A Fast and Efficient Method for Processing 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 investigates the possibility of realizing some Web document processing tasks in the context of modal, especially description logics, providing a precise theoretical framework with well-analyzable computational properties. A fragment of SHIQ description logic which can primarily be used in document processing is introduced. The paper also presents a linear time algorithm for model checking Web documents proving that the logical approach can compete even in efficiency with other industrial solutions.

1 Introduction

During the last ten years, the success of World Wide Web was increasing and it has become part of our daily life. Due to this enormous success, several techniques for processing, transforming or searching Web documents, like XML or HTML, have been developed. Unfortunately, these techniques are usually based on different theoretical approaches, no uniform representation is known. Primary consequence of different theoretical frameworks is that several parts of them are reinvented and reimplemented at each of the techniques. Hence, some of these frameworks are lack of simple formal semantics or efficient algorithms.

Description logics are simple logical formalisms which primarily focus on describing terminologies and graph style knowledge [1,2]. Therefore, they seem to be an adequate basis for developing a common computational environment for several Web document processing tasks [3].

The origin of this work was motivated by a Web filter project. Several elements of the project and logic presented in this paper were previously published in [4,5]. However, non of the algorithmic aspects were considered yet.

The reminder of this paper is organized as follows. The fragment of SHIQ, and some of its application areas are introduced in section 2. Section 3 presents the basic idea behind the model checking algorithm. Last but not least section 4 draws some conclusions.

M. Bubak et al. (Eds.): ICCS 2004, LNCS 3036, pp. 553–556, 2004.

2 A Logical Approach for Processing Web Documents

This section briefly introduces a fragment of SHIQ description logic, which fragment has primary importance in Web document processing. 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 a document is basically an ordered tree which nodes are associated with atomic predicates.

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

1. V is a set of nodes of the graph, AP is a set of atomic predicate, top∈ V is the top node.
2. *c*, *ap* and *n* binary relations describe the structure of an ordered tree which nodes are labeled by atomic predicates.

This definition seems natural for an XML document. For example, tags can be translated to nodes and embedding of tags represents the children relation. The definition is less trivial for an HTML document, consequently pre-transformations and pre-filters need to be applied.

Syntax and semantics of the logic are based on roles and concepts (Table 1.). In order to define a formal semantics of the syntax, an *I* interpretation function is considered, which assigns to every concept a set of nodes of a given 'd' document model and to every role a binary relation over V×V.

Table 1. Syntax and semantics of the logical framework.

Constructor	Syntax	Semantics
Concept Constructors		
atomic concept	a	$a^I = \{ v \in V \mid a \in ap(v)\}$
disjunction	or	$(C_1 \text{ or } C_2)^I = C_1^I \cup C_2^I$
conjunction	and	$(C_1 \text{ and } C_2)^I = C_1^I \cap C_2^I$
complement	not	$(\text{not } C)^I = V \setminus C^I$
universal quant.	all	$(\text{all } R.C)^I = \{v \in V \mid \forall w. <v,w> \in R^I \text{ implies } w \in C^I\}$
existential quant.	some	$(\text{some } R.C)^I = \{v \in V \mid \exists w. <v,w> \in R^I \text{ and } w \in C^I\}$
top concept	every	$\text{every}^I = V$
bottom concept	none	$\text{none}^I = \varnothing$
Role Constructors		
next role	next	$\text{next}^I = n$
children role	child	$\text{child}^I = c$
inverse role	inverse	$(\text{inverse } R)^I = \{<w,v> \in V \times V \mid <v,w> \in R^I\}$
transitive closure	infinite	$(\text{infinite } R)^I = \cup_{j>=1}(R^I)^j$

Using a logic in real life applications requires the existence of several basic reasoning services and efficient algorithms for computing these services. One of the most important and most efficient basic reasoning service is model checking but others like equivalence, querying or subsumption could also be used widely.

