A DOCUMENT RULES DESCRIPTION LANGUAGE BASED ON FEATURE LOGIC FOR XML DOCUMENT EXCHANGE

Makoto Imamura[1], Yasuhiro Takayama[1], Yasuhiro Okada[1] and Norihisa Komoda[2]
[1] *Mitsubishi Electric Corporation,* [2]*Osaka University*

Keywords: Document Exchange, XML Validation, XML Transformation, Schema Description, Logic Programming.

Abstract: We propose a minimum Document Rules Description Language DRDL for XML validation and translation in web-based document exchange. DRDL has small syntax and simple semantics so that it can ease to reuse document rules. Furthermore, DRDL can describe mapping constraints from XML tree structure to table structure for web input-form generation. The technical feature of DRDL is to extend and reinterpret a Feature Logic, which has been used for representing linguistic knowledge, in order to allow existential and universal quantifiers over list-value to cope with XML. DRDL has already been applied to real systems such as elevator design support, Web-EDI, government-to-business and facility management.

1 INTRODUCTION

As XML formats have been widely adopted for a standard, interoperable document exchange format, XML validation and translation become common and critical component for document exchange both within and across enterprises. In developing web based XML document exchange systems, it is a problem how to efficiently make data validation function and table style form layout.

XML standard technologies, such as XML schema, XSLT and XQuery, have been developed to increase expressive power for document validation, transformation and query. However these standards are too huge to easily master and are too heavy to be used in some applications.

In this paper, we propose a simple language DRDL which has the following features to describe semantic constraints in document rules.

1. DRDL has small syntax and simple semantics so that it can ease to reuse document rules, such as input-form check for elevator specification or Web-EDI, by translating rules to other languages such as JavaScript.
2. DRDL can describe table mapping constraints from XML tree structure to table structure for web input-form.

The remainder of this paper is organized as follows: Section 2 presents the requirements for DRDL. Section 3 defines the syntax and semantics of DRDL. Section 4 presents an DRDL-based framework, which consists of an editor, rule translators and a processor. Section 5 discusses related works, while section 6 presents conclusions and open issues.

2 REQUIREMENTS

We present here the requirements of expressive power and the development productivity for DRDL.

2.1 Expressive Power

In the industrial and business document exchange systems, document validation and transformation are indispensable functions for smoothly processing the exchanged document data to prevent errors. DRDL needs to have sufficient expressive power to describe the following functions to treat cross-field, cross-field-structure cross-field-type and table mapping constraints, which the first version of XML Schema can not handle (Daum, 2003).

a) Cross-field constraint processing
To validate constraints across the contents of elements or attributes in XML documents. For example, to check that each element <A> is equal to the sum of each element <B> and <C> in their respective orders for multiple <A> elements.

Imamura M., Takayama Y., Okada Y. and Komoda N. (2007).
A DOCUMENT RULES DESCRIPTION LANGUAGE BASED ON FEATURE LOGIC FOR XML DOCUMENT EXCHANGE.
In *Proceedings of the Second International Conference on e-Business*, pages 71-77
DOI: 10.5220/0002111800710077

To assign the value to an element which has been computed for the contents of elements (e.g. the sum of multiple elements).

b) Cross-field-structure constraint processing
To validate constraints between the structure of an element and the content of an element in XML documents. For example, to validate whether the multiplicity of element <A> is equal to the content value of element <B>.

To change the structure of an element to satisfy a cross-field-structure constraint. For example, to add up the <A> elements such that the multiplicity of element <A> is equal to the content value of element <B>.

c) Cross-field-type constraint processing
To validate constraints between the type of an element and the content of an element.

To assign a value to element <A> depending on the type of the content of element <B>.

To validate if data in an XML document are consistent with those in RDB (Relational-Database).

d) Table Mapping Constraint Processing
To describe mapping constraints from XML tree structure to table structure for input form generation.

