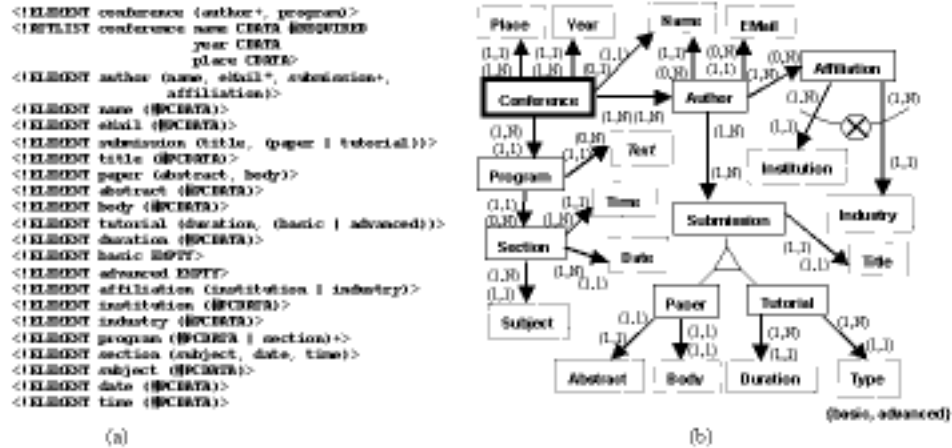


Figure 1: A DTD to the domain of *Conferences* (a) and the corresponding XCM schema (b)

2.2 The Integration Approach

A bottom-up integration approach (*ref. paper WIW*) is supported by a system architecture for accessing XML sources (*ref thesis*). This system accomplishes XML schemata integration, query translation to XML sources and merge of query results. This approach is defined as a semi-automatic process that comprises the *integration layer* of the system architecture, as shown in figure 2.

The integration process has two steps. In the first step, called *DTD-to-XCM Conversion*, each DTD associated to an XML source is converted to a conceptual schema in XCM (see section 3). In the second step, called *XCM Integration*, a set of XCM schemata are integrated to generate an *Ontology*. The ontology provides an unified conceptual vocabulary for all considered DTD elements and attributes and acts as a front-end for queries to XML sources. The process is semi-automatic because consider the intervention of an *human expert* to validate the generated XCM schemata as well as the ontology.

The focus of this paper is on the *DTD-to-XCM conversion*, that, in turn, has three steps: (i) *conversion rules application*; (ii) *analysis of XML instances* and; (iii) *user validation*. Steps (i) and (iii) are detailed in sections 4 and 5, respectively. Step (ii) is optional (set to be executed by an human expert) because may increase the integration process complexity if the number of XML instances in the considered XML sources is large. Basically, this analysis tries to infer:

- The cardinality of the relationship between a component element and the element that contains it. Such information is not available in a DTD but it is required in a relationship defined in a conceptual model. A cardinality N is set as default if an assertion cannot be done;
- More precise data types to #PCDATA components. Such checking tries to match these component's values with one of the following data types: *integer, float, character, date* or *time*. If no one match succeeds, the *string* data type is set as default.

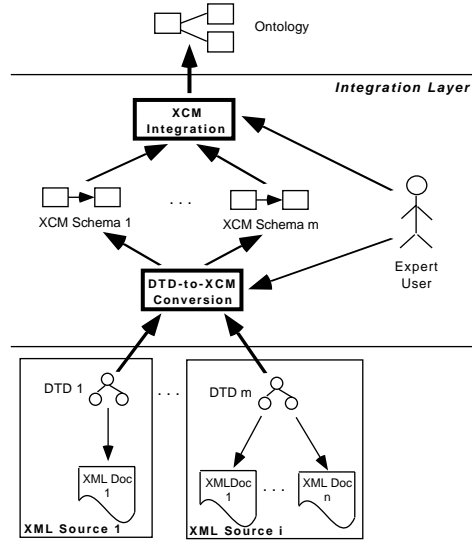


Figure 2: The bottom-up integration approach

More details about this analysis of XML instances as well as the overall integration approach are available in (*ref Thesis*).

3 XCM

XCM (*XML Conceptual Model*) is a conceptual abstraction for XML data modeled through a DTD. It supports the representation of DTD elements and attributes, and their typical composition relationships with associated cardinality constraint. Inheritance relationships are also supported as a type of mutual exclusion association or an implied specialization between elements. Despite of being applied to model DTDs, XCM is suitable for semi-structured data modeling considering that it is able to abstract semi-structured tree-based logical models, like *XML-schema*, *OEM* and *YaT*.

Section 3.1 presents an overview of the XCM constructs and section 3.2 formally defines the XCM concepts and relationships¹.