Basic reasoning services can be used in a wide variation of document processing tasks. Simple model checking is the basic reasoning mechanism of a searching process (e.g. searching in an XML database or searching the WWW). A logical expression could be the searching statement and documents, for which the evaluation of the statement is not an empty set, form the result of the search. Beside search, model checking can be used in several other areas, e.g. document categorization. In document transformation (e.g. XSLT, XQuery) or information extraction, the principal problem is to select some tags of the document which match with a predefined template. It is the most natural application area of querying because logical expressions can easily be regarded as templates. Last but not least, document checking (e.g. DTD) can be efficiently supported by subsumption or equivalence of model checking. For example, the following statement would be true for only those XML documents in which every slideshow tag contains only slide or title tags: 'slide $\Rightarrow$ all child.(title or slide)'.

3 Model Checking Algorithm

The model checking algorithm is based on the algebraic approach of the semantics. Expressions are interpreted as sets so concept constructors can be interpreted as operations between sets. For example, a conjunction can be regarded as a binary operation which transforms two input sets to an output one ('and': $2^V \times 2^V \rightarrow 2^V$). Similarly, universal or existential quantifications can be interpreted as unary operations associating input sets with output ones ('all child': $2^V \rightarrow 2^V$). Role constructors are manifested as variations in the unary operations. For instance, 'all child' represents a different unary operation as 'all infinite child' does.

The only question which highly effects efficiency is how to represent sets and relations of the document model. In our approach, nodes of the document model are labeled by integers in the [0….|V|-1] domain, where |V| denotes the cardinality of the node set. Each node has exactly one integer label. Primary consequence of this labeling is that most part of the algorithm can be built on hash tables and simplified hash joins.

The structure of the document is stored in five tables. For example, 'parenttable' is an array of integers associating each integer label of a node with the integer label of its parent node (according to the inverse of '*c*' binary relation of the document model). The algorithm implements a realization for each operation. As an example, we can consider 'some infinite child' operator which requires the identification of nodes that can be reached from a given set of nodes. It can be implemented by a depth-first search of the graph described by 'parenttable'. Since the number edges of the graph are linear in the size of nodes, depth first search runs linear time in the size of nodes of the document model.

This approach has the following important property:

Proposition.
If the number of possible atomic predicates of each node is bound, the model checking algorithm has O(l*|V|) time and space complexity (where l is the length of the logical expression, and |V| is the number of nodes of the document model).

Beside theoretical investigation, an experimental architecture has also been implemented in C# to test the concepts and algorithms between real circumstances. The architecture realizes XML and HTML parsers which load the administration tables directly and an algorithm for evaluating logical expressions over document models.

4 Conclusion

This paper analyzed the possibilities of using description logics in web document processing. It has identified a fragment of SHIQ which has primary importance in document processing and briefly introduced how specific document processing problems can be solved by this fragment. It has several benefits comparing to other industrial solutions of document processing. It provides a uniform knowledge representation with well defined syntax, semantics and algorithms, which representation is sometimes more expressive than industrial ones. Hence, description logic integrates several previously unrelated document processing problems like categorization or document checking into one common framework. Besides, the article introduced an efficient algorithm for evaluating logical expressions over Web documents. Since the algorithm is linear, it can compete even in efficiency with other industrial solutions.

References

1. 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) 47-100
2. Borgida, A., Brachman, R. J.: Conceptual Modeling with 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) 359-381
3. Calvanese, D., Giacomo, G., Lenzerini, M.: Representing and reasoning on XML documents: A description logic approach Journal of Logic and Computatio*n*, 9(3) (1999) 295-318
4. Szegő, D.: Using Description Logics in Web Document Processing, SOFSEM vol. II. (2004) 256-263
5. Szegő, D.: A Logical Framework for Analyzing Properties of Multimedia Web Documents, Workshop on Multimedia Discovery and Mining, ECML/PKDD-2003, (2003) 19-30.