2.2 Development Productivity

In order to apply DRDL to developing industrial systems, the XML application development frameworks need to have the following requirements.

a) Development efficiency
Firstly, from the development efficiency point of view, the frameworks have the following requirements for easily revising document processing functions with schema upgrade:

1. Editability of document rules
Document rules need to be edited not only by the programmers in an information system division, but also by the end-users in a related business division.

2. Reusability of document rules
Document rules need to be easily shared among organizations participating in the document exchange, and to be utilized easily by each organization's business sub-systems.

3. Replacability of document rules
Document rules need to be easily replaced in the distributed sub-systems which use a common document format.

b) Connectivity to other functions
Document rules need to be processed with other functions in the application system. For example, rule validation functions should be able to be communicated with a document view controller in a client-side document input tool.

3 XML DOCUMENT RULES DESCRIPTION LANGUAGE

We define here DRDL which has the expressive power stated in 2.1. 3.1 presents design principles. 3.2 and 3.3 defines the syntax and the semantics of the core part of DRDL which is called XML constraint language (XCL). 3.4 presents DRDL as the extension of XCL with optional functions for application systems.

3.1 Design Principles

The main design principle of DRDL is to define a minimum language needed to describe semantic constraints in Web-form based XML document exchange. To define DRDL, we use the concept of unification grammar formalism (Shieber, 1986) which describes natural language syntactic and semantic constraints. Description in unification grammar consists of production rules described by context-free grammar, and feature constraints described by feature logics whose typical example is (Smolka, 1992). Production rules correspond to document structure definition described by DTD or XML schema, and feature constraints correspond to semantic constraints described by XSLT or XQuery. We extend the syntax of a feature logic and reinterpret the semantics of it in order to cope with XML.

Table 1: Analogy with feature logic and XML processing.

Feature logic	XML processing	
	Validation	Transformation
path in feature structure /	location path in XML structure	
and $\wedge$	and	sequential statement
implication $\Rightarrow$	implication	if statement
universal quantification $\forall$	for-each statement	
existential quantification $\exists$	cardinality check of elements	addition or deletion of elements
equation =	equality check	unification (variable assignment)
sort :	type check	(no correspondence)
subsumption $\subseteq$	upper compatibility of schema	(no correspondence)

The main syntactical extension is to introduce quantifiers over list-value. Semantic reinterpretation is to introduce validation interpretation for XML document validation functions in 2.1, and assignment interpretation for XML document transformation realized by the assignment functions in 2.1.

A core idea of DRDL which will be stated in the following subsections is to use semantical analogy between a feature logic and XML processing in table 1.

3.2 Syntax of XCL

We define in this subsection the syntax of XCL, the core part of DRDL, and a term and a formula in XCL like those in a first-order language.

Definition: XCL term

An XCL term, which denotes a list of XML nodes, is defined as follows:

(1) A constant (a list of XML nodes) is an XCL term.
(2) A location path in XML is an XCL term.

Definition: XCL formula

An XCL formula is inductively defined as follows:

(1) XCL comparison formulae

If s iand t are XCL terms, then $s = t$, $s \neq t$, $s \leq t$, $s \geq t$, $s>t$ and $s<t$ are XCL formulae.

(2) XCL type-constraint formula

If s is an XCL term and t is a data type, then s:t is an XCL formula.

(3) XCL logical formulae

If F and G are XCL formula, then $F \wedge G$, $F \vee G$, $\neg F$, and $F \Rightarrow G$ are XCL formulae.

(4) universally quantified XCL formula

If x is a variable, p is a location path, and F is a XCL formula, then $\forall x \in p$. F is an XCL formula.

(5) existentially quantified XCL formula

If x is a variable, p is a location path, n is a non-negative integer or an expression whose evaluated value is a non-negative integer, and F is XCL formula, then $\exists x \in p$ s.t. (count(.)=n).F is an XCL formula. In (count(.)=n), the symbol "=" may be $\neq$, $\leq$, $\geq$ or > and the symbol "n" may be an arithmetic expression.

3.3 Semantics of XCL

We present the operational semantics of each formula stated in section 3.2.

