

# An Approach for Semi-Automatic Derivation of XSLT Information Based on DTD Descriptions

Martin Endig, Thomas Herstel, Eike Schallehn, and Zoltan Sera

Otto-von-Guericke-University Magdeburg  
Department of Computer Science  
P.O. Box 4120, D-39016 Magdeburg, Germany  
{endig|herstel|schallehn|sera}@iti.cs.uni-magdeburg.de

**Abstract.** Today, the integration of heterogenous information systems is subject of current research work, mainly using mediator or adapter based approaches. The implementation of adapters is a complex task, and providing generic adapters for special classes of sources is a practicable approach. We developed a generic XML adapter for accessing data in XML format. In order to instantiate this adapter configuration information is required. In this paper we suggest a first approach for the semi-automatic derivation of such information based on XSLT. As a result tool support for the creation of XSLT information can be supplied.

## 1 Motivation

It is not conceivable that the application of standards like XML [3] will solve all problems of conflicting representations of data in one or several domains. Therefore, support for the transformation of XML data is required. The XSLT language available today provides powerful means to transform XML documents, but the numerous language features make the design of this transformation information a complex task [4]. To support this design step we propose concepts for a semi-automatic derivation of XSLT information based on an analysis of DTDs and the features provided by XSLT. The intention is to provide tool support for creating the information from given source and target DTDs. Our research is motivated by a project from the field of digital libraries, where we apply XML to integrate various sources of bibliographical information, containing bibliographical meta data about publications from various fields. Currently the development of adapters is subject of a number of research projects, whereby the implementation of one adapter for all kinds of data sources is unconceivable. Rather, the preparation of generic adapters for certain classes of data sources is more practicable. In this case, the access to a given data source is possible through a previous instantiation process of a special adapter. For this process, configuration data about the special source are necessary. In the project “Federation Services for Heterogenous Document Sources” we developed a generic adapter for accessing

---

\* This research was partially supported by the BMBF (08SFB031) and DFG (FOR345/1-1)

data, which are stored in various XML formats. For cooperative data providers the specific DTD is known. Because of the autonomy of data sources the processing of XML data requires a transformation into an adapter understandable format at first. In the context of XML the usage of the standard XSLT is one solution for providing this mapping information. But currently the complex XSLT information has to be created manually. In this paper we present an approach for semi-automatic derivation of XSLT information based on source and target DTDs, whereby a user can separately assign the nodes from the source to nodes of the target DTD. According to this basic approach a tool support for the creation of XSLT information, without requiring user knowledge of the structure of commands or the concrete usage of XSLT, is provided.

## 2 Requirements Analysis

For the transformation of XML documents using XSLT different possibilities must be considered. XML concepts like namespace mechanism, processing instructions and comment nodes are not discussed in the remainder of this paper, because they are mostly irrelevant for the real contents of documents. This means, we consider only element, attribute and text node transformations, whereby each node has a special name and a value. Starting with this assumption, we first can specify some elementary transformations based on XSLT:

**Identity Transformation:** A node  $n_s$  according to the source DTD is copied to exactly the same node  $n_t$  of the target DTD. The name and the value of  $n_s$  is equal to the name and value of  $n_t$ . Contained nodes, including all attribute nodes in the case of an element node, are copied, too.

**Node Renaming:** A node  $n_s$  according to the source DTD is renamed to a node  $n_t$  from the target DTD. The value of  $n_s$  is copied to the value of  $n_t$ .

**Type Conserving Node Transformation:** An element or attribute node  $n_s$  according to the source DTD is copied to an element or attribute node  $n_t$  of the target DTD. The name and the value of  $n_s$  is associated with the name and the value of  $n_t$ .

