PERIODICA POLYTECHNICA SER. EL. ENG. VOL. 47, NO. 3–4, PP. 311–324 (2003)

STRUCTURED DOCUMENT LOGIC

Dániel SZEGŐ

Department of Measurement and Information Systems
Budapest University of Technology and Economics
H–1521, Budapest, Hungary
e-mail: szegod@mit.bme.hu

Received: Oct. 1, 2003

Abstract

This paper describes some practical and theoretical foundations of Structured Document Logic (SDL[1]), which is a logical methodology for analyzing properties of Web documents, like XML or HTML. SDL can make benefits in searching of HTML pages, or in defining filters for web documents. Both syntax and semantics of SDL are described, and an efficient evaluation algorithm is also introduced.

Keywords: Web document, XML, HTML, modal logic.

1. Introduction

As computers are becoming part of our daily life, non-professional users face more and more difficulties and information overload. This is especially true for the Web. During the last ten years, the success of World Wide Web was increasing, and it has become part of our daily life. This success results in an almost infinite number of various Web pages, easily accessible for everyone. To avoid information overload of non-professional users, sophisticated filters for Web documents need to be developed and applied. The purpose of these filters is to select some pieces of the necessary information from the Web and show them to the user in a user-specific way [1, 2, 3]. This can be another Web document accessible by browsers, but other solutions like mobile phones, sms, or wap could also be imagined.

As a simple example, we can imagine a businessman, who would like to be informed regarding the money market. He surely does not want to be bothered about the sources of information. This represents two main constraints on filters of Web documents. Firstly, filters should provide a way to select some pieces of information from different pages and present them in a user-specific way, for example on one Web page. Secondly, filters should keep track of the changes of the source Web documents, and present the same information even if the structure of the sources is totally reconstructed.

[1]SDL was developed within the framework of IKTA-0186 project sponsored by the Hungarian Ministry of Education.

SDL (Structured Document Logic) is a logical formalism for analyzing properties of Web documents, typically HTML or XML documents. SDL does not realize a Web filter itself, but it can serve as a basic building block of designing and implementing filters. Firstly, it makes the information search easy, since evaluating an SDL expression over a Web document is the same as finding a part of the document which is similar to a template. Secondly, keeping track of changes of documents can be realized by evaluating and reevaluating SDL expressions over a document.

The content of this paper is organized as follows. In Section 2, related work is surveyed. Section 3 and 4 introduce mathematical foundations behind the logic containing model, syntax, semantics and some demonstrating examples. An efficient algorithm for evaluating SDL expressions based on relational algebraic approach is proposed in Section 5. Section 6 and 7 briefly summarize a possible extension to SDL, and some results related to implementing an experimental architecture. Finally, section 8 and 9 draw some conclusions.

2. Related Work

Several approaches have been developed, which use modal logics in the context of Web [4, 5, 6]. However, these approaches concentrate on handling the Web as a whole, rather than modelling structure of stand-alone documents. They provide an excellent way for analyzing connectivity properties of pages like checking whether every page of a Web site is reachable from all other pages or that all paths from a main page to pages with confidential information must go through an access control page. Unfortunately, they do not provide an adequate way for modelling the structure of documents in detail, which has primary importance in filtering information of Web documents.

On the other hand, there are several non-logical approaches for analyzing structural properties of XML documents [7, 8, 9, 10]. Inevitably, XPath [7, 8] (XML Path Language) is the most important one from an industrial point of view. It is commonly used in several widespread industrial technologies like in XSLT [11] or in XQuery [12]. Unfortunately, these approaches are usually in lack of simple formal semantics. For example, the XPath specification [7, 8] does not contain anything which could be called formal semantics, although attempts were made for formalizing fragments of it [10, 13, 14]. Another example is TQL (Tree Query Language) [9] which is a template matching language rather than a logical approach.

Structured Document Logic can be regarded as an attempt to overcome some of the limitations of XPath, and to define a pure logical approach for analyzing properties of XML or HTML documents. A more detailed comparison between SDL, XPath and other logical formalisms can be found in the comparison section.

3. SDL Model

First of all, the model of the logic has to be specified exactly, which is practically a formalized view of a Web document. The model of Structured Document Logic is basically a directed tree graph, nodes of which are associated with atomic predicates.