1. XCL Comparison Formulae

"s = t" is evaluated as follows in validation interpretation. If the evaluation value of "s" is equal to the evaluation value of "t" in the sense of list, then it is "true". Otherwise it is "false".

"s=t" is evaluated as follows in assignment interpretation. An evaluation value of "t" is assigned to the position that is addressed by a localtion path "s". This evaluation procedure corresponds to a DOM-API "set node value" which assigns a value to a node in a DOM tree.

2. XCL type-constraint formula

"s: t" is evaluated as follows in validation interpretation. If the evaluation value of s belongs to data type t, then it is true. Otherwise, it is false.

The evaluation of "s: t" is not defined in assignment interpretation.

3. XCL Logical Formula

XCL logical formulae in validation interpretation are evaluated as those in propositional logic. XCL logical formulae in assignment interpretation are evaluated as follows: For $F \wedge G$, F is firstly evaluated and secondly G is evaluated; For $F \vee G$, only F is evaluated; For $\neg F$, it is reduced to the normal form F', in which all $\neg$ s are attached to comparison formulae by rewriting with tautology in first-order predicate logic, and then F' is evaluated; For $F \Rightarrow G$, if the evaluated value of F in the validation interpretation is true, G is evaluated. This evaluation procedure corresponds to the "if" statement in XSLT.

4. Universally quantified XCL Fformula

In respect of validation interpretation, $\forall x \in p$. F is evaluated as follows. Firstly, x in formula F is substituted by each element of a list value, which is obtained by evaluating a location path p, and we can then obtain formulae Gs which are the variants of F. The number of formulae Gs is equal to the cardinality of list values of a location path p. If all the formulae Gs are evaluated and all the values of those are true, the formula F is true; otherwise, it is false.

In respect of assignment interpretation, $\forall x \in p$. F is evaluated as follows. Formulae Gs as the variants of F are obtained by the substitution of x in F in the same way as that used in validation interpretation. Each formula G is evaluated in the same order as that in the list value of p. The order of an element in the list value of p is naturally defined by the order of a value in an XML document. This evaluation procedure corresponds to the "for-each" statement in XSLT.

5. Existentially quantified XCL formula

In respect of validation interpretation, $\exists x \in p$ s.t.(count(.)=n). F is evaluated as follows. Formulae Gs as variants of F are obtained by the substitution of x in F in the same way as $\forall x \in p$. F. If the number of these formulae is equal to the evaluation value of "n"

and at least one of the evaluation values of G is true, then it is true. Otherwise it is false. This evaluation procedure corresponds to minOccurs and maxOccurs in XML Schema, which check the multiplicity of elements. Quantified XCL formulae are much more powerful than those in XML schema, because "n" can be an arithmetic expression.

In respect of assignment interpretation, $\exists$ x $\in$ p s.t.(count(.)=n).F is evaluated as follows. XML elements are added or deleted until the number of nodes, which are addressed by a location path p, is equal to the evaluation value of "n". Formulae Gs as variants of F are obtained by the substitution of x in F in the same way as that for $\forall$ x $\in$ p. F. Only the first formula in Gs is evaluated. This evaluation procedure corresponds to an instruction for node operation in DOM such as "appendChild" or "removeChild".

3.4 Examples

We present in this subsection examples of the cross-field, cross-field-structure and cross-field-type constraints described in 2.1.

1. Cross-field Constraint
The formula shown in figure 1 describes a constraint whereby, for each <p-list>, the <total> is equal to the <sum> of each <price> in <product>s. An example XML document is shown in fig. 2. Validation interpretation achieves a validation function for XML documents, while assignment interpretation achieves a computation function among cells in a table like a spread-sheet software.

$\forall$ x $\in$ /root/p-list. x/total = sum(x/product/price)

Figure 1: Cross-field Constraint with DRDL.

```
<root>  <p-list date = "2002-10-01">
            <product><name>PC</name>
                    <price>950</price></product>
            <product><name>Disk</name>
                    <price>500</price></product>
            <total>1450 </total>  </p-list>
        <p-list date = "2002-11-01">.... </p-list> </root>
```

Figure 2: An Input XML Document.