3.1 Overview

XCM is based on ORM (*Object with Role Model*) [?] and EER (*Extended Entity-Relationship*) [?] models. From ORM, it borrows the definitions of *lexical concept*, *non-lexical concept* and *disjoint relationships*. From EER, it borrows the *cardinality constraint* (cardinality pair) and *inheritance relationship* notations. Besides, XCM introduces *directed relationships* to model semi-structured data composition and *root concepts* that are special non-lexical concepts.

Figure 1 (b) presents the corresponding XCM modeling to the DTD of figure 1 (a). Labeled rectangles are used to model XCM concepts. A thick rectangle in XCM indicates a *root concept*, like *Conference*. Root concepts have a particular definition

¹For sake of paper space, XCM constraints are not formally described.

in XCM because they are the first concepts that must be matched in a search to an XML source. *Non-lexical concepts* are represented as solid rectangles, like *Author*. A non-lexical concept models information that is composed by other information. *Lexical concepts* are represented as dotted rectangles, like *Year*. A *lexical concept* models information that has direct computer representation, i.e., has an associated value. Lexical concepts may be specialized in *enumerated lexical concepts*, that have a list of permitted values, like *TutorialType*.

A *composition relationship* is a binary association between concepts with a direction indication from the composite element to the component element. It is denoted by an arrow with direct and inverse cardinality pairs. The relationship between *Conference* and *Place* is an example: a *Conference* takes place in one and only one *Place*, and a *Place* may receive one or more *Conferences*.

XCM still supports two types of *mutual exclusion relationships* to model alternative representations of semi-structured data (the case of DTD choices): *inheritance relationship* and *disjoint relationship*. An *inheritance relationship* (triangle graphic notation) represents an association between concepts where a semantics of specialization with mutual exclusion makes sense. This is the case of the *Work* concept, that may be a *Paper* or a *Tutorial*. A *disjoint relationship* (an "X" circled graphic notation) is a mutual exclusion constraint on composition relationships. An example is the concept *Affiliation*, that may be related to an *Institution* or an *Industry*.

3.2 Formal Definitions

Let NL be the set of non-lexical concepts; L the set of lexical concepts; EL is the set of enumerated lexical concepts; CR the set of composition relationships; IR the set of inheritance relationships; and DR the set of disjoint relationships.

Definition 1 (Non-Lexical Concept) A non-lexical concept $nl \in NL$ has the form $nl = \langle name_{nl}, type_{nl} \rangle$, where $name_{nl}$ is the label of nl and $type_{nl}$ is the type of the non-lexical concept, with $type_{nl} \in \{ 'NL', 'RNL', 'VNL' \}$. A non-lexical concept has $type_{nl} = 'NL'$ as default.

Definition 2 (Root Concept) A root concept rnl is a non-lexical concept defined as $rnl = \langle name_{nl}, 'RNL' \rangle$.

Definition 3 (Virtual Non-Lexical Concept) A virtual non-lexical concept vnl is a non-lexical concept defined as $vnl = \langle name_{nl}, 'VNL' \rangle$. Vnl encapsulates a complex component element of a composite element, i.e., encapsulates the *sequence* or *choice* specification presented in the complex component element. Vnl is virtual because has no correspondence with any DTD element.

Definition 4 (Lexical Concept) A lexical concept $l \in L$ is a 2-uple with the form $l = \langle name_l, datatype \rangle$, where:

- $name_l$ is the label of l ;
- $datatype$ is the data type of l , with $datatype \in \{ string, integer, real, char, date, time \}^2$.

Definition 5 (Enumerated Lexical Concept) An enumerated lexical concept $el \in EL$ is a 3-uple with the form $el = \langle name_l, datatype, permittedvalues \rangle$, where:

²Currently supported data types.

- $name_l$ and $datatype$ have the same definition given to a lexical concept;
- $permittedvalue$ is the set of valid constants $\{<vc>\}$ that may be used as value of el , so that $\forall vc \in permittedvalue$ ($domain(vc = datatype)$) (the domain of each constant value must be the defined $datatype$ of el).

Definition 6 (Composition Relationship) A composition relationship $cr \in CR$ between two concepts is a 5-uple with the form $cr = <name_s, name_t, cardinality_s, cardinality_t, name_{cr}>$ where:

- $name_s$ is the label of the source concept, with $name_s \in nl.name_{nl}$, for $nl \in NL$;
- $name_t$ is the label of the target concept, with $name_t \in nl.name_{nl} \cup l.name_l$, for $nl \in NL$ and $l \in L \cup EL$;
- $cardinality_s$ is the cardinality (pair) of the source concept, with the form $cardinality_s = (minCardinality_s, maxCardinality_s)$;
- $cardinality_t$ is the cardinality (pair) of target concept, with the form $cardinality_t = (minCardinality_t, maxCardinality_t)$;
- $name_{cr}$ is the optional relationship label.