For the sake of simple semantics, edges of the graph are not expressed in the usual way, instead they are described by two maps.

Definition 1 (SDL Model) The SDL model is a six tuple $< V, AP, t, p, c, ap >$ where V is a set of nodes of the graph, AP is a set of atomic predicate, $t \in V$ is the top node.

$p : V \to V$	is a partial map associating each node with its parent node.
$c : V \to 2^V$	is a partial map associating each node with its set of children nodes.
$ap : V \to 2^{AP}$	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 natural for an XML document [15]. For example tags can be translated to nodes, and embedding of tags represents the parent-children mapping. It is less trivial for an HTML document [16], consequently pre-transformations need to be applied. These transformations attempt to capture the necessary parts of an HTML document for a given task. SDL model will be set up with these pre-filtered parts.

The transformations are strongly task-dependent and consist of plenty ad-hoc mechanisms.

4. SDL Syntax and Semantics

Creating logic usually consists of two major steps. Firstly, syntax needs to be exactly specified, typically as a formal language. Secondly, relationship between the syntax and the model has to be given in the form of rules [17].

Syntax needs to express atomic statements, conjunctions, disjunction and negation of atomic statements, and some structural properties related to parent-children relationship. The exact meaning of the syntactical elements can be given with the semantics.

Definition 2 (SDL Syntax)

$$S ::= \{a\} | T | \bot | S \wedge S | S \vee S | \quad \neg S | SPS | SC_{\exists} S | SC_{\forall} S,$$

where $\{a\}$ does not represent a syntactic form, but the abbreviation of one piece of atomic predicate. T and $\bot$ represent the top and bottom, $\wedge$, $\vee$, $\neg$ 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.

Definition 3 (SDL Semantics) The semantic meaning of an SDL expression is interpreted with an 'x' node of an 'M' SDL model. We can say that an 'x' node of a given SDL 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 $x \in V$ (where V is the node set of SDL model).
$M, x| = \bot$, for none of the $x \in V$ nodes.
$M, x| = a$, if and only if, $a \in ap(x)$.
$M, x| = \neg S$, if and only if, not $M, x| = S$.
$M, x| = S_1 \wedge S_2$, if and only if, $M, x| = S_1$ and $M, x| = S_2$.
$M, x| = S_1 \vee S_2$, if and only if, $M, x| = S_1$ or $M, x| = S_2$.
$M, x| = S_1 P S_2$, if and only if, $M, x| = S_1$ and $M, p(x)| = S_2$.
$M, x| = S_1 C_\exists S_2$, if and only if, $M, x| = S_1$ and exists an $y \in c(x)$ for which $M, y| = S_2$.
$M, x| = S_1 C_\forall S_2$, if and only if, $M, x| = S_1$ and exists an $y \in c(x)$, and for all $y \in c(x) M, y| = S_2$.

In the following, a simple HTML document, its translation to SDL model, and some true and false SDL expressions will be presented to demonstrate the previous concepts.

Let a simple HTML document be the following one:

```
<html>
    <head>
      <title> Trial HTML document </title>
    </head>
    <body>
      <b> bold text </b>
      <p> simple text
      <p> <u> underlined text </u>
    </body>
</html>
```

Its SDL model might be the following:

It is important to note that the SDL model does not need to copy the exact structure of a Web document, because the model is set up with the help of several pre-transformations and filters. In our example, some of the tags do not appear in the model at all, and some atomic predicates are newly introduced. It can also be imagined, that the model hardly copies the syntactic structure of the document, instead it consists of semantic and pragmatic hints about the content of the text.