**Concatenation of Node Values with Same Type:** The values of element nodes  $n_{s_1}$  to  $n_{s_n}$  with the same name according to the source DTD are linked to one element node  $n_t$  of the target DTD. The name of  $n_t$  is arbitrary and the value is equivalent to the linked values. In the same manner, the concatenation of element nodes with different names in the source DTD to one element node in the target DTD is possible, too. In the case of attribute nodes, the same operation can be used with the restriction that the nodes are all contained in the same element node.

**Concatenation of Node Values with Different Type:** The values of element and attribute nodes  $n_{s_1}$  to  $n_{s_n}$  with different names according to the source DTD are merged to one element or attribute node  $n_t$  of the target DTD. The value of  $n_t$  is equivalent to the linked values and the name is arbitrary. The concatenation of selected element or attribute nodes from the source DTD is a special case of this operation.

**Splitting of Node Values:** The value of an element or attribute node  $n_s$  according to the source DTD is split into several element or attribute nodes  $n_{s_1}$  to  $n_{s_n}$  of the target DTD. The values  $n_{s_i}$  are equivalent to the results of splitting. The names of  $n_{s_i}$  are equal for all nodes in the case of element nodes and different in the case of attribute nodes.

**Structure Transformation:** For structure transformation two different operations must be distinguished: operations for the introduction or removal of either a complete node layer or for a single node. In the case of modifying a node layer, a node layer will be introduced or removed in the target document. This is only possible for element nodes. Furthermore, the introduction or removal of attribute nodes from the source document is possible.

**Non Type Conserving Node Transformation:** Contrary to type conserving node transformation, with this operation the transformation of node names to node values can be realized.

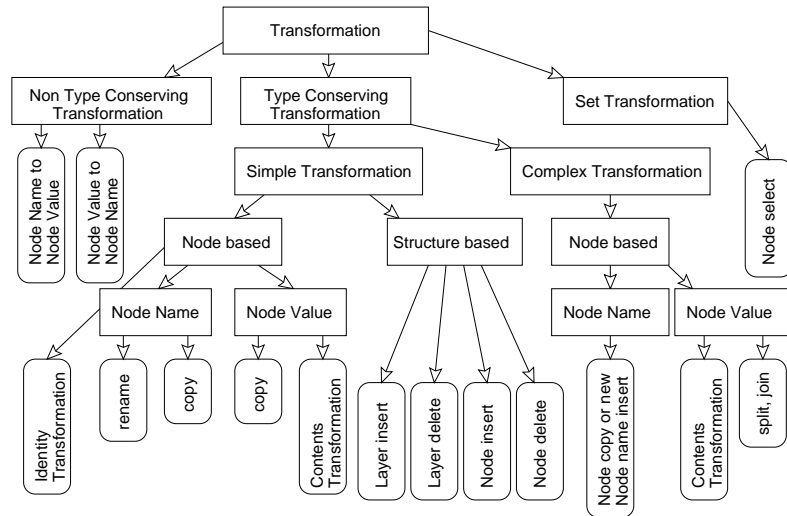
**Set Transformations:** Of a set of nodes in the source document certain nodes are transformed to the target DTD. For this selection in the source a special condition depending on either the value or the name of nodes can be specified.

For these elementary transformations a classification of XSLT operations can be specified. This classification is given in figure 1. The operations can be subdivided in operations for non type conserving transformation, like transformation from node value to node name and vice versa, type conserving transformation and set transformation. In the class of type conserving transformation, a classification into simple and complex transformation is possible, whereby the operations in the class simple transformation are node based or structure based. The class structure based transformation comprises all transformations like introduction or removal of node layers or single nodes. Node based operations are identity transformation, node name or node value transformation. For the last two classes the operations for renaming, copying and contents transformation are provided.

