A Logical Framework for Analyzing Properties of Multimedia Web Documents

Dániel Szegő

Budapest University of Technology and Economics
Department of Measurement and Information Systems
H-1521, pf. 91, Budapest, Hungary
szegod@mit.bme.hu

Abstract. This paper describes some practical and theoretical foundations of Transformational Structured Document Logic (TSDL), which is a logical methodology for analyzing properties of Web documents, XML or HTML, consisting multimedia data, like image, natural language text, video or audio. TSDL can make benefits in searching, or in defining filters for multimedia Web documents. Both syntax and semantics of TSDL are described, and an efficient evaluation algorithm is also briefly introduced.

1 Introduction

During the last ten years, the success of World Wide Web was increasing and it has become part of our daily life. At the golden ages of WWW, pages mainly contained textual elements in well defined formats, like HTML or XML. In our days, this situation has become more complicated. Most Web documents contain non-textual elements either like images, videos, sounds or script language elements. These documents are usually referred as multimedia Web documents. Analyzing such a document is much more challenging than handling simple ones, because it requires knowledge of handling simultaneously different media and the structure of documents. For example, medical or geographical documents usually contain high-resolution images. Automatic processing of such a document surely requires the cooperation of image processing and traditional Web document techniques. Similarly, ornithological documents might include audio elements, which might be analyzed by signal processing techniques.

Transformational Structured Document Logic (TSDL) is a logical framework for analyzing properties of Web documents consisting of non-standard elements like image or audio components. TSDL itself does not realize signal processing, image processing or natural language understanding elements, but it provides a general logical framework in which these elements can easily be integrated. Hence, these elements can be cooperated with each other and with the structural part of a Web document to identify the existence of properties regarding to both structure and multimedia.

The property of a document simply means a true or false value which is true for some multimedia documents and false for the others. This property might seem to be a simple service; however, its importance cannot be overestimated. First of all, proper-

ties can be used in a searching process. A searching query can be implemented as a property, so documents for which the property is true provide the result of the query. This mechanism can be used in a simple WWW search, and it might be the basic query process of an XML database [1]. Secondly, properties of documents can be used to implement Web filters[1] [2,3,4]. The main purpose of a Web filter is to select a few pieces of information from the WWW and to present it to the user in a specific way (e.g. by WAP). As a simple example, we can imagine a businessman desiring monitor the money market. Properties can be used in both finding the necessary information and maintaining the information even if the source documents are reconstructed. Thirdly, documents can be validated with the help of properties, similarly as DTD (Document Type Definition) does for XML documents [5]. If a property is true for a given document, it represents a valid one, whilst non-valid documents are indicated by false properties. Unfortunately, existing validation techniques do not pay attention on multimedia elements [5]. Last but not least, automatic categorization or data mining of documents can be supported by property analysis [6]. For example, a simple categorization process can be realized by identifying the existence of a set of properties over a set of Web documents. Documents with given properties could form a category.

Several approaches were developed to analyze properties of Web documents. Some of them focus on the structure of documents [7,8,9], whilst others deal with the connection of several different but linked document [10,11]. However, they usually do not pay attention on multimedia elements. These elements are usually treated in the same way as other parts of the documents; no special multimedia processing appears.

The reminder of this paper is organized as follows. In section 2, basic concepts behind the logic and the basic architecture are demonstrated. Section 3 introduces mathematical foundations of the logic containing model, syntax, semantics and some demonstrating examples. An efficient algorithm for evaluating TSDL expressions is also proposed in section 3. Finally, section 4 draws some conclusions.

2 Basic Architecture

The basic architecture of TSDL mainly focuses on HTML and XML documents, however theoretically other formats could also be considered. Documents are read and parsed by XML and HTML parsers. Parsing produces a document model which can be considered as the inside representation of the analyzed document. This representation is a directed tree which nodes are tags of the document and edges represent the embedding of tags. Nodes of the graph are usually marked by attributes (which will also be called as atomic predicates in the followings). Multimedia elements are also represented as tags, but they might refer to other resources like files of a directory system or objects of a multimedia database. For example, images of an HTML document are usually stored as standalone jpeg files. Parsers produce only an initial document model which might be further modified by different kinds of transformations.