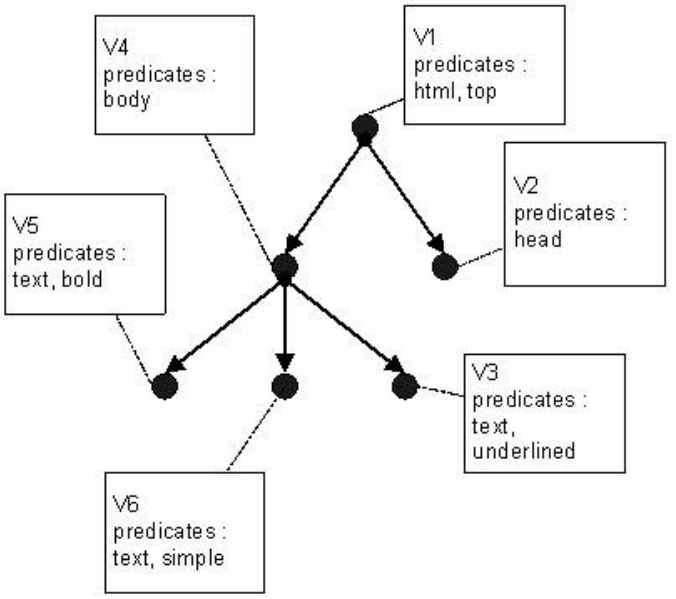

Fig. 1. SDL example

With the help of pre-transformations and filters, the model of a Web document might become simpler, more structured, and might contain special semantic hints.

Considering the previous example, the following expressions can be evaluated.

- body = true
- html $\wedge$ top = true
- text $\wedge$ top = false
- body $C_{\exists}$ text = true
- body $C_{\forall}$ text = true
- ($T\,P$ top) $C_{\forall}$ text = true
- ($T\,P$ top) $C_{\exists}$ underlined = true
- ($T\,P$ top) $C_{\forall}$ underlined = false
- ($T\,P\,(T\,C_{\exists}$ head)) $\wedge\neg$ head = true
- (($T\,P\,(T\,C_{\exists}$ head)) $\wedge\neg$ head) $C_{\forall}$ text = true
- (($T\,P\,(T\,C_{\exists}$ head)) $\wedge\neg$ head) $C_{\forall}$ underlined = false

Expressions like these can be formed to query certain properties of Web documents. Web documents are represented as SDL models, and properties as SDL expressions which are evaluated over the model. This can be regarded as a special model checking, therefore most of the efforts were focused on the model theoretic questions of the logic, whilst proof theory has not been investigated yet in detail.

One of the motivations behind SDL semantics is to easily represent simple graph-matching (finding the sub-graphs of the model graph which are isomorphs with a given sample graph). This sample graph can easily be expressed by SDL expressions using only conjunction, exist-children, and parent operators. For example, finding all sub-graphs of the model which are two length paths, can be realized by evaluating the '$(T\,P\,(T\,P\,T))$' expression. That is the reason why modal operators are defined by conjunction instead of disjunction or implication. Note, however, that other modal operators with disjunction or implication could also be easily expressed with the help of the existing ones.

For example, we could imagine a disjunctive parent operator:

$$M, x| = S_1 \underline{P} S_2, \text{ if and only if, } M, x| = S_1 \text{ or } M, p(x)| = S_2$$

However, $\underline{P}$ could be expressed by the existing parent and disjunction operators.

$$a_1 \underline{P} a_2 = a_1 \vee (T\,P\,a_2).$$

On the one hand, SDL can be used in a searching process. Queries can be formed from SDL expressions, and evaluation produces not only the truth value, but certain nodes of the model for which the expression is true. On the other hand, monitoring the truth value of certain SDL expressions over time can clearly identify how much an HTML document was reconstructed.

5. Relational Algebraic Semantics of SDL

The previous section introduced the semantics of SDL in detail. However, constructing an evaluation algorithm directly based on the definitions raises several problems. On the one hand, a brute force algorithm can be very inefficient, its time complexity might be exponential. On the other hand, identifying true or false attribute of an SDL expression is not adequate for searching purposes. It is also necessary to identify certain nodes of the model for which an SDL expression is true or false.

These are the reasons why a more sophisticated evaluation algorithm has been developed. This algorithm takes advantage of the relational algebra, commonly used in relational database systems [18]. Further information about relational algebra can be found in [18]. Core idea of the algorithm is based on the relational algebraic semantics of SDL, which is expressed by the following statements.

Definition 4 (depth of an SDL expression) Depth v of an SDL expression is defined in the following way:

1. $v(a) = 0$ if a is an atomic predicate.
2. $v(T) = 0$, and $v(\bot) = 0$.
3. $v(\neg S) = 1 + v(S)$.

4. $v(S_1 X S_2) = 1 + \max(v(S_1), v(S_2))$ if ${}_S1$, S_2 are SDL expressions, and $X \in \{\wedge, \vee, P, C_\exists, C_\forall\}$ is a binary operator.