### 3 Transformation Problems

The usage of XSLT raises two problems, an “ID” and a “cardinality” problem. The last problem occurring only for transformation of element and attribute nodes. In general, an attribute node can be defined as unique with regard to the values. During the transformation of such attributes, the problem of transformation from non “ID” attribute to “ID” attribute can occur. We refer to this as the “ID” problem. Furthermore, the cardinality range of elements and attributes must be defined. While for elements the cardinality “optional” (?), “exactly once” ( $e$ ), “zero or more” ( $*$ ) and “one or more” ( $+$ ) are valid, for attributes “implied” and “required” can be differentiated. For a transformation the concrete cardinality must be considered, so that no violations occur. This problem will be identified as “cardinality” problem.

The solution of the “ID” problem includes the consideration of four different cases. Only one of these is non-trivial: the transformation of an attribute



**Fig. 1.** Classification of XSLT Operations

declared as non “ID” to an “ID” attribute, violating the “ID” property. The introduction of an automatic assignment of valid values is a possible solution for this problem. By considering all possible “cardinality” assignments from element nodes in the source to element nodes in the target DTD a solution of the “cardinality” problem for element nodes can be given. The number of relevant combinations, which are essential to consider for a solution, can be reduced, if the trivial assignments (cp.  $e \rightarrow e$ ) are omitted. If the cardinality of element nodes in the source is a subset of the cardinality in the target DTD, then all instances of elements can be carried over to a target document. This way all assignments with the aim  $*$  and the starting  $e$  can be omitted. As a result, for the solution of the “cardinality” problem regarding element nodes seven combinations have to be considered. Finally, in table 1 the concrete solutions are summarized.

In the same manner, the solution of the “cardinality” problem for attribute nodes is defined, considering four different cases. Three of these cases are trivial. The transformation from attribute with the cardinality “implied” to attribute with the cardinality “required” is a problem. If the attribute exists in a given instance, the transformation is possible, otherwise the introduction of an attribute with fixed value is required. In general, for the solution to all transformation problems XSLT provides corresponding features.

## 4 Transformation Techniques

The description of DTDs via the mathematical concepts of context-free grammars is suitable for providing the theoretical background for transformation techniques [6]. As our general intention is the derivation of XSLT information, the transformation problem can be summarized as follows: based on the source

**Table 1.** Solution for the transformation “cardinality” problem

No.	Source	Target	Comments
1	*	e	For the solution two different cases must be considered: <ul style="list-style-type: none"> <li>• <math>? \rightarrow e</math>: cp. No. 5</li> <li>• <math>+ \rightarrow e</math>: cp. No. 4</li> </ul>
2	*	?	For the solution two different cases must be considered: <ul style="list-style-type: none"> <li>• <math>? \rightarrow ?</math>: trivial</li> <li>• <math>+ \rightarrow ?</math>: cp. No. 6</li> </ul>
3	*	+	For the solution two different cases must be considered: <ul style="list-style-type: none"> <li>• <math>? \rightarrow +</math>: cp. No. 7</li> <li>• <math>+ \rightarrow +</math>: trivial</li> </ul>
4	+	e	For the solution the concatenation of elements from the source document to exactly one element in the target document is required.
5	?	e	The solution includes several aspects. In the case of an existing element contained in a given instance, the transformation in the target document is possible. If this is not the case, an empty element or an element with fixed values must be added to the target document.
6	+	?	The solution includes only the consideration of $+ \rightarrow e$ (cp. No. 4), because of the optional cardinality of elements in the target document.
7	?	+	For the solution the consideration of two different aspects are necessary: <ul style="list-style-type: none"> <li>• <math>e \rightarrow +</math>, if the element exists in the source document: trivial</li> <li>• If the element does not exist, then an empty element or an element with fixed values must be added to the target.</li> </ul>

DTD, described through a context-free grammar  $G_S$ , the target DTD, described through context-free grammar  $G_T$ , and a source document  $D_S$ , which is valid for  $G_S$ , we are looking for the transformation from  $D_S$  to a target document  $D_T$ , so that  $D_T$  is valid for  $G_T$ . In the literature different approaches for this problem are presented, considering only *syntax-directed translation techniques* in this paper. In general, these techniques are based on the recognition of not only the contents but also the structure of the document, before a special transformation is performed. These methods are used in a number of different commercial transformation languages. In some languages the approach has been extended to deal with attributes. In this case, these languages are based on *attribute grammars* (AG). In some syntax-directed translations the explicit declaration of a target grammar is required [10], in order to ensure the syntactical correctness of the target documents. Examples of these techniques are *syntax-directed translation schema* (SDTS) and *tree transformation grammars* (TT-grammar).