Definition 7 (Mutual Exclusion Relationship) A mutual exclusion relationship $mr \in IR \cup DR$ among concepts is a 3-uple with the form $mr = <name_s, datatype, targetConcepts>$ where:

- $name_s$ is the label of the source concept, with $name_s \in nl.name_{nl}$, for $nl \in NL$;
- $datatype$ is the type of the mutual exclusion relationship, with $datatype \in \{'IR', 'DR'\}$;
- $targetConcepts$ is the set of mutual excluded target concepts $\{<name_t>\}$ where $name_t$ is the label of a target concept, with $name_t \in nl.name_{nl} \cup l.name_l$, for $nl \in NL$ and $l \in L \cup EL$.

Definition 8 (Inheritance Relationship) An inheritance relationship $ir \in IR$ is a mutual exclusion relationship defined as $ir = <name_s, 'IR', targetConcepts>$.

Definition 9 (Disjoint Relationship) A disjoint relationship $dr \in DR$ is a mutual exclusion relationship defined as $ir = <name_s, 'DR', targetConcepts>$.

4 DTD to XCM Conversion Rules

The conversion rules are the core of the *DTD-to-XCM conversion* process. They consider the DTD syntax for element and attribute definition and heuristics regarding to semantic interpretations in order to automatically generate an XCM schema that requires as few as possible manual semantic adjustments. Such heuristics are called *semantic heuristics*.

The conversion rules are organized in two classes:

- **Transformation rules:** convert DTD elements and attributes in related XCM concepts. Such rules are organized in three classes:

- **Cardinality Rules:** defines the source cardinality in composition relationships. They are auxiliary rules used by the other two categories;
 - **Element Rules:** convert composite elements defined through *sequence* or *choice* constructs, and simple elements to XCM;
 - **Attribute Rule:** convert attributes.
- **Restructuring rules:** generate root concepts and perform simplifications on the XCM schema generated by the transformation rules.

The formal definition of these rules are presented in the following.

4.1 Transformation rules

4.1.1 Cardinality rules

Rules 1 and 2 determine cardinalities of composition relationships, considering regular expression operators associated to composite or component elements. Rule 3 defines a *general cardinality* that embodies a set of cardinalities.

Rule 1 (Source Cardinality Definition) A $cr \in CR$ between a *source non-lexical concept* generated from a composite element and a *target component concept* generated from a simple component element e , generates a *cardinality*_s:

1. **(0,1)**, if e is followed by the regular expression operator '?';
2. **(0,N)**, if e is followed by the regular expression operator '*';
3. **(1,N)**, if e is followed by the regular expression operator '+';
4. **(1,1)**, otherwise.

Rule 2 (Source Cardinality with Repetition Definition) A $cr \in CR$ between a *source non-lexical concept* generated from a repeatable composite element E and a *target component concept* generated from a simple component element e , generates a *cardinality*_s as follows:

1. **(0,N)**, if E is defined as a choice;
2. If E is defined as a sequence:
 - (a) **(0,N)**, if:
 - i. e or E is followed by the regular expression operator '*';
 - ii. e is followed by the regular expression operator '?' and E is followed by the regular expression operator '+';
 - (b) **(1,N)**, otherwise.

Rule 3 (General Cardinality) A set C of cardinality pairs $\{c_1, \dots, c_n\}$ so that $c_i = (c_iMin, c_iMax)$ for $c_i \in C$, generates a general cardinality (*minCardinality*, *maxCardinality*) where:

1. *minCardinality* = MIN(C), where MIN(C) is the lowest value of c_iMin , for $c_i \in C$;
2. *maxCardinality* = N, if $\exists c_i \in C$ ($c_iMax = N$); otherwise, *maxCardinality* = MAX(C), where MAX(C) is the highest value of c_iMax , for $c_i \in C$.

4.1.2 Element rules

This class of rules uses the heuristics proposed to the *DTD-to-XCM conversion* process: the *inheritance semantics* heuristic and the *enumeration semantics* heuristic. The first one is based on the use of a lexical dictionary for inferring an inheritance relationship. The last one verifies if a component element e of a composite element E is, in fact, a constant value of E (a type value for E).

Definition 10 (Inheritance Semantics) A pair of DTD element names (e_1, e_2) have inheritance semantics if it is possible to infer a hypernymy relationship (e_1 is more general than e_2) through a lexical dictionary, and e_1 and e_2 do not have inheritance conflict. Inheritance conflict means that there is not a same component element in the composition hierarchy rooted at e_1 but e_2 , and the composition hierarchy rooted at e_2 .

Definition 11 (Enumeration Semantics) A component element e has enumeration semantics if e is defined as an EMPTY type element without associated attributes.