Lemma 1 *Let V and W be two sets, $Q, R \subseteq V \times W$ binary relations on the sets and π_V is the projection to the first set of a binary relation.*

$$v \notin \pi_V(R - Q)$$

if and only if

1. *there is no $q \in V$ for which $< v, q > \in R$ or,*
2. *for all $q \in W$, for which $< v, q > \in R$ it is also true that $< v, q > \in Q$.*

Proof. If 1, or 2 is true, $v \notin \pi_V(R - Q)$ obviously holds.

Conversely, suppose that there is at least one q for which $< v, q > \in R$ and $< v, q > \notin Q$. This means that $< v, q > \in R - Q$, consequently $v \in \pi_V(R - Q)$.

□

Theorem 1 *Let V be the set of nodes of an M SDL model, exp an SDL expression and $V_{\text{exp}} \subseteq V$ is the set of x nodes for which $M, x| = \text{exp}$.*

V_{exp} can be computed from V with relational algebra operators.

Proof. Proof is based on induction of depth of SDL expressions.

The SDL model is stored in three relations: $V\_P \subseteq V \times V$, $V\_C \subseteq V \times V$, $V\_AP \subseteq V \times AP$, describing the p, c, and ap maps as binary relations. In the following π_V denotes the projection to the first set of a binary relation.

If $v(\text{exp}) = 0$.

- $V_T = V$ based on the definition.
- $V_\perp = \emptyset$ based on the definition.
- $V_a = \pi_V(\sigma_{AP=`a'}(V\_AP))$ where 'a' is an atomic predicate.

  If $v \in \pi_V(\sigma_{AP=`a'}(V\_AP))$, that means '$v$' is associated with the 'a' predicate both in the V_AP and in the M model with the help of 'ap' map. Consequently $M, v| = a$.

  Conversely, if $v \notin \pi_V(\sigma_{AP=`a'}(V\_AP))$ there is no 'a' predicate associated with v, so v surely does not satisfy 'a'.

Supposing that E and H are SDL expressions of maximum depth l, and V_E, V_H has already computed by relational algebra operators.

- $V_{E \wedge H} = V_E \cap V_H$ where E and H are SDL expressions.

  $v \in V_E \cap V_H$ if and only if $v \in V_E$ and $v \in V_H$, which means that $v| = E$ and $v| = H$.
- $V_{E \vee H} = V_E \cup V_H$ where E and H are SDL expressions.

  $v \in V_E \cap V_H$ if and only if $v \in V_E$ or $v \in V_H$, which means that $v| = E$ or $v| = H$.

- $V_{\neg E} = V - V_E$ where E is an SDL expression.

  $v \in V - V_E$ if and only if $v \in V$ and $v \notin V_E$ which means that v does not satisfy E.

- $V_{EPH} = \pi_V((V\_P) \cap (V_E \times V_H))$

  $< v, q > \in (V\_P) \cap (V_E \times V_H)$ if and only if $< v, q > \in (V\_P)$ and $< v, q > \in (V_E \times V_H)$ which means that $p(v) = q$, and $v \in V_E$ and $q \in V_H$. Consequently $v| = E$, and $q| = H$ and $p(v) = q$.

- $V_{EC\exists H} = \pi_V((V\_C) \cap (V_E \times V_H))$

  $< v, q > \in (V\_C) \cap (V_E \times V_H)$ if and only if $< v, q > \in (V\_C)$ and $< v, q > \in (V_E \times V_H)$ which means that $q \in c(v)$, and $v \in V_E$ and $q \in V_H$. Consequently $v| = E$ and at least one $q \in c(v)$ exists for which $q| = H$.

- $V_{EC\forall H} = \pi_V((V\_C) \cap (V_E \times V)) - \pi_V(((V\_C) \cap (V_E \times V)) - ((V\_C) \cap (V_E \times V_H)))$

  $v \in \pi_V((V\_C) \cap (V_E \times V)) - \pi_V(((V\_C) \cap (V_E \times V)) - ((V\_C) \cap (V_E \times V_H)))$ if and only if $v \in \pi_V((V\_C) \cap (V_E \times V))$, and $v \notin \pi_V(((V\_C) \cap (V_E \times V)) - ((V\_C) \cap (V_E \times V_H)))$.