In [8] a number of different transformation techniques are introduced and evaluated extensively in conjunction with preparation of a more general approach of “Structured Document Transformations”. For the evaluation of these techniques some important properties are selected, that match our requirements for a transformation technique. These properties are for example the necessity

**Table 2.** Comparison of Transformation Techniques

Transformation Technique	SG	TG	TC	GEN	I/R	O	EL
Simple SDTS [1]	+	+	+	semi	-	-	-
SDTS [1]	+	+	+	semi	+	+	-
Predicate SDTS [9]	+	+	-	semi	+	+	+
Semantic Syntax-Directed Translation [9]	+	+	-	semi	+	+	+
Generalized Syntax-Directed Translation [2]	+	+	-	semi	+	+	+
AG [5]	+	(+)	-	semi	+	+	+
TT-grammar [7]	+	+	+	man	+	+	+

(+ support, - no support, +\* restricted support, ? unknown, (+) optional)

of a specification of a source grammar (SG) and target grammar (TG) of documents for each transformation technique, the correctness of generated target documents with regard to the target grammar (TC) and the type of generating the transformation information (GEN). As a result of analyzing the possible transformations provided by XSLT in section 2, other important requirements also have to be considered. This includes for instance the introduction and removal of nonterminals (I/R), the changing of the order (O) of nonterminals in the grammar and the introduction and removal of new element layers (EL). In table 2 some possible transformation techniques and the corresponding properties based on the considerations in [8] are summarized. The evaluation of this table shows that for the solution of our transformation problem the application of a TT-grammar seems to be a solution of our problem.

#### 4.1 TT-Grammar

Generally speaking, using TT-grammars the relationship between a tree according to a grammar  $G_1$  and another tree according to a grammar  $G_2$  can be specified. This relationship can be interpreted as tree transformation from  $G_1$  to  $G_2$ . According to [7] a TT-grammar is a sextuplet  $G=(G_S, G_T, S_S, S_T, PA, SA)$  with the following meaning:  $G_S$  and  $G_T$  are the source and the target grammars,  $S_S$  and  $S_T$  are sets of input and output subgrammars, PA is a set of production associations, consisting of pairs of associations from an element of  $S_S$  to an element of  $S_T$ , and SA being a set of symbol associations. The source and target grammars are context-free grammars. Each subgrammar contains a subset of productions of a grammar. In the input subgrammar the patterns of a special subtree contained in the source document are specified. In the assigned target subgrammar (of a given production association) the construction statements for subtrees of the result document are contained. Therefore, in the target subgrammar the occurrence of more than one start symbol is possible, describing a set of subtrees. A symbol association is a relation  $(x, y)$  of symbol occurrences with  $x \in S_S$  and  $y \in S_T$  and explicit assignments of symbols from the source to symbols of the target subgrammar. For a concrete transformation of an XML document a special algorithm based on a corresponding TT-grammar has to be specified, referring to [7, 8] for precise descriptions.

## 4.2 Example

In table 3 examples for a source and a target DTD are given, considering only meta data about books, in particular only author and title of books. In comparison to the source DTD in the target DTD the names of elements and the structure of documents are slightly changed. For the application of a TT-grammar at first the transformation of these DTDs to context-free grammars  $G_S$  and  $G_T$  is necessary. The resulting grammars are contained in table 4.