2. Cross-field-structure Constraint
The formula shown in fig. 3 describes a constraint in which a multiplicity of <device>s is equal to the content of <device-number>. Fig. 4 shows an example of an XML document as the presumed input. Validation interpretation achieves a cardinality checking function for the XML elements. Assignment interpretation achieves addition and deletion operations on XML elements which correspond to DOM APIs such as "appendChild" and "removeChild".

An assignment interpretation of the formula in figure 3 for the input XML document in figure 4 is as follows: Three <device> elements are added, consequently the multiplicity of element <device> is equal to the content of an element <device-number> which is 5.

$\exists$ x $\in$ /device-list/device s.t.(count(.) = /device-number).

Figure 3: Cross-field Constraint with DRDL.

```
<device-number> 5 </device-number>
<device-list><device> pencil </device>
  <device> pencil sharpener </device></device-list>
```

Figure 4: An Input XML Document.

3. Cross-field-type constraint
An XCL type–constraint formula can define the data-type of an element depending on the content of another element, which cannot be described by an XML Schema. The formula in fig. 5 describes a constraint whereby, for each <product>, if the <category> in the <product> is "communication-facility", the data type of the <number> of the product is "communication-facility code".

$\forall$ x $\in$ //product. (x/category = "communication-facility" $\Rightarrow$ (x/number: "communication-facility-code")

Figure 5: Cross-field-type Constraint with DRDL.

3.5 DRDL

DRDL is an extended XCL which has the following optional descriptions.
(1) Table mapping constraint (stated in 2.1d))
(2) RDB mapping constraint
(3) Vector expression constraint

In this subsection, we describe a table mapping constraint which is the main feature of DRDL. Table mapping constraint consists of a matrix constraint and a connection constraint. Matrix constraint describes the mapping between a location path in XML documents and the row and column of a table. A connection constraint describes the condition for tables to be connected with those of rows or columns.

For example, the constraints in fig. 6 describe the table mapping denoted by fig. 7. (1) and (2) in fig.6 describe mapping constraints from the left part of doc1 to the left part of the table (tbl1). In a similar way, (3) and (4) in fig.6 describes mapping

constraints for the right part of doc1 and tbl2. (5) and (6) in fig.6 describe constraints to connect the left and the right part of the table. Connection constraints can describe dependency between the number of the column or row of a table and that of the other table.

(a) Matrix constraint

tbl1 ! **row**	= doc1 ! a1/b	(1)
tbl1 ! **column**	= doc1 ! a1/b/c	(2)
tbl2 ! **row**	= doc1 ! a2/bb	(3)
tbl2 ! **column**	= doc1 ! [a2/bb/e, a2/bb/f]	(4)

(b) Connection constraint

count(a1/b) = count (a2/bb)	(5)
horizontal_connect ([tbl1, tbl2])	(6)

Figure 6: Table mapping Constraints with DRDL.

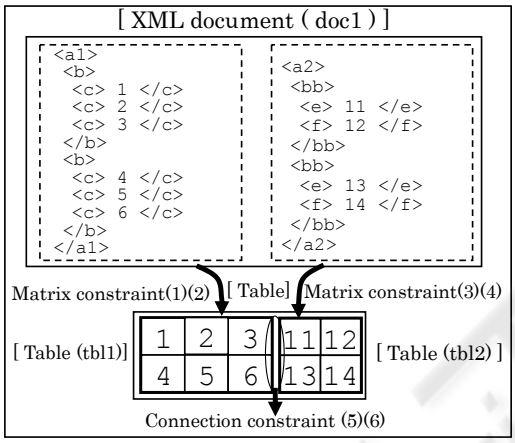

Figure 7: Table Mapping Example.

4 XML APPLICATION DEVELOPMENT FRAMEWORK

a) Architecture of the DRDL Framework

DRDL Framework consists of a rule file, an editor, a processor and a rule translator. Figure 8 shows the whole structure of DRDL Framework.