Rules 4 to 8 are related to the conversion of a composite element E , with name $EName$, defined as a DTD *sequence*, as shown in figure 3. The basic followed reasoning concerns to the generation of a non-lexical concept for E , default lexical concepts for each one of its components e , if e is not already defined as a non-lexical concept, and composition relationships between E and e . When a component element is a lexical component element, a special name is generated to the corresponding lexical concept.

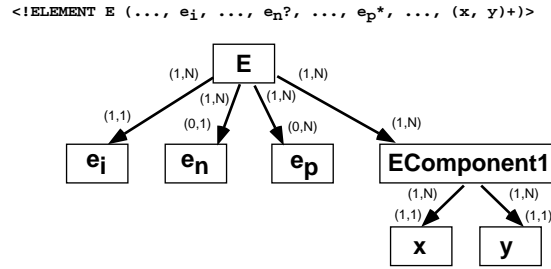


Figure 3: Conversion of a DTD element defined as a sequence

The *inheritance semantics* heuristics is tested in rule 5 to verify if a simple component e is a specialization of E . If so, an inheritance relationship is generated.

Rule 4 (Composite Element Conversion) A composite element E generates a non-lexical concept $\langle EName, 'NL' \rangle$, where $EName$ is the element name, if E is not already defined as a non-lexical concept. If E is already defined as a lexical concept l , l is removed.

Rule 5 (Simple Component Element Conversion) A simple component element e , with name $eName$, generates:

1. a lexical concept $\langle eName, string^3 \rangle$, if e is not already defined as a non-lexical or lexical concept;

³default L data type.

2. If $(EName, eName)$ has *inheritance semantics*, then generates an inheritance relationship $ir = \langle EName, \{eName\} \rangle$, if $\neg \exists ir' \in IR (ir'.name_s = ir.name_s)$. Otherwise, $ir.targetConcepts \leftarrow ir.targetConcepts \cup \{eName\}$;
3. If $(EName, e_iName)$ has no *inheritance semantics*, then generates a composition relationship $cr = \langle EName, eName, cardinality_s, (1,N)^4, null \rangle$, where $cr.cardinality_s$ is defined according to the rules 1 or 2, depending if E is defined as a composite element or a repeatable composite element.

Rule 6 (Repeated Simple Component Element Conversion) A repeated simple component element e generates:

1. a lexical concept $\langle eName, string \rangle$, if e is not already defined as a non-lexical or lexical concept;
2. a composition relationship $cr = \langle EName, eName, cardinality_s, (1,N), null \rangle$ where $cr.cardinality_s = (minCardinality_s, maxCardinality_s)$ is defined as follows:
 - (a) For each occurrence e' of e_i (p occurrences), apply rule 1 to define the source cardinality of the relationship between E and e' with the form $\langle minCardinality_{e'sc}, maxCardinality_{e'sc} \rangle$;
 - (b) $cr.cardinality_s$ is:
 - i. $minCardinality_s = \sum_{i=1}^p minCardinality_{e'sc}$;
 - ii. $maxCardinality_s = N$, if $\exists e' (maxCardinality_{e'sc} = N)$; otherwise, $maxCardinality_s = \sum_{i=1}^p maxCardinality_{e'sc}$.

Rule 7 (Complex Component Element Conversion) A complex component element e generates:

1. a virtual non-lexical concept $vnl = \langle name_{nl}, 'VNL' \rangle$ so that $vnl.name_{nl} = EName + "Component" + sequentialNumber$, where $sequentialNumber$ is a sequential that starts with 1 and it is increased step one for each complex component of E ;
2. a composition relationship $cr = \langle EName, name_{vnl}, cardinality_s, (1,N), null \rangle$, where $cr.cardinality_s$ is defined according to the rules 1 or 2.

Once generated, vnl is consider a composite element E where $EName = name_{nl}$, and the component element conversion rules are applied to its components.

Rule 8 (Lexical Component Conversion) A lexical component generates:

1. A lexical concept $l = \langle name_l, string \rangle$ where $l.name_l = EName + "Text"$;
2. A composition relationship $cr = \langle EName, name_l, cardinality_s, (1,N), null \rangle$ where $cr.cardinality_s$ is defined according to rules 1 or 2.

Rules 9 and 10 provide the conversion of a component element e that belongs to a composite element E defined as a DTD choice.

Rule 9 states that the relationship between E and e must match with one of the following mutual exclusion semantics (figure 4):

⁴default target cardinality.

- *Inheritance semantics*: if $(EName, eName)$ has *inheritance semantics*, then an inheritance relationship is generated;
- *Enumeration semantics*: if e has *enumeration semantics*, then e is a constant value associated to E . In this case, e is converted to an enumerated lexical concept and a composition relationship is generated;
- *Disjoint association*: default semantics if the previous ones do not match. In this case, a composition relationship is generated.