  - $v \in \pi_V((V\_C) \cap (V_E \times V))$ if and only if there is a $q \in V$ for which $< v, q > \in V\_C$, and $< v, q > \in V_E \times V$.
    Consequently $v \in V_E$ implying $v| = E$, and there is at least one q for which $< v, q > \in V\_C$.
  - $v \notin \pi_V(((V\_C) \cap (V_E \times V)) - ((V\_C) \cap (V_E \times V_H)))$ can occur in two ways (based on the previous lemma):
    1. There is no $q \in V$ for which $< v, q > \in ((V\_C) \cap (V_E \times V))$, which contradicts the previous part of the proof
    2. for all $q \in V$, for which $< v, q > \in ((V\_C) \cap (V_E \times V))$ it is also true that $< v, q > \in ((V\_C) \cap (V_E \times V_H))$, which implies that $v| = E, q| = H$ and $q \in c(v)$. For a given v, $< v, q > \in ((V\_C) \cap (V_E \times V))$ pairs consist of all children of v as $q - s$, therefore for all q children of v, $q| = H$ holds.

□

With the help of the previous theorem, an SDL expression can be translated into a relational algebraic formula. This formula is the linear size of the size of SDL expression.

Since all relational algebraic operators can be evaluated in polynomial time, therefore an SDL expression can be evaluated on an M model in polynomial time of both the size of M, and the size of the expression.

More precisely, all but the all-children operator can be computed by maximum two pieces of embedded loops over the node set of a model. All-children operator can be computed by three embedded loops. Consequently, time complexity of the algorithm based on relational algebraic approach is $O(l * v^3)$, where l is the 'full

length' of the SDL expression, which is manifested in the number of operators and atomic predicates, and v is the number of nodes of the model (O represents the asymptotic bound of the algorithm [19]). This complexity might be reduced by using sophisticated data structures, like Hash tables and trees [19]. Space complexity of the algorithm is determined by the binary relations (V_P, V_C), supposing that the number of possible atomic predicates is limited. Consequently, the algorithm uses $O(v^2)$ space, where v is the number of nodes of the model.

6. Extension of SDL

Although SDL provides an adequate way of evaluating expressions over models of Web documents, it is far from being perfect. Both its model and its description capacity can be extended. A possible extension could be the first-order version of the logic, which would extend both the syntax and the model. In this section, a less ambitious extension will be introduced to better handle paths of the model graph.

Unfortunately, SDL does not really handle paths of the model graph. Its operators are applied to individual edges rather than to sequences of edges. To handle sequence of edges, SDL syntax and semantics are extended, yielding a new logic called ESDL (Extended Structured Document Logic).

Definition 5 (ESDL Syntax)

$$S ::= \{a\}|T|\bot|S \wedge S|S \vee S|\neg S|SPS|SC_\exists S|SC_\forall S|SP^\infty S|SC_\exists^\infty S.$$

The definition consists of two additional operators for handling sequences, P^∞ is the ancestor and $C_\exists^\infty$ the descendant operator.

For defining semantics of ESDL, paths of the model have to be further characterized. Let x be a node of the SDL model. Parent path of x node is a $< v_1, v_2, v_3, \ldots v_{N-1}, v_N >$ sequence of nodes of the model, for which $v_1 = x$ and $p(v_i) = v_{i+1}$ for all $v_i \in V$. Children path of x node is a $< v_1, v_2, v_3, \ldots v_{N-1}, v_N >$ sequence of nodes of the model, for which $v_1 = x$ and $v_{i+1} \in c(v_i)$ for all $v_i \in V$. With the help of parent path and children path concepts, semantics of ESDL can be easily defined.

Definition 6 (ESDL Semantics) $M, x| = S_1 P^\infty S_2$, if and only if, $M, x| = S_1$ and there exists a $< v_1, v_2, v_3, \ldots v_{N-1}, v_N >$ parent path of x, and there exists a y element of the path for which $M, y| = S_2$.

$M, x| = S_1 C_\exists^\infty S_2$, if and only if, $M, x| = S_1$ and there exists a $< v_1, v_2, v_3, \ldots v_{N-1}, v_N >$ children path of x, and there exists a y element of the path for which $M, y| = S_2$.