[1] TSDL was primarily motivated by the IKTA-0186 project which focused on studying and implementing Web filters.

Each transformation creates a new piece of document model which also must be a directed tree. Consequently, transformations can add and delete nodes from the document model or they can change attributes of nodes. Basic concepts of the architecture can be seen on Figure 1.

There are three major kinds of transformations.

1. *Pre-transformations* try to increase the quality of inside representation by adding further important features. As an example, most HTML documents do not follow the HTML specification, therefore transformations need to be used to get the necessary document model.
2. *Filters* delete unnecessary parts of document models. For example, if we are not interested in handling comments or script language nodes, an adequate filter can easily eliminate them.
3. Perhaps the most important transformation is the *multimedia analysis*. It may include transformations for images, sound or natural language texts. These transformations analyze the necessary subparts of the document model and create new models which contain information about the result of the analysis. For example, an image analysis may indicate that a picture is colored by adding an attribute to a node of the picture. Similarly, a natural language transformation can indicate topics of all natural language texts by adding some nodes and attributes to the original graph.

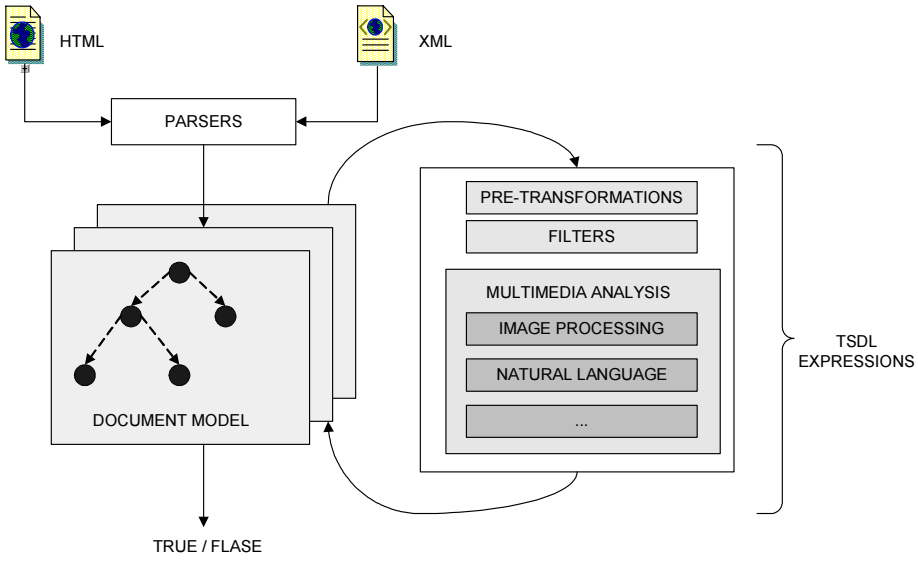

Fig. 1. Basic Architecture

Initially, there is only one document model, but different new models can also be created by applying different transformations. The exact order and number of trans-

formations are controlled by TSDL expressions. Properties of multimedia Web documents are computed simultaneously from the initial and the newly created models.

3 Mathematical Foundations

This section introduces mathematical foundations of TSDL, which are based on the concepts of the previous architecture. The architecture can be formalized on two different levels. Firstly, mathematical model of Web documents can be expressed, which is called as *document level*. It describes the structure of individual documents, focusing mainly on tags and relations between tags. Secondly, transformation of documents, the *transformational level*, will be described, which analyzes interoperations of documents and transformations. Our approach is based on modal logic. Consequently, three main elements of the logic have to be analyzed: the model theory, the syntax and the semantics. All three elements will be described on both document and transformational level. From clear theoretical point of view document level can be regarded as the object level of the formalism, and transformations represent meta level description.

3.1 TSDL Model

The document level of the model is a simple directed tree-graph, which nodes are labeled by atomic predicates. Transformational model is an edge labeled directed graph. Nodes of the transformational level graph are document models and labels of edges are atomic transformations. If two document models are connected by an edge labeled by 't', it means that applying 't' transformation on the first document results in the second document. Theoretically, transformational level graph could be infinite; however, only finite subparts are taken into account at real problems.