Besides, if not all component have inheritance semantics, the generated composition relationships must be disjoint and a disjoint relationship is generated as consequence.

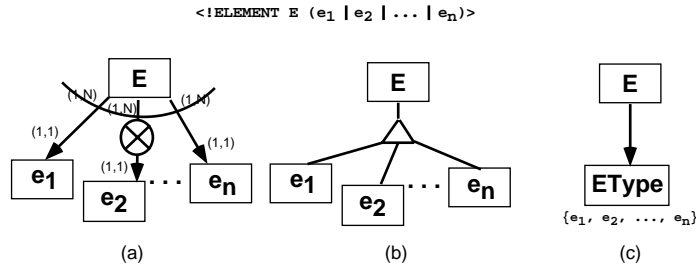


Figure 4: Semantic interpretations of a DTD element defined as a choice

Rule 9 (Choice Conversion) A composite element E defined as a DTD choice generates:

1. *Step 1* (inheritance and/or composition relationships definitions):
 - (a) Considering that $\{x_1, \dots, x_n\}$ is the set of all simple component elements of E so that $(EName, x_iName)$, $x_i \in \{x_1, \dots, x_n\}$, has inheritance semantics, generate an inheritance relationship $\langle EName, 'IR', \{x_1Name, \dots, x_nName\} \rangle$;
 - (b) Considering that $\{y_1, \dots, y_m\}$ is the set of all simple component elements of E so that $y_j \in \{y_1, \dots, y_m\}$ has enumeration semantics, generates, for each y_j :
 - i. an enumerated lexical concept $\langle name_l, string, \{y_1Name, \dots, y_mName\} \rangle$, where $name_l = EName + "Type"$;
 - ii. a composition relationship $cr = \langle EName, name_l, cardinality_s, (1, N), null \rangle$, where $cr.cardinality_s$ is defined as follows:
 - A. for each $y_j \in \{y_1, \dots, y_m\}$ apply rule 1 to define the source cardinality $y_j.sc$ of the relationship between E and y_j . Call Sc the set of all y_j source cardinalities $y_j.sc$;
 - B. apply rule 3 on Sc to obtain $cr.cardinality_s$.
 - (c) Considering that $\{z_1, \dots, z_p\}$ is the set of all simple component elements of E so that $z_i \in \{z_1, \dots, z_p\}$ and $z_i \cap \{x_1, \dots, x_n\} \cup \{y_1, \dots, y_m\} = \emptyset$, apply rule 3 for each z_i ;

- (d) Apply rule 8 for each lexical component of E ;
 - (e) Apply rule 7 for each complex component element of E .
2. *Step 2* (disjoint relationship definition): if two or more composition relationships were generated in step 1, generates a disjoint relationship $dr = \langle EName, 'DR', targetConcepts \rangle$, so that:
- (a) for each generated composition relationship cr in step 1, includes an element $\langle cr.name_t \rangle$ in $dr.targetConcepts$;
 - (b) if an inheritance relationship ir was generated in step 1, includes, for each element $name_t \in ir.targetConcepts$, an element $\langle name_t \rangle$ in $dr.targetConcepts$.

The conversion of repeatable choices, treated by rule 10, consider that inheritance and disjoint relationships are never generated because an inheritance relationship is supposed to be always one-to-one and, instead, more than one component element may be related to E as a consequence of the allowable repetition.

Rule 10 (Repeatable Choice Conversion) A repeatable composite element E defined as a DTD choice generates:

1. Considering that $\{y_1, \dots, y_m\}$ is the set of all simple component elements of E so that $y_j \in \{y_1, \dots, y_m\}$ has enumeration semantics, generates, for each y_j :
 - (a) an enumerated L concept $\langle name_e, string, \{y_1Name, \dots, y_mName\} \rangle$, where $name_e = EName + "Type"$;
 - (b) a CR $\langle EName, name_e, (0,N), (1,N), null \rangle$
2. Considering that $\{z_1, \dots, z_p\}$ is the set of all simple component elements of E so that $z_i \in \{z_1, \dots, z_p\}$ and $z_i \cap \{y_1, \dots, y_m\}$, apply rule 5 for each z_i ;
3. Apply rule 8 for each lexical component of E ;
4. Apply rule 6 for each complex component element of E .

Rule 11 deals with simple elements, generating a correspondent lexical concept.

Rule 11 (Simple Element Conversion) A simple element e generates a lexical concept $\langle eName, string \rangle$, if e is not already defined as a lexical concept.

4.1.3 Attribute rule

A single rule deals with attribute conversion, generating a correspondent lexical concept for each attribute. An enumerated lexical concept may be generated, depending on the value constraint associated to the attribute. Equal names for attributes and elements in the conceptual schema are treated through the generation of special names.