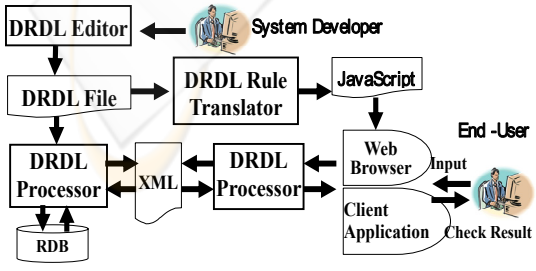

Figure 8: Architecture of DRDL Framework.

1. DRDL Editor

In order to satisfy the requirement for "editability of document rules (2.2a)1)", we have developed a DRDL editor for end-users who do not have specific IT knowledge to enable them to make DRDL formulae. The DRDL editor has a table-like GUI interface.

The left part of this GUI is for editing the document schema, and the right part is for editing the document content constraint or the XML-RDB mapping constraint.

2. DRDL Rule Translator

To fulfill the requirement for "reusability of document rules (2.2a)2)", we have developed DRDL rule translators which transform DRDL formulae to another programs or scripts for other XML processors. For example, a translator can generate JavaScript for web input forms from DRDL formulae. These JavaScript are used for input checking in our XSLT based XML input form (Imamura et.al., 2005).

3. DRDL File

To fulfill the required "replacability of document rules (2.2 a) 3)", we have introduced DRDL files to describe document rules independently from application programs. Sending and replacing DRDL allows client XML tools to replace the document processing function and server systems to replace the XML transformation function.

4. DRDL Rule Processor

To fulfill the requirement for "connectivity to other functions(2.2b))", we have developed a DRDL processor which can cooperatively process XML documents sharing a DOM tree with other processes.

b) Application Systems of DRDL

The first version of DRDL framework was built in 1999 and has been applied to the following systems (Imamura et.al., 2000). The number between parentheses denotes the start year of the system operation.

(1) Elevator design support system (1999)
(2) G2B document-exchange system (2000)
(3) Web EDI system (2001)
(4) Facility management system (2002)

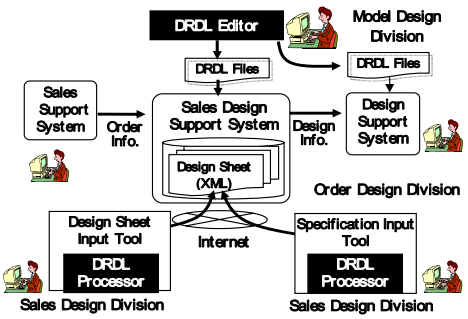

Figure 9: Elevator Design Support System.

For example, we show the elevator design support system in Figure 9. Elevator design sheets are sent from the sales design divisions to the order design divisions. DRDL processors are used to check whether the elevator design sheets satisfy the document rules stipulated in the input conditions for the design support systems. The distinct number of XML elements for each design sheets is about 300, and the total number of check rules is about 900. DRDL is used effectively to describe table mapping constraints depending on the number of floors and lifts in the building. The DRDL framework decreased 50 % of the time for creating Web-input forms compared with an existing form builder.

5 RELATED WORKS

Various schema languages have been proposed in order to describe document constraints stated in 2.1 without hard-coding (Murata et al., 2005). Famous ones are Schematron (Jelliffe, 2004) and DSD (Klarlund et al., 2002) as a pattern-based language. DRDL is different form them in logic-based. DRDL is expected easy to translate semantic constraint to other languages by referring logic-based existing formal specification works.

XSLT is a standard and popular language for XML transformation. However its specification is too huge to easily master, so there are some tools to support XSLT generation (Tang et al., 2001). Table mapping constraint description in DRDL is an approach to specify the mapping pattern from XML tree structure to table structure. In fact, DRDL has a translation function from table mapping constraints in DRDL to XSLT descriptions.