The ***document model*** is a six tuple <V, AP, top, *p*, *c*, *ap*>.

- V is a set of nodes of the graph.
- AP is a set of atomic predicate, top$\in$ V is the top node.
- p:V$\rightarrow$V is a partial map associating each node with its parent node.
- c:V$\rightarrow 2^{V}$ is a partial map associating each node with its set of children nodes.
- ap:V$\rightarrow 2^{AP}$ is a partial map associating each node with a set of atomic predicates.
- Paths of the graph are represented by $<v_1,v_2,v_3,\ldots v_{N-1},v_N>$ sequences, where $v_i \in V$, $p(v_i)=v_{i-1}$, and $v_{i+1} \in c(v_i)$.
- Each path of the graph must be circle free, each maximal long path has to start from the *t* top node, and each node of the graph must be reached from the top node through one of the paths.

This definition seems trivial for an XML document [5]. For example, tags can be translated to nodes and embedding of tags represents the parent-children mapping. This transformations is less trivial for an HTML document [12], consequently pre-transformations need to be applied. A multimedia element of a document initially appears as a node whose atomic predicates represent type and resource information.

Applying multimedia transformations, this node might get further atomic predicates or other related multimedia nodes might be added to the model. For example, an image element of an HTML document is a simple node with an 'image' and a file location atomic predicate. After applying a 'resolution identification' transformation, a new atomic predicate may be introduces indicating the resolution of the image (e.g. 'high resolution' or 'low resolution').

The ***transformational model*** is a graph whose nodes are document models, and the edges of the graph are labeled by atomic transformations. More formally, the transformational model is a three tuple <D, T, η>.

- D is a set of document models.
- T is a set of atomic transformations.
- η: D×D→T is a partial map associating each pair of document models (edges) with an atomic transformation.
- Each transformation has to be deterministic: If $\eta(<d_1,d_2>)=\eta(<d_1, d_3>)$ then $d_2=d_3$ (where $d_1,d_2,d_3\in$ D).
- Primary consequence of deterministic behavior is that the following notations can be used: $t_k(d_i) = d_j$, if and only if $\eta(<d_i,d_j>)=t_k(d_i,d_j\in$ D, $t_k\in$ D).

Example 1. As a simple example, one can imagine a company whose confidential Web documents are marked by special confidentiality notes. There are two kinds of notes, an image and a natural language text, indicating that the given document is confidential. Although these notes represent the same content, it is not sure that they are equal bit by bit. For example, the confidentiality image might appear with different resolution, size or colors. Assume that we would like to develop a tool which identifies secret documents. In this case, at least three different kinds of media must be handled. Images must be analyzed by image processing, natural language texts by text analysis and the structure of the document by structure analysis. Analyzing such documents requires at least three different atomic transformations (see Figure 2.). Pre-transformation would eliminate all parts of the document which are irrelevant to confidentiality. Natural language analysis would identify the text and image processing the image of confidentiality.

The formal description of transformational and document model is the following:

- $\{d_0,d_1,d_2,d_3\}\subseteq$D, T={ t_1,t_2,t_3}, $\{<d_0,d_1,t_1>,<d_1,d_2,t_2>,<d_1,d_3,t_3>\}\subseteq\eta$
- $d_1=<V_1, AP_1, top_1, p_1, c_1, ap_1>$, $V_1=\{v_1, v_2, v_3, v_4, v_5\}$,
  AP_1 ={'top','table','emphasis','image','text','resource_ref1','resource_ref2'}
  $top_1=v_1$, $p_1=\{<v_2,v_1>,<v_3,v_1>,<v_4,v_2>,<v_5,v_3>\}$,
  $c_1=\{<v_1,\{v_2,v_3\}>,<v_2,\{v_4\}>,<v_3,\{v_5\}>\}$, $ap_1=\{<v_1,\{$'top'$\}>$,
  $<v_2,\{$ 'table'$\}>,<v_3,\{$'emphasis'$\}>,<v_4,\{$'image','resource_ref1'$\}$,
  $<v_5,\{$'text', 'resource_ref2'$\}>\}$
- $d_2=<V_2, AP_2, top_2, p_2, c_2, ap_2>$, $V_2=\{w_1, w_2, w_3, w_4, w_5\}$
  $AP_2=\{$'top','table','emphasis','image','text','resource_ref1','resource_ref2','conf_text' $\}$ $top_2=w_1$, $p_2=\{<w_2,w_1>,<w_3,w_1>,<w_4,w_2>,<w_5,w_3>\}$,
  $c_2=\{<w_1,\{w_2,w_3\}>,<w_2,\{w_4\}>,<w_3,\{w_5\}>\}$, $ap_2=\{<w_1,\{$'top'$\}>,<w_2,\{$'table'$\}>$,
  $<w_3,\{$ 'emphasis'$\}>,<w_4,\{$'image','resource_ref1'$\}$,
  $<w_5,\{$'text', 'resource_ref2', 'conf_text'$\}>\}$

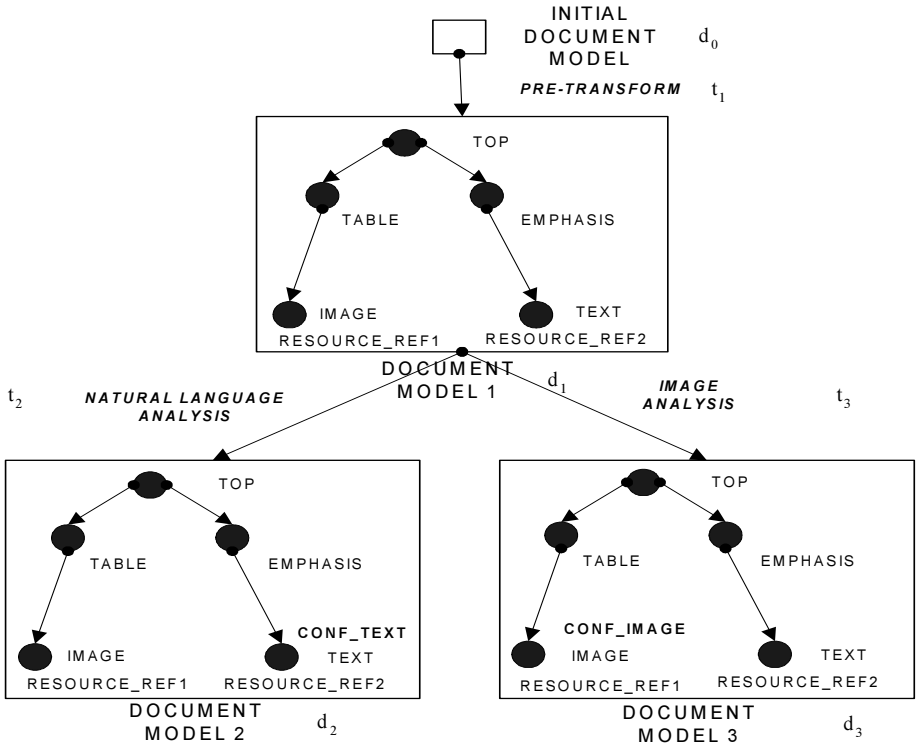

Fig. 2. Simple Example

- d_3=<V_3, AP_3, top_3, p_3, c_3, ap_3>, V_1={n_1, n_2, n_3, n_4, n_5},
  AP_3={'top','table','emphasis','image','text','resource_ref1','resource_ref2',
  'conf_image' } top_3=n_1, p_3={<n_2,n_1>,<n_3,n_1>,<n_4,n_2>,<n_5,n_3>},
  c_3={<n_1,{n_2,n_3}>,<n_2,{n_4}>,<n_3,{n_5}>}, ap_3={<n_1,{'top'}>,<n_2,{ 'table'}>,
  <n_3,{ 'emphasis'}>,<n_4,{'image','resource_ref1','conf _image'},
  <n_5,{'text', 'resource_ref2'}>}

It is important to note that transformational model is only partially determined in this simple example. For example, the structure of the initial document or the structure of the d_2 document after the application of one of the transformations is not known exactly. However, for solving real-world problems, this partial knowledge is sufficient.