Rule 12 (Attribute Conversion) Given a set of attributes $\{a_1, \dots, a_n\}$, $n \geq 1$, defined to an element E :

1. Apply rule 4 to generate a non-lexical concept for E ;
2. For each $a_i \in \{a_1, \dots, a_n\}$ with name a_iName :
 - (a) Generates:

- i. An enumerated lexical concept $\langle a_iName, string, V \rangle$, where $V = \{v_1, \dots, v_t\}$, $t \geq 1$, if a_i has a constraint that defines an enumeration $\{v_1, \dots, v_t\}$ of permitted values; or $V = \{v\}$ if a_i value is constrained a constant value;
 - ii. A lexical concept $\langle a_iName, string \rangle$, otherwise. If there is a defined lexical concept l so that $l.name_l = a_iName$, then defines $a_iName = a_iName + EName$;
- (b) Generates a composition relationship $cr = \langle eName, a_iName, cardinality_s, (l, N), null \rangle$, where $cr.cardinality_s$ is:
- i. (1,1), if it is defined that a_i is a required attribute;
 - ii. (0,1), otherwise.

4.2 Restructuring Rules

Four rules comprises this class. Rule 13 performs the identification of non-lexical concepts that are *root concepts* and their proper conversion. A concept nl is considered root in an XCM schema if nl is not target concept of composition relationships.

Rule 13 (Root Concept Generation) Given a non-lexical concept nl , so that:
 $\neg \exists cr \in CR (cr.name_t = nl.name_{nl}) \wedge \neg \exists dr \in \{DR\} (\exists tc \in dr.targetConcepts (tc.name_t = nl.name_{nl})) \wedge \neg \exists ir \in \{IR\} (\exists sc \in ir.targetConcepts (sc.name_t = nl.name_{nl}))$
holds, then $nl.type_{nl} \leftarrow 'RNL'$.

The following rules perform simplifications on the XCM schema. Rule 14 states that A non-lexical concept nl (not root concept) that is composed by only one concept c is, in fact, an unnecessary intermediate concept in the chain of composition relationships $nl' \rightarrow nl \rightarrow c$. Such chain may be reduced to a single relationship $r nl' \rightarrow c$ where r is named nl .

Rule 14 (Named Relationship Generation) Given a concept $nl \in NL$, if:
 $nl.type_{nl} \neq 'RNL' \wedge \exists cr \in CR (cr.name_s = nl.name_{nl} \wedge \neg \exists cr' \in CR (cr' \neq cr \wedge cr'.name_s = nl.name_{nl}))$, then:

1. $\forall nl' \in NL (\exists cr'' \in CR (nl'.name_{nl} = cr''.name_s \wedge cr''.name_t = nl.name_{nl}))$ do:
 - (a) $cr''.name_t \leftarrow cr.name_t$;
 - (b) $cr''.name_{cr} \leftarrow nl.name_{nl}$;
 - (c) apply rule 3 to the set $\{ cr''.cardinality_s, cr.cardinality_s \}$ to determine $cr''.cardinality_s$;
2. remove nl .

Rule 15 states that in an inheritance relationship with a source non-lexical concept nl and a set C of specialized concepts $\{c_1, \dots, c_n\}$, $n > 1$, if all element $c_i \in C$ is a source concept of a concept x in an composition relationship, then x becomes a direct component of nl (x is generalized).

Rule 15 (Relationship Generalization) Given an inheritance relationship ir , if:
 $\forall name_t \in ir.targetConcepts (\exists x \in NL \cup L (\exists cr \in CR (cr.name_s = name_t \wedge cr.name_t = x)))$, then:

1. generate a composition relationship $cr = \langle ir.name_s, x, cardinality_s, (1,N), null \rangle$, where $cr.cardinality_s$ is defined according to rule 3;
2. remove cr .

Rule 16 states that a virtual non-lexical concept vnl that is a component of a non-lexical concept nl , so that $cardinality_s$ of their composition relationship is (0,1) or (1,1), is an unnecessary intermediate concept in the chain of composition relationships that links nl to the set C of vnl components. In this case, vnl is removed and all elements of C become direct components of nl .

Rule 16 (Virtual Non-Lexical Concept Remotion) Given a virtual non-lexical concept vnl , if:

$\exists cr \in CR (cr.name_t = vnl.name_{nl} \wedge cr.cardinality_s.maxCardinality = 1)$, then:

1. $\forall cr' \in CR (cr'.name_s = vnl.name_{nl})$ do:
 - (a) generate a composition relationship $cr'' = \langle cr.name_s, cr'.name_t, cardinality_s, cr'.cardinality_t, cr'.name_{cr} \rangle$, applying rule 3 to the set $\{cr.cardinality_s, cr'.cardinality_s\}$ to determine $cr''.cardinality_s$;
 - (b) remove cr' ;
2. $\forall dr \in DR (dr.name_s = vnl.name_{nl})$ do $dr.name_s \leftarrow cr.name_s$;
3. remove vnl .