The semantics of other operators remain unchanged.

Relational algebraic semantics of SDL can be extended to ESDL, with storing not only V_C and V_P relations, but two additional ones. $V\_IC \subseteq V \times V$ binary

relation associates a given node to those which can be reached by children paths from the given one. Similarly, $V\_I\,P \subseteq V \times V$ binary relation associates a given node to those which can be reached by parent paths from the given one. Time complexity of evaluating an ESDL expression is the same as in the SDL case, but space complexity is slightly increased because of the additional relations.

7. Implementation

To analyze ESDL under real circumstances, a shell architecture has been implemented and tested in Java. It consists of a core ESDL implementation, and some additional technological elements like XML and HTML parsers. Web documents are read from files by XML and HTML parsers and transformed to an internal representation. During the transformation different pre-transformations and filters can be used to select and transform parts of the input document. For example, a filter might delete tags from an input HTML document which are related to comments or script language. These pre-transformations and filters are implemented as Java objects.

The internal representation consists of the ESDL model in the form of binary relations. ESDL expressions are read from an XML file, and evaluated over models of HTML or XML input files (*Fig. 2*). The result of the evaluation is not only a true or false signature. If an expression is true over a model, the evaluation produces all the nodes for which the expression is true. For false expressions, there is a soft evaluation algorithm. It produces a real number between 0 and 1 indicating how much the expression is false. The soft evaluation is based on comparing the size of the whole expression and the size of sub-expressions which themselves are true.

8. Comparison

As mentioned earlier, ESDL can be regarded as an extension of XPath. It is important to examine what expressions can be formed in XPath, in ESDL, or in both of them.

Although XPath has a wide range of path expressions, which are similar to ESDL modal operators, these expressions cannot be combined freely with other logical operators [7, 8]. For example, the following table contains several ESDL expressions which can be easily expressed by XPath.

However, there is no way for freely combining path expressions like parent or child, with disjunction conjunction or negation in Xpath [7]. Consequently, expressions like '(a P (not $((bC_\forall$ c)$\vee d$))' cannot be expressed with one XPath expression.

Certainly, XPath is more an industrial than a theoretical solution, therefore it contains several elements which do not appear in ESDL like function calls, numbers, equations, strings, and string handling procedures.

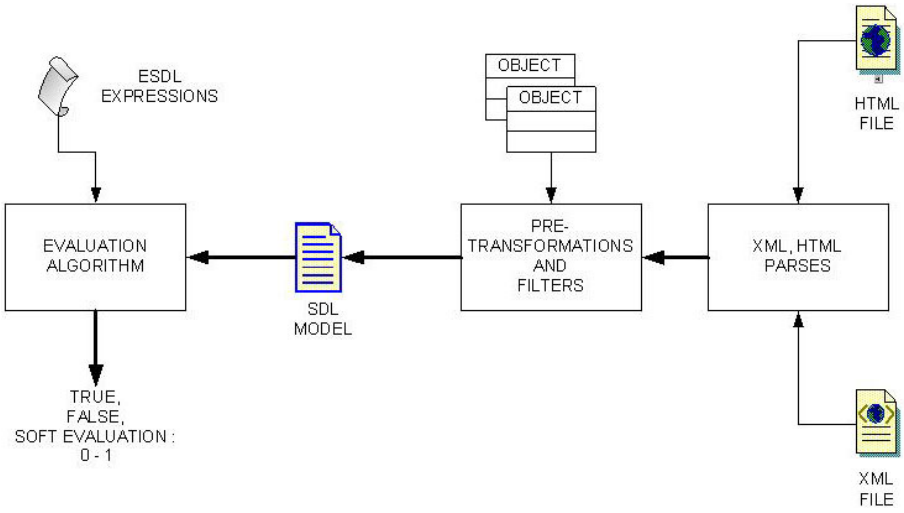

Fig. 2. Expression evaluation

Table 1. SDL and Xpath expressions

Xpath expression	ESDL expression
Parent	P
Child	$C_\exists$
Ancestor	P^∞
Descendant	$C_\exists^\infty$
ancestor-or-self::element	element $\vee(T\,P^\infty$ element)
descendant-or-self::element	element $\vee(T\,C_\exists^\infty$ element)
|, and	$\wedge$
Or	$\vee$

Expressiveness of ESDL could also be examined in conjunction with other logical formalisms. From a clear mathematical point of view, ESDL is a multimodal logic. Nodes of the model represent the possible worlds. Parent and children maps realize the relations between worlds in SDL, whilst the transitive closure of parent and children maps are the relations between possible worlds in ESDL. Since general type logics, like Montague's logic [17], entail all possible modal logics, therefore ESDL can be regarded as a very special case of them.