3.2 Syntax and Semantics

Similarly to model theory, syntax and semantics are described at both document and transformational level. Document level focuses on expressing statements on stand-alone documents, including expressions on atomic predicates, nodes and relation of

nodes. Transformational level focuses on statements of transformations and documents. The exact meaning of the ***syntax*** can be given with the semantics.

- CT ::= *{***t***}* | CT **;** *{***t***}*
- D :: = *{***a***}*| **T** | ⊥ | D ∧ D | D ∨ D | ¬ D | D **P** D | D $\mathbf{C}_{\exists}$ D | D $\mathbf{C}_{\forall}$ D
- TSDL :: = D | TSDL ∧ TSDL | TSDL ∨ TSDL | <<CT>> TSDL

D describes the document level language and TSDL describes the transformational one. The '*{*a*}*' expression does not represent a syntactic form, but the abbreviation of one piece of atomic predicate. Similarly, '*{*t*}*' denotes one piece of atomic transformation. CT represents the language of complex transformations which is simply sequence of atomic transformations separated by ';'. T and ⊥ represent the top and bottom of a document model which can be regarded as being true or false for the whole document. ∧, ∨, ¬ are the basic logical operators. P should be read as the parent operator, $C_{\exists}$ is the exist-children and $C_{\forall}$ is the all-children operator. <<T>> is a transformational level expression called as deterministic execution of transformations.

The ***semantics of a document level expression*** is interpreted with an 'x' node of an 'M' document model. In a theoretical point of view, nodes of the document model represent possible worlds of the modal logic, parent and children maps realizes relations between possible worlds [13]. We can say that an 'x' node of a given document model 'M' satisfies an expression 'exp', denoted by M, x |= exp. In other words, 'exp' expression is true for 'x' node of 'M' model. An expression is true for an 'M' model if there is an 'x' node for which M, x |= exp.

- M, x |= T, for all x∈ V (where V is the node set of document model).
- M, x |= ⊥, for none of the x∈ V nodes.
- M, x |= a, if and only if, a∈ *ap*(x).
- M, x |= ¬D, if and only if, not M, x |= D.
- M, x |= D_1∧D_2, if and only if, M, x |= D_1 and M, x |= D_2.
- M, x |= D_1∨D_2, if and only if, M, x |= D_1 or M, x |= D_2.
- M, x |= D_1P D_2, if and only if, M, x |= D_1 and M, *p*(x) |= D_2.
- M, x |= $D_1C_{\exists}D_2$, if and only if, M, x |= D_1 and exists an y∈ *c*(x) for which M, y |= D_2.
- M, x |= $D_1C_{\forall}D_2$, if and only if, M, x |= D_1 and exists an y∈ *c*(x), and for all y∈ *c*(x) M, y |= D_2.

The ***semantics of a TSDL expression*** is interpreted with a 'd' document model of a 'TM' transformational model. We can say that a 'd' document model of a given transformational model 'TM' satisfies an expression 'exp', denoted by TM, x |= exp. In other words, 'exp' expression is true for a 'd' document model of 'TM' transformational model.

- TM, d |= exp, if exp is a D expressions and there is an 'x' node of 'd' document model for which d, x |= exp.
- TM, d |= $TSDL_1$∧$TSDL_2$, if and only if, TM, d |= $TSDL_1$ and TM, d |= $TSDL_2$.
- TM, d |= $TSDL_1$∨$TSDL_2$, if and only if, TM, d |= $TSDL_1$ or TM, d |= $TSDL_2$.
- TM, d |= <<t_1;t_2;t_3;..;t_N>> TSDL, if and only if, there is a <d_0,d_1,d_2,…, d_N> sequence of document level models for which d=d_0, $\eta(d_{i-1},d_i) = t_i$ (for all I∈ {1..N}) and TM, d_N |= TSDL.