**Table 3.** Examples for DTDs

Source DTD	Target DTD
<!ELEMENT dblp (book*) >	<!ELEMENT result_set (result*)>
<!ELEMENT book (author, title)	<!ELEMENT result book>
<!ELEMENT author (#PCDATA)>	<!ELEMENT book (au, ti)>
<!ELEMENT title (#PCDATA)>	<!ELEMENT au (#PCDATA)>
	<!ELEMENT ti (#PCDATA)>

In order to provide the necessary mapping information including symbol and production associations, a manual analysis of  $G_S$  and  $G_T$  is performed. By comparing  $G_S$  and  $G_T$  the production association shown in table 5 can be derived. In each production association the assignment of a special subgrammar of  $G_S$  to a subgrammar of  $G_T$  is realized. For the symbol associations in the target grammar the notation of

<source symbol>.<target symbol>

is used. For instance, in the production association number 4 the assignment of element <book> is defined, containing the assignment of element <author> to <au> and <title> to <ti>. The transformation of these elements is described in the production associations number 5 and 6. With the notation “PCDATA.PCDATA” the reproduction of the contents of elements is specified.

Based on these considerations the TT-grammar for the given transformation problem can be specified through the context-free grammars  $G_S$  and  $G_T$ , the subgrammars  $S_S$  and  $S_T$  contained in the second and third column of table 5, the production association PA defined in the rows of table 5, and the symbol associations SA. These associations are indirectly contained in the table.

## 4.3 Evaluation of TT-grammars

As a result of the definition of TT-grammars not all classes of operations provided by XSLT are supported. According to the context-free grammars only element nodes are regarded, with the result that copying, renaming, restructuring and deleting element nodes, the introduction and removal of node layers based on element nodes etc. can be expressed in TT-grammars. On the other hand, the application of XSLT permits the transformation of attribute nodes and non type conserving transformations between nodes. These classes of operations cannot be described in TT-grammars. For the solution of the described problems the extension of TT-grammars is necessary in order to support all operations provided by XSLT. First, an explicit notation for accessing attributes in TT-grammars is

Table 4. Context-free Grammars for the DTDs shown in Table 3

Context-Free Grammar $G_S$	Context-Free Grammar $G_T$
$N = \{\text{dblp}, \text{book}, \text{author}, \text{title}, \text{book\_list}\}$ $T = \{\langle \text{dblp} \rangle, \langle / \text{dblp} \rangle, \langle \text{book} \rangle, \langle / \text{book} \rangle, \langle \text{author} \rangle, \langle / \text{author} \rangle, \langle \text{title} \rangle, \langle / \text{title} \rangle, \text{PCDATA}\}$ $S = \{\text{dblp}\}$ $P = \{\text{dblp} \rightarrow \langle \text{dblp} \rangle \text{book\_list} \langle / \text{dblp} \rangle$ $\text{book\_list} \rightarrow \text{book book\_list}$ $\text{book\_list} \rightarrow \epsilon$ $\text{book} \rightarrow \langle \text{book} \rangle \text{author title} \langle / \text{book} \rangle$ $\text{author} \rightarrow \langle \text{author} \rangle \text{PCDATA} \langle / \text{author} \rangle$ $\text{title} \rightarrow \langle \text{title} \rangle \text{PCDATA} \langle / \text{title} \rangle\}$	$N = \{\text{result\_set}, \text{result}, \text{book}, \text{au}, \text{ti}, \text{result\_list}\}$ $T = \{\langle \text{result\_set} \rangle, \langle / \text{result\_set} \rangle, \langle \text{result} \rangle, \langle / \text{result} \rangle, \langle \text{book} \rangle, \langle / \text{book} \rangle, \langle \text{au} \rangle, \langle / \text{au} \rangle, \langle \text{ti} \rangle, \langle / \text{ti} \rangle, \text{PCDATA}\}$ $S = \{\text{result\_set}\}$ $P = \{\text{result\_set} \rightarrow \langle \text{result\_set} \rangle \text{result\_list} \langle / \text{result\_set} \rangle$ $\text{result\_list} \rightarrow \text{result result\_list}$ $\text{result\_list} \rightarrow \epsilon$ $\text{result} \rightarrow \langle \text{result} \rangle \text{book} \langle / \text{result} \rangle$ $\text{book} \rightarrow \langle \text{book} \rangle \text{au ti} \langle / \text{book} \rangle$ $\text{au} \rightarrow \langle \text{au} \rangle \text{PCDATA} \langle / \text{au} \rangle$ $\text{ti} \rightarrow \langle \text{ti} \rangle \text{PCDATA} \langle / \text{ti} \rangle\}$