5 User Validation

The *user validation* step concerns to the intervention of a human expert with the purpose of validating the preliminary XCM schema generated after the execution of the conversion rules. Such validation accomplishes manual adjustments in order to obtain a conceptual schema semantically correct in terms of the considered domain.

Actually, the user can do any change in the XCM schema, since this change do not invalidate a conversion rule. In practice, the user validation points in a XCM schema are related to defaults assumed by the conversion rules. The most relevant ones are:

- *Target cardinalities of directed relationships*: the $cardinality_t$ of CRs and DRs may not reflect the semantics of the domain. For example, a preliminary $cardinality_t$ set to (1,N) as default, even after an analysis of XML instances that could not precise it, in fact, it is always (1,1) according to the domain requirements;
- *Data types of lexical concepts*: if the data type of a concept $l \in L$ does not match the real domain of l values, even after an analysis of XML instances, the user may set a more appropriated data type to it;
- *Names of directed relationships*: Names of DTD elements that become relationships' names in XCM, according to rule 14, may not be suitable to the considered domain. Suppose a concept *Author* with two relationships with a concept *Address* named *RAddress* and *WAddress*. If these relationships mean the residential address and the work address, respectively, it would be better that these relationships were changed to much expressive names like *HomeAddress* and *WorkAddress*;

- *Generated names of concepts*: the names of virtual non-lexical, enumerated lexical and lexical concepts corresponding to #PCDATA components of composite elements must also be changed to much expressive names. In fact, a virtual non-lexical concept called *Address*, for example, is better than one default name called *AuthorComponent1*, supposing a virtual non-lexical concept that abstracts an embedded sequence in *Author* element definition that contains data about his/her address;
- *Mutual exclusion semantics*: a disjoint association may be replaced by an inheritance relationship, or vice-versa. The most usual case is when an implied hypernym relationship is not detected by a lexical dictionary, becoming a disjoint relationship. In the *conference* domain, for example, if a *Submission* of an *Author* is called *Work*, the lexical dictionary could not infer that a *Paper* is a *Work*. Changing the mutual exclusion semantics requires the execution of some *triggers* to keep valid relationships' specifications in a XCM schema⁵.

It is foreseen an automated support to the user in this step through an user interface where a graphical representation of a XCM schema is presented, and some changes to concepts and relationships are allowed. Parts of the schema that really deserves validation, like some target cardinalities and names of generated concepts, are supposed to be graphically highlighted to take the user attention.

6 Exemplifying the DTD-to-XCM Conversion Process

The *DTD-to-XCM Conversion* process works as follows. Basically, the first step - *conversion rules application* - follows the DTD definition order, applying the appropriated rules based on the syntax for element or attribute definition. The preliminary generated XCM schema is input to the second optional step - *analysis of XML instances* - works on the specifications of directed relationships as well as L concepts to infer cardinalities and data types, respectively. So, the final XCM schema is obtained after a human expert validation provided by the last step - *user intervention*.

The conversion process is now exemplified to the DTD of figure 1 (a).

6.1 Step1: Conversion Rules Application

The rules are applied to some lines of the DTD, as described in figure 1 (a), and the resulting parts of the XCM schema are shown in formal and graphical notations.

DTD analysis - lines 1 and 2

The *conference* element becomes a non-lexical concept according to rule 4. As it is defined like a *sequence* of simple component elements, rule 5 generates the lexical concepts *author* and *program*, their relationships with *conference* as well as the relationships' cardinalities. Rule 12 convert the *conference* attributes *name*, *year* and *place* in associated lexical concepts (Figure 5).

DTD analysis - lines 3 to 5

Rule 4 converts the previous lexical concept *author* in a non-lexical concept. As *author* is also defined as a *sequence* of simple elements, rule 5 generates the lexical

⁵These triggers are not explained here for sake of paper space

```

<Conference, 'NL'>
<Author, string>
<Conference, Author, (1,N), (1,N), null>
<Program, string>
<Conference, Program, (1,1), (1,N), null>
<Name, string>
<Conference, Name, (1,1), (1,N), null>
<Year, string>
<Conference, Year, (0,1), (1,N), null>
<Place, string>
<Conference, Place, (0,1), (1,N), null>

```

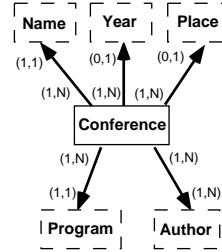


Figure 5: Conversion of the lines 1 and 2 of the DTD

concepts corresponding to its component elements and the associated relationships. Rule 11 confirms *name* and *eMail* as lexical concepts (Figure 6).

```

<Author, string>
<Author, 'NL'>
<Author, Name, (1,1), (1,N), null>
<eMail, string>
<Author, eMail, (0,N), (1,N), null>
<Submission, string>
<Author, Submission, (1,N), (1,N), null>
<Affiliation, string>
<Author, Affiliation, (1,1), (1,N), null>