In practical applications, there is an initial document model which is the starting node of the transformational level graph. In our case, this starting document is the direct output of HTML or XML parsers. It consists of all tags of the document as nodes and all texts or attributes as atomic predicates. TSDL expressions are usually evaluated over the initial document. Certainly, evaluating a TSDL expression requires transforming the existing documents so new document models will be created during the evaluation process. From a clear theoretical point of view, model theory of the logic is sufficient for evaluating properties over Web documents. Properties are formalized as TSDL expressions and evaluating the expressions clearly identifies whether a property holds or not. Therefore, proof theory of TSDL is not studied in this article.

Example 2. Considering the previous example (see Figure 2.), let us suppose that confidentiality text and image cannot be placed anywhere in the document but they must be highlighted somehow. For example, confidentiality image must be in a table and confidentiality text must be emphasized (for example by or HTML tags). Assume that one would like to develop a document checking tool which filters out the documents with non-highlighted confidentiality notes (non-valid documents). It can also be imagined as a document categorization process where there are three categories: documents with no privacy notes, documents with non-valid privacy notes and documents with valid privacy notes. Realizing such a categorization requires the cooperation of different media analysis like image or natural language text processing with structural analysis of the document.
Different properties of Web documents, containing confidentiality text or image, can be easily analyzed by TSDL expressions:

- $<<t_1;t_2>>$conf_text expression is true for those documents which contain confidentiality text.
- $<<t_1;t_3>>$conf_image expression is true for those documents which contain confidentiality image.
- $<<t_1>>((<<t_2>>$conf_text$)\vee(<<t_3>>$conf_image)) expression is true for those documents which contain confidentiality text or image.
- $<<t_1;t_2>>$(conf_ text P emphasis) expression is true for those documents which contain confidentiality text and this text is emphasized (valid confidentiality text).
- $<<t_1;t_3>>$(conf_image P table) expression is true for those documents which contain confidentiality image and this image is in a table (valid confidentiality image).
- $<<t_1;t_2>>$(conf_text P $\neg$emphasis) expression is true for those documents which contain confidentiality text but this text is not emphasized (non-valid confidentiality text).
- $<<t_1;t_3>>$(conf_image P $\neg$table) expression is true for those documents which contain confidentiality image and this image is not in a table (non-valid confidentiality image).
- $<<t_1>>((<<t_2>>$(conf_text P emphasis))$\vee($ $<<t_3>>$ (conf_image P table))) expression is true for those documents which contain valid confidentiality text or image.
- $<<t_1>>((<<t_2>>$(conf_text P emphasis)) $\wedge$ ($<<t_3>>$ (conf_image P table))) expression is true for those documents which contain both valid confidentiality text and image.

- $<<t_1>>$ ((<<t_2>>(conf_text P ¬emphasis))∨(<<t_3>> (conf_image P ¬table))) expression is true for those documents which contain at least one non-valid confidentiality note.

3.3 Evaluation algorithm

Using only model theoretical approach can be sufficient for analyzing properties of multimedia Web documents because a property can be represented by a TSDL expression, Web documents and transformations are represented as the model of the logic and evaluating an expression over a model clearly identifies if the property is hold or not. Consequently, the major question is that what kind of algorithms can be provided to compute the truth of an expression over a model. Similar problems usually arise in model checking. The problem can be divided into two questions. Firstly, a document level approach will be presented to evaluate document level expressions over document level models. Secondly, a transformational level algorithm will be presented with the contribution of document level one.

Document level algorithm is based on a relational algebraic approach. Since expressions are true for some nodes of the document model, therefore a document level expression can be regarded as a subset of nodes of the original document model. First of all, partial maps are stored as binary relations: $V\_P \subseteq V \times V$ for *p* parent map, $V\_C \subseteq V \times V$ for *c* children map and $V\_AP \subseteq V \times AP$ for *ap* atomic predicate map where V is the set of nodes of the document model. Secondly, computational expressions must be associated with the logical operators. These computational expressions determine the set of nodes for which the expression is true. Let us consider that we have two expressions, e_1 and e_2, subsets of nodes of the model for which these expressions are true: V_{e1}, V_{e2}. We can compute the subset of nodes for which $e_1 \wedge e_2$ is true by making an intersection of the two sets ($V_{e1 \wedge e2} = V_{e1} \cap V_{e2}$). Similarly, disjunctions can be computed by union.