required. Corresponding to the XSLT standard the following notation for access to attributes of a given element is introduced:

ElementName@AttributeName

The essential characteristic of non type conserving transformations is the transformation of a node name to a node value and vice versa. In TT-grammars the transformation of node values is only valid through “PCDATA.PCDATA”. For defining the access to a node name or node value another extension of the notation is required:

- V(NodeName) stands for an explicit access to the value of a node
- N(NodeName) stands for an explicit access to the name of a node

These unique references to the name and value of a node are required for the determination of the source of data in the context of non type conserving transformations. Additionally, for the concatenation and splitting of nodes other extensions of notations are required. These extensions are not subject of the consideration in this paper, but are part of our current research work.

## 5 An Approach for Generating the XSLT File

The semi-automatic derivation of XSLT information in form of an XSLT file is possible through the application of extended TT-grammars. In principle, in each file a number of nested XSLT template rules are contained. Based on this the approach for deriving XSLT information is defined using the target subgrammar of production associations. In our approach, for each target subgrammar a corresponding template rule is generated automatically. In the end all separated rules are summarized in one XSLT file.

**Table 5.** Production Associations of the TT-Grammar

No.	Source Subgrammar	Target Subgrammar
1.	$\text{dblp} \rightarrow \langle \text{dblp} \rangle$ $\text{book\_list}$ $\langle / \text{dblp} \rangle$	$\text{dblp.result\_set} \rightarrow$ $\langle \text{dblp} \rangle . \langle \text{result\_set} \rangle$ $. \langle \text{result} \rangle$ $\text{book\_list.result\_list}$ $. \langle / \text{result} \rangle$ $\langle / \text{dblp} \rangle . \langle / \text{result\_set} \rangle$
2.	$\text{book\_list} \rightarrow \text{book}$ $\text{book\_list}$	$\text{book\_list.result\_list} \rightarrow$ $\text{book.book}$ $\text{book\_list.result\_list}$
3.	$\text{book\_list} \rightarrow \epsilon$	$\text{book\_list.result\_list} \rightarrow \epsilon$
4.	$\text{book} \rightarrow \langle \text{book} \rangle$ $\text{author}$ $\text{title}$ $\langle / \text{book} \rangle$	$\text{book.book} \rightarrow \langle \text{book} \rangle . \langle \text{book} \rangle$ $\text{author.au}$ $\text{title.ti}$ $\langle / \text{book} \rangle . \langle / \text{book} \rangle$
5.	$\text{author} \rightarrow \langle \text{author} \rangle$ $\text{PCDATA}$ $\langle / \text{author} \rangle$	$\text{author.au} \rightarrow \langle \text{author} \rangle . \langle \text{au} \rangle$ $\text{PCDATA.PCDATA}$ $\langle / \text{author} \rangle . \langle / \text{au} \rangle$
6.	$\text{title} \rightarrow \langle \text{title} \rangle$ $\text{PCDATA}$ $\langle / \text{title} \rangle$	$\text{title.ti} \rightarrow \langle \text{title} \rangle . \langle \text{ti} \rangle$ $\text{PCDATA.PCDATA}$ $\langle / \text{title} \rangle . \langle / \text{ti} \rangle$