From a more practical point of view, ESDL can be regarded as a sublogic of three general logical frameworks, two of them are modal, and one of them is first order logic. Since mainly the model theory of both SDL and ESDL was investigated,

the term sublogic simply means that both the syntax and the semantics of the logic are restricted.

First of all, ESDL can be regarded as a sublogic of a general PDL, Propositional Dynamic Logic, called DIFR [20]. DIFR is a logic for arguing about actions of a dynamic system. Its model theory is an edge labelled graph. Its syntax is able to express statements about states, actions, precondition, effects and composition of actions. An ESDL model is a special case of a general edge labelled graph, because it is a special tree and there is only one edge label, called 'children'. Similarly, all ESDL statements can be expressed in PDL with the help of only four action expressions, C, C^-, C^*, C^{-*} (where C denotes the 'children' action).

Secondly, ESDL can be regarded as a subset of CTL (Computational Tree Logic), but only if CTL consists of explicit representation of the past [21].

Last but not least, ESDL expressions can be given with help of some description logics [22, 23]. Not mentioning the basic operators like conjunction, the DL must contain full negation, and some special role constructors, like inverse or transitive closure. Certainly, models of ESDL are restricted to trees of relations, as opposed to DL where no such restriction exists.

In a sense SDL and ESDL do not represent a new logic, because it is a subset of three general logical frameworks. However, those frameworks are more general than ESDL. For example, with the help of DIFR different actions, conjunctions, disjunctions of actions, sequential compositions and test of actions can also be expressed, which are not used in ESDL at all. On the other hand, speaking about actions or time instead of parent or children in the context of HTML or XML documents would be a little strange. Therefore SDL and ESDL can be regarded as a domain-specific logic, syntax and semantics of which also focus on Web documents. This specialization results in several benefits. Firstly, the syntax of the logic directly expresses the necessary formulas for the most common applications. For example, with the help of children, parent and conjunction operators a very common problem, the graph-matching, can be easily expressed. Secondly, the limited approach entails several computational benefits which are primarily manifested in fast evaluation and soft evaluation algorithms. Although this article covers mainly the model theory of the logic, the limited approach could cause significant simplifications in proof theory.

Besides, the situation is very similar to other logics. Although type theoretical logic can be regarded as a general extension of every modal and non-modal logic [17], usually domain-specific logics are used in applications, e.g. PDL in describing dynamic systems, DL for terminological inference, and so on.

9. Conclusion and Further Work

This paper summarizes some practical and theoretical foundations of Structured Document Logic, which is a logical methodology for analyzing properties of XML or HTML documents. Both syntax and semantics, and motivations behind were

discussed. Beside the normal semantics, a relational algebraic semantics was also given, mainly for efficient computational purpose. Some issues regarding the extensions and implementation have also been briefly discussed in the latest sections. Although there are some logical frameworks, which are more expressive than ESDL, they do not focus on Web documents. Therefore, ESDL can be regarded as a new logic among domain-specific modal logics. It can be more naturally and efficiently used in the context of Web documents than other logical formalisms.

ESDL can be extended and further analyzed from both practical and theoretical point of view.

- The efficiency of the evaluation algorithm can be further increased by using sophisticated data structures like Hash tables or Hash trees. This is more important than it seems to be, because examining linked documents simultaneously might cause exponential growth in the size of SDL model.
- It is important to investigate soft evaluation techniques. With their help, not only the truth or falseness of a property of Web documents can be identified, but a more descriptive value would be presented. At this point, only an ad-hoc soft evaluation exists, which might be extended to fuzzy logic.
- Pre-transformations and filters should be further investigated. At this point they are quite ad-hoc and implemented by Java objects. However, ontology of filters and text-based representation could also be useful.
- Proof theory of the logic would be useful. With the help of proof theory, consequences or common properties of the truly evaluated expressions could be identified.
- Beside the theoretical issues and experimental implementation, developing an industrial application using SDL has remained an open question, which needs further investigation and research.