```

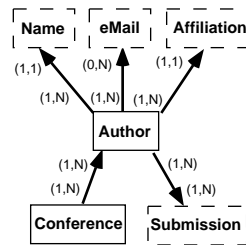


Figure 6: Conversion of the lines 3 to 5 of the DTD

DTD analysis - lines 6 to 14

Rule 4 converts *submission* to a non-lexical concept. Rule 5 generates the lexical concept *title* and its relationship with *submission*. Rule 7 generates a virtual non-lexical *submissionComponent1* to the choice component of *submission*. Rule 9 is applied to *submissionComponent1*: as an inheritance semantics is not considered to the pairs (*submissionComponent1*, *paper*) and (*submissionComponent1*, *tutorial*), a disjoint association semantics is applied, generating the lexical concepts *paper* and *tutorial* and a disjoint relationship between them and *submission* (Figure 7).

In the sequence, rule 11 confirms *title* as a lexical concept. Rule 4 convert *paper* and *tutorial* to non-lexical concepts. Rule 5 converts the sequence definition of *paper* in the lexical concepts *abstract* and *body* with their associated composition relationships with *paper*. Rule 11 confirm them as lexical concepts.

The *tutorial* element definition generates a lexical concept *duration* through the rule 5. As it also includes an embedded choice, a virtual non-lexical concept *TutorialComponent1* is generated according to rule 7. Such virtual non-lexical concept is treated by the rule 9: an enumeration semantics is detected for all simple components, resulting in an enumerated lexical concept *tutorialComponent1Type* (having {basic, advanced} as permitted values) related to *tutorialComponent1*.

DTD analysis - lines 15 to 17

Rule 4 converts the previous lexical concept *affiliation* to a non-lexical concept. Rule 9 is applied to the *affiliation* definition, generating a disjoint relationship with

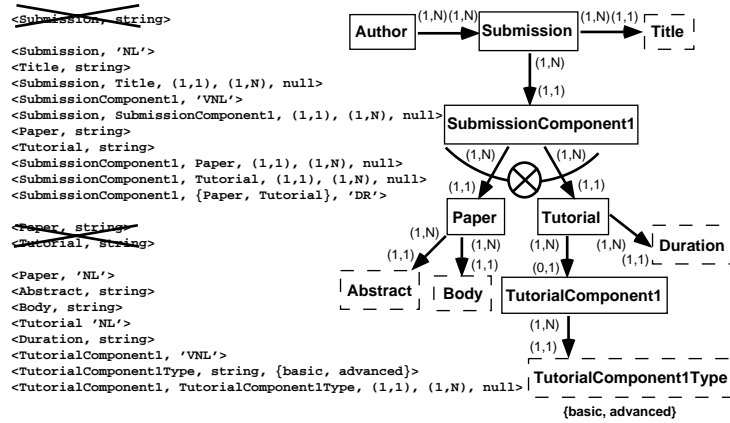


Figure 7: Conversion of the lines 6 to 14 of the DTD

the lexical concepts *institution* and *industry* (generated by rule 5) because an inheritance semantics is not considered. Rule 11 confirms *institution* and *industry* as lexical concepts (Figure??).

DTD analysis - lines 18 to 22

Rule 1 converts the previous lexical concept *program* to a non-lexical concept. Rule 10 is applied to the *program* definition: the lexical component (#PCDATA) generates a lexical concept *programText* through the rule 8, and the lexical concept *session* through the rule 5. The further defined element *session* is then converted to a non-lexical concept by rule 4, and the rule 5 is applied to each one of its components, generating the lexical concepts *subject*, *date* and *time*. Such concepts are confirmed as lexical through the rule 11 (Figure??).

Restructuring rules

Rule 13 converts *conference* in a root concept. Rule 16 removes the virtual non-lexical concepts *submissionComponent1* and *tutorialComponent1*, generating a direct association of *paper* and *tutorial* with *submission*, and *tutorialComponent1Type* with *tutorial*. The restructuring performed by this last rule is shown in figure??.

6.2 Step2: Analysis of XML Instances

As a set of XML instances related to the considered DTD is not presented, such step cannot be exemplified. However, it is possible to suppose that at least some data types of lexical concepts may be inferred, as *time* to *duration* concept, and *integer* to *year* concept.

6.3 Step3: User Intervention

At least three required points of user intervention must be taken into account, as follows.

Target cardinalities

Generated names of concepts

Mutual exclusion semantics

Concluding the step 3, the resulting XCM schema is the one shown in figure 1 (b).

7 Related Works

8 Conclusion