Table 1. Document level logical operators and the associated relational algebraic expressions.

Logical expression	Relational algebraic expression
'a' (atomic predicate)	$\sigma_{AP=a}(V)$
T	V
$\bot$	$\varnothing$
¬	$V - V_e$
$\wedge$	$V_{e1} \cap V_{e2}$
$\vee$	$V_{e1} \cup V_{e2}$
P	$\pi_V((V\_P) \cap (V_{e1} \times V_{e2}))$
$C_\exists$	$\pi_V((V\_C) \cap (V_{e1} \times V_{e2}))$
$C_\forall$	$\pi_V((V\_C) \cap (V_{e1} \times V)) - \pi_V(((V\_C) \cap (V_{e1} \times V)) - ((V\_C) \cap (V_{e1} \times V_{e2})))$

Table 1. lists all the relational algebraic expressions that are associated with document level logical expressions[2] (σ denotes selection, π_V is a projection to the first set of a binary relation). Further information about relational algebra can be found in [14].

Unfortunately, TSDL expressions cannot be computed similarly because transformational level models are usually infinite. Therefore, a tableau based method is applied. Complex expressions are decomposed to atomic expressions by applying syntactic rewriting rules. Atomic expressions are document level expressions so they can be evaluated by the previous relational algebraic approach. Using the syntactic rewriting rules on Figure 3 with a depth-first search strategy, a simple evaluation algorithm is presented.

Evaluating a document level expression requires polynomial time and space complexity, because relational algebraic operators can be evaluated in polynomial time, and space is only required for storing the relations (which is polynomial in the size of nodes). Unfortunately, the situation is not so good at transformational level. Syntactic rewriting has branching at disjunctions. Consequently, time complexity of evaluating a TSDL expression can be exponential in the size of the number of TSDL level disjunctions. At practical applications, the number of disjunctions are not extremely high, so time complexity remains acceptable. Applying depth-first search at syntactic rewriting, space complexity of the algorithm remains polynomial.

$$\frac{TM, d \models TSDL_1 \wedge TSDL_2}{TM, d \models TSDL_1 \text{ , } TM, d \models TSDL_2}$$

$$\frac{TM, d \models TSDL_1 \vee TSDL_2}{TM, d \models TSDL_1} \qquad \frac{TM, d \models TSDL_1 \vee TSDL_2}{TM, d \models TSDL_2}$$

$$\frac{TM, d \models \langle\langle t_1;t_2;t_3;..;t_N \rangle\rangle TSDL}{TM, t_N(\ldots t_2(t_1(d))) \models TSDL}$$

TM, d |= TSDL, if TSDL is a document level expressions, and there is an ‘x’ node of ‘d’ document model for which d, x |= exp.

Fig. 3. Tableau based algorithm for evaluating a TSDL expression over a transformational model. The algorithm is a simple modification of the tableau based algorithm of Hennessy-Milner logic, which is primarily used in model checking [15].

[2] Proof of the correctness of relational algebraic expressions is being published in Periodica Polytechnica.

4 Conclusion and Further Work

This paper summarizes some theoretical foundations of Transformational Structured Document Logic, which is a logical methodology for analyzing properties of XML or HTML documents containing multimedia elements. Beside theory, a shell architecture has also been implemented in Java to test the concepts between real circumstances. The architecture realizes an HTML and an XML parser, the algorithms for evaluating document level and transformational level expressions, and several basic atomic transformations.