References

[1] LAU, R. – ARTHUR, H. M, A Logic-Based Approach for Adaptive Information Filtering Agents, *Lecture Notes in Computer Science*, **2112** (2001), pp. 269–280.

[2] COOLEY, R. – TAN, P. – SRIVASTAVA, J., Websift: The Web Site Information Filter System, *Proceedings of the 1999 KDD Workshop on Web Mining*, San Diego, 1999.

[3] ACKERMAN, M. – STARR, B. – PAZZANI, M., The Do-I-Care Agent: Effective Social Discovery and Filtering on the Web, *In RIAO '97: Computer-Assisted Information Searching on the Internet*, Montreal, CA, 1997, pp. 17–31.

[4] LODAYA, K. – RAMANUJAM, R., An Automation Model of User-Controlled Navigation on the Web, *Implementation and Application of Automata, 5th International Conference, CIAA* 2000.

[5] ALECHINA, N. – DEMRI, S. – DE RIJKE, M., A Modal Perspective on Path Constraints. *Journal of Logic and Computation*, **13** No. 6 (2003), pp. 939–956.

[6] DE ALFARO, L., Model Checking the World Wide Web, *Lecture Notes in Computer Science*, **2102** (2001), pp. 337–350.

[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] CONFORTI, G. – GHELLI, G. – ALBANO, A. – COLAZZO, D. – MANGHI, P. – SARTIANI, C., The Query Language TQL, *Proc. of 5th International Workshop on Web and Databases (WebDB 2002)*, 2002.
[10] GOTTLOB, G. – KOCH, C., Monadic Queries over Tree-Structured Data, *In Proc. LICS'02*, Copenhagen, Denmark, 2002.
[11] XSL Transformations (XSLT), Version 1.0, *W3C Recommendation*, http://www.w3.org/TR/xslt, 1999.
[12] XQuery 1.0: An XML Query Language, *W3C Working Draft*, http://www.w3.org/TR/xquery/{#}nt-bnf, 2002.
[13] BEX, G. J. – MANETH, S. – NEVEN, F., A Formal Model for an Expressive Fragment of XSLT, *Lecture Notes in Computer Science*, **1861** (2000), pp. 1137–1152.
[14] XQuery 1.0 and XPath 2.0 Formal Semantics, *W3C Working Draft*, http://www.w3.org/TR/xquery-semantics/, 2003.
[15] XML Base, *W3C Recommendation*, http://www.w3.org/TR/xmlbase/, 2001.
[16] HTML 4.01 Specification, *W3C Recommendation*, http://www.w3.org/TR/html401/, 1999.
[17] RUZSA, I., *Introduction to Modern Logic*, Osiris Press, 2000, (in Hungarian).
[18] ULLMAN, J. D. – WIDOM, J., *A First Course in Database Systems*, Prentice Hall, Inc, 1997.
[19] CORMEN, T. H. – LEISERSON, C. E. – RIVEST, R. L., *Introduction to Algorithms*, MIT Press, 1990.
[20] DE GIACOMO, G. – LENZERINI, M., PDL-based Framework for Reasoning about Actions, In *Lecture Notes in Artificial Intelligence*, **992**, Springer Verlag, 1995, pp. 103–114.
[21] EMERSON, E. A., Temporal and Modal Logic, In J. van Leeuwen editor, *Handbook of Theoretical Computer Science*, Vol. 2, chapter 16, Elseiver Science Publisher B. V. 1990, pp. 995–1072.
[22] BAADER, F. – NUTT, W., Basic Description Logics, In the *Description Logic Handbook*, edited by F. Baader, D. Calvanese, D.L. McGuinness, D. Nardi, P.F. Patel-Schneider, Cambridge University Press, 2002, pp. 47–100.
[23] NARDI, D. – BRACHMAN, R. J., An Introduction to Description Logics, In the *Description Logic Handbook*, edited by F. Baader, D. Calvanese, D.L. McGuinness, D. Nardi, P.F. Patel-Schneider, Cambridge University Press, 2002, pp. 5–44.