In principle, the generation of XSLT template rules is based on fixed defined rules. Because of our current research work only a short example for clarifying the approach is given in the following. With regard to the production association number 1 of table 5 the corresponding XSLT template rule is created:

```

<xsl:template match="dblp">
  <xsl:element name="result_set">
    <xsl:element name="result">
      <xsl:apply-templates select="book_list"/>
    </xsl:element>
  </xsl:element>
</xsl:template>

```

At first to generate this template rule, the determination of the pattern contained in the source document is required. On the left hand side of the production in the target grammar the pattern is contained, using as “match” attribute for creation of the new template rule. For terminal symbols on the right hand side of the production after the dot corresponding element nodes are generated in the template rule. These nodes are carried over to the target document. For nonterminal symbols on the right hand side of the production before the dot only a connection to the corresponding template rule is introduced. The concrete template rule is generated automatically when the corresponding production association is transformed to an XSLT template. Finally, for the generation of text nodes instead of the symbolical representation of text contents with “PCDATA” the XSLT template `<xsl:value-of select="."/>` is created in the resulting template.

## 6 Conclusion and Outlook

In the field of digital libraries diverse adapters are used for the access to sources of bibliographical information. In general, the development of adapters is a complex task, and providing generic adapters for special classes of sources is an appropriate solution. For the instantiation of such adapters special configuration information are required. In our project we developed a generic XML adapter for accessing data, which are stored in arbitrary XML formats. Because of the autonomy of data sources a transformation of XML data is necessary, using the XSLT standard. In this paper we described an approach for semi-automatic derivation of XSLT information based on given source and target DTDs. For this purpose, at first the transformation possibilities and the occurring problems were discussed. Then, a short introduction to the theoretical background for transformation techniques was presented. With the aim of selecting a transformation technique diverse approaches were discussed in short. The main result of this investigation was the choice to use TT-grammars for describing the transformation of XML documents. Because of the definition of TT-grammars some extensions were necessary for considering attributes and non type conserving transformations. A first proposal for these extensions was described, too. The paper concluded with a presentation of an approach for generating the XSLT information based on a given TT-grammar.

Based on the presented concepts future work will go in different directions, including the extension and formalization of new concepts for TT-grammars, which are required for the complete support of all XSLT transformation operations, as well as the implementation of a corresponding prototype tool.

## References

1. A. Aho and J.D. Uilman. *The Theory of Parsing, Translation, and Compiling*, volume 1. Prentice-Hall, 1972.
2. A. Aho and J.D. Uilman. Translations on a  $\lambda$ -Free Grammar. *Information and control*, 19(5):439-475, December 1971.
3. T. Bray, J. Paoli, and C.M. Sperberg-McQueen. Extensible Markup Language (XML) 1.0. Technical report,
4. J. Clarc. XSL Transformations (XSLT) - Version 1.0. Technical report, World Wide Web Consortium, November, 1999
5. P. Deransart, M. Jourdan, and B. Lorho, editors. *Attribute Grammars: Definitions, Systems and Bibliography*, volume 323 of LNCS. Spriger, 1988.
6. J.E. Hopcroft and J.D. Uilman. *Introduction to Automata Theory, Languages, and Computation*, 1999.
7. S.E. Keller, J.A. Perkins, T.F. Payton, and S.P. Mardinly. Tree Transformation Techniques and Experiences. *ACM SIGPLAN Notices*, 19(6):190-201, June 1984.
8. G. Linden *Structured Document Transformations*. PhD thesis, Department of Computer Science, University of Helsinki, Finland, 1997. Series of Publications A, Report A-1997-2.
9. A.B. Pyster and H. W. Buttelmann. Semantic-Syntax-Directed Translation. *Information and Control*, 36(3):320-361, March, 1978.
10. Q.Y. Shi. Semantic-Syntax-Directed Translation and its Application to Image Processing. *Information Sciences. An International Journal*, 32:75-90, 1984.