In a sense, TSDL does not represent a new logic, because it is a subset of several more general logical frameworks. Document level of TSDL can be considered as a special kind of Propositional Dynamic Logic. For example, DIFR logic (a kind of Propositional Dynamic Logic) contains more syntactic elements than document level of TSDL does [16]. Transformational level of the logic realizes a special Hennessy-Milner (HM) logic which nodes are document level models, and there is only one modal operator at this level called deterministic execution of transformations. Deterministic execution is the same as all executions and some executions in HM, because executions are not necessarily deterministic in HM but surely deterministic in TSDL. However, these general logical frameworks usually do not concentrate on Web documents, instead they deal with actions or time [13,15]. Therefore, TSDL can be considered as a domain specific logic which syntax and semantics also focus on Web documents. This specialization causes several benefits. Firstly, syntax of the logic directly expresses the necessary formulas for the most common applications. Secondly, the limited approach entails several computational benefits which are primarily manifested in easy and relatively fast evaluation algorithms. Although this article covers mainly the model theory of the logic, the limited approach could result in significant simplification in proof theory.

TSDL can be extended and further studied from both practical and theoretical points of views:

- The efficiency of the evaluation algorithm can be further increased by using sophisticated data structures, like Hash tables or Hash trees [17].
- The applications of soft evaluation techniques would lead not only to the identification of the truth or falsity of a property of Web documents, but to the recognition of a more descriptive value as well.
- Pre-transformations, filters and multimedia transformations should be further investigated. At this point they are quite ad-hoc and implemented by Java objects. However, an ontology of multimedia transformations would also be useful.
- Proof theory of the logic should be developed. With proof theory, consequences or common properties of the truly evaluated expressions could be identified.
- Beside theoretical issues and experimental implementation, developing industrial application using TSDL remains an open question needing further investigation and research.

References

1. XQuery 1.0: An XML Query Language, W3C Working Draft, http://www.w3.org/TR/xquery/#nt-bnf, 2002.
2. Raymond Lau, Arthur H.M, A Logic-Based Approach for Adaptive Information Filtering Agents, Lecture Notes in Computer Science, Vol. 2112, pp.269-280. 2001.
3. R. Cooley and P. Tan and J. Srivastava, Websift: The Web site information filter system, In Proceedings of the 1999 KDD Workshop on Web Mining, San Diego, 1999.
4. M. Ackerman and B. Starr and M. Pazzani, The Do-I-Care Agent: Effective Social Discovery and Filtering on the Web, In RIAO '97: Computer-Assisted Information Searching on the Internet, pages 17--31, Montreal, CA, June 1997.
5. XML Base, W3C Recommendation, http://www.w3.org/TR/xmlbase/, 2001.
6. William I. Grosky and Yi Tao, Multimedia Data Mining and Its Implications for Query Processing, "{DEXA} Workshop 1998.
7. XML Path Language (XPath), Version 1.0, W3C Recommendation, http://www.w3.org/TR/xpath, 1999.
8. XML Path Language (XPath) 2.0, W3C Working Draft, http://www.w3.org/TR/xpath20/ 2002.
9. G. Conforti and G. Ghelli and A. Albano and D. Colazzo and P. Manghi and C. Sartiani, The Query Language TQL, Proc. of 5th International Workshop on Web and Databases (WebDB 2002), 2002.
10. K. Lodaya, R. Ramanujam, An Automation Model of User-Controlled Navigation on the Web, Implementation and Application of Automata, 5th International Conference, CIAA 2000.
11. Luca de Alfaro, Model Checking the World Wide Web, Lecture Notes in Computer Science, Volume 2102, pp. 337-350, 2001.
12. HTML 4.01 Specification, W3C Recommendation, http://www.w3.org/TR/html401/, 1999.
13. Imre Ruzsa, Introduction to Modern Logic, in Hungarian, Osiris Press 2000.
14. Jeffrey D.Ullman, Jennifer Widom, A First Course in Database Systems, Prentice Hall, Inc, 1997.
15. M. Müller-Olm, D. Schmidt, B. Steffen, Model-Checking: A Tutorial Introduction. Lecture Notes in Computer Science (LNCS), Vol. 1694, pp. 330--354. Springer-Verlag, 1999.
16. Giuseppe De Giacomo, Maurizio Lenzerini, PDL-based framework for reasoning about actions, In Lecture Notes in Artificial Intelligence n. 992, pages 103--114. Springer-Verlag, 1995.
17. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Introduction to Algorithms, MIT Press, 1990.