Recently, in the database community, declarative and unifying approaches with document schema and database query language have been proposed for data integration and exchange. The BEA AquaLogic Data Service Platform adopts data integration framework based on XML Schema and XQuery to modelling and accessing the variety of data source types such as relational, Web service, function-based and file-based (Reveliotis et al., 2006). In this paper, we discussed the data integration and exchange from a document processing and knowledge representation point of view.

6 CONCLUSIONS

We have proposed a minimum semantic constraint description language DRDL which satisfies the XML document-processing requirements of expressive power, development efficiency and connectivity to other functions needed in real application systems. Interpretation of the DRDL formula realizes a validation function for XML documents. Assignment interpretation of DRDL formulae realize operations on DOM trees, such as addition, deletion and value-assignment like DOM-API, and also control structures, such as the "if " statement in imperative languages and the "for-each" statement in XSLT. Furthermore, DRDL provide table mapping constraint for Web-input form generation.

Open issues are the following.

(1) Logic programming with terms as XML elements We have treated the assignment interpretation in a way analogous to popular imperative languages such as Java or C for usability and performance. From the theoretical point of view, however, it is important that a determination procedure for the satisfiability of XCL formulae should give a unification algorithm between XCL formulae (Smoka, 1992). Replacing a term in Prolog with an XCL formula allows us to obtain a new constraint logic programming (Jaffar et al., 1994)(Mukai, 1991). This logic programming language is an alternative XML transformation language to XSLT.

(2) Decision procedure for satisfiability of subsume relation of XCL formulae.

(3) Comparison with other declarative XML constraint description languages such as Xcerpt (Schaffert, 2004) and Relational.OWL (Laborda et.al., 2005).

REFERENCES

Daum,B. 2003, Validation beyond XML Schema, *Modelling Business Objects with XML Schema*, Morgan Kaufmann Publishers, pp.323 -362.

Imamura, M. et al. 2000. Metadata representation for Internet-based XML application from business to government, *in proc. of 7th International Conference on Parallel and Distributed Systems Workshops,* pp.387-392.

Imamura, M. et al. 2005. An XML Input Form Generation Method Based on a Tree-table mapping Model, *Transactions of Information Processing Society of Japan, Vol.46 No.12,* pp3066-3077.

Jaffar,J. and Maher,M.J. 1994. Constraint Logic Programming: A Survey, *Journal of Logic Programming* 19, 20 pp.503-581.

Jelliffe, R., 2004. The Schematron, http://xml.ascc.net/schematron/

Klarlund, N., Anders Møller,A., Schwartzbach ,M.I. 2002, The DSD Schema Language, *Automated Software Engineering* 9 3, 285-319

Laborda, C. P. and Conrad, S. 2005. Relational.OWL, in *proc. of Second Asia-Pacific Conference on Conceptual Modelling (APCCM2005)*, pp.89-96.

Mukai, K. 1991. Constraint Logic Programming and the Unification of Information, *Doctoral Dissertation, Dept. of Computer Science*, Faculty of Engineering, Tokyo.

Murata, M., Lee, D., Mani, M., Kawaguchi,K, 2005. Taxonomy of XML schema languages using formal language theory, *ACM Transactions on Internet Technology (TOIT) Volume 5 ,Issue 4,* pp.660-704

Reveliotis, P., Carey, M., 2006. Your Enterprise on XQuery and XML Schema: XML-based Data and Metadata Integration, in *proc. of 22nd International Conference on Data Engineering Workshops*, pp80 – 89.

Schaffert,S. 2004, Xcerpt: A Rule-based Query and Transformation Language for the Web, http://edoc.ub.uni-muenchen.de/archive/

Shieber, S. M. 1986. *An Introduction to Unification-based Approaches to Grammar*, CSLI Lecture Notes Number4, Stanford University

Smolka, G. 1992. Feature Constraint Logics for Unification Grammars, *Journal of Logic Programming*, New York, PP51-87.

Tang,X. and Tompa,F.W. 2001. Specifying Transformations for Structured Documents, *Proc. of the 4th International Workshop on the Web and Databases,* pp.67-72