TCS-TR-A-07-28

TCS Technical Report

Keyword Query Processing Using Binary Decision Diagrams under a Taxonomy Model

by

Shin-ichi Minato and Nicolas Spyratos

Division of Computer Science

Report Series A

August 9, 2007

Hokkaido University
Graduate School of
Information Science and Technology

Email: minato@ist.hokudai.ac.jp

Phone: +81-011-706-7682
Fax: +81-011-706-7682

Keyword Query Processing Using Binary Decision Diagrams under a Taxonomy Model

Shin-ichi Minato
Division of Computer Science
Hokkaido University
North 14, West 9, Sapporo 060-0814, Japan

Nicolas Spyratos
Laboratoire de Recherche en Informatique
Univ. Paris-Sud
91400 Orsay, France

August 9, 2007

(Abstract) We consider a publish/subscribe system for digital libraries which continuously evaluates queries over a large repository containing document descriptions. The subscriptions, the query expressions and the document descriptions, all rely on a taxonomy that is a hierarchically organized set of keywords, or terms. The digital library supports insertion, update and removal of a document. Each of these operations is seen as an event that must be notified only to those users whose subscriptions match the document's description.

In this paper, we present a novel method of processing such keyword queries. Our method is based on Binary Decision Diagram (BDD), an efficient data structure for manipulating large-scale Boolean functions. We compile the given keyword queries into a BDD under a taxonomy model. The number of possible keyword sets can be exponentially large, but the compiled BDD gives a compact representation, and matching process will become faster. In addition, our method can deal with any Boolean combination of keywords from the taxonomy, while our previous result considered only a conjunctive keyword set. In this paper, we describe the basic idea of our new method and show a preliminary experimental result.

1 Introduction

The publish/subscribe interaction paradigm provides subscribers with the ability to express their interest in classes of events generated by publishers. A system that supports this paradigm must be able to find, for each incoming event e, the subscriptions that match e, in order to determine which subscribers should be notified. Many typical web applications can be seen as variants of this general framework, including auction sites, on-line rental offices, virtual bookshops, etc. They act as brokers which store only descriptions of the published items.

We now consider a publish/subscribe system for digital libraries which continuously evaluates queries over a large repository containing document descriptions[3]. The subscriptions, the query expressions and the document descriptions, all rely on a taxonomy that is a hierarchically organized set of keywords, or terms. The digital library supports insertion, update and removal of a document. Each of these operations is seen as an event that must be notified only to those users whose subscriptions match the document's description.

In this paper, we present a novel method of processing such keyword queries. Our method is based on Binary Decision Diagram (BDD)[2], an efficient data structure for manipulating large-scale Boolean functions. We compile the given keyword queries into a BDD under a taxonomy model. The number of possible keyword sets can be exponentially large, but the compiled BDD gives a compact representation, and matching process will become faster. In addition, our method can deal with any Boolean combination of keywords from the taxonomy, while the previous result[3] considered only a conjunctive keyword set. In this paper, we describe the basic idea of our new method and show a preliminary experimental result.

2 Preliminaries

Here we describe the basic framework of subscription-notification service which we consider in this paper.

2.1 Subscription-Notification Service in Digital Library

Figure 1 shows the basic processing flow of our system. A document is represented in the digital library repository by a *description* of its content together with an identifier (say, the document's URI) allowing to access the document's content. A number of documents are incoming to the digital library every day, and the identifier list of recent new documents is notified to each user periodically (daily, weekly, or monthly). Each user can specify a *subscription* to show his/her interesting keywords. Every document has a description with a set of keywords, and if one of the keyword matches to some users's subscriptions, the identifier of the document is added to the notification list. Here, we assume that user's subscription is not so frequently updated as the notification periods.

If we consider a nation-wide digital library system, the number of documents and the number of users become quite huge, and it is an important problem to reduce the computation cost for matching (or filtering) of the documents to each user.

2.2 Keyword Query and Taxonomy Graph

A subscription represents the user's interesting topics with some keywords. In our previous work[3], due to technical limitation, we only assume the subscription as a set of keywords (i.e. a conjunctive query) for each user. However, now we consider any keyword query that allows all propositional logic operations, conjunction, disjunction and complement. For example,

```
( ("Languages" & "Sorting") | "Java" )
 & (! "C++")
```

represents that the user likes to see the documents including the keyword *Languages* and *Sorting* together, or having the keyword *Java*, provided that the document is not related to *C++*.

In addition, we consider the hierarchical taxonomy of the keywords. Figure 2 shows an example of taxonomy graph. In this graph, upper keywords represent more general concept of its descendants. For example, if a user subscribes a keyword *Algorithms*, the documents having any one of keywords *Algorithms, Sorting, Merge, Quick* and *Bubble* will be notified to the user. In this example, the taxonomy graph has a tree structure, but in general, some sub-trees can be shared with each other, and the taxonomy graph can be a directed acyclic graph. Employing such a hierarchical taxonomy graph will greatly improve the usability of the system, but the computation cost may grow larger.

2.3 Binary Decision Diagrams (BDDs)

In this paper, we discuss a new method of keyword query processing based on Binary Decision Diagrams (BDDs). Here we briefly describe our data structure.

A *Reduced Ordered Binary Decision Diagram* (ROBDD)[2] is a compact graph representation of the Boolean function. It is derived by reducing a binary tree graph representing the recursive *Shannon expansion.* ROBDDs provide canonical forms for Boolean functions when the variable order is fixed. (In the following sections, we basically omit "RO" from BDDs.) As shown in Fig. 3, a set of multiple BDDs can be shared with each other under the same fixed variable ordering. In this way, we can handle a number of Boolean functions simultaneously in a monolithic memory space.

A conventional BDD package supports a set of basic logic operations (i.e., AND, OR, XOR) for given a pair of operand BDDs. Those operation algorithms are based on hash table techniques, and the computation time is almost linear with data size unless the data overflows main memory. By using those inter-BDD operations, we can generate BDDs for given Boolean expressions or logic circuits. (See [2] for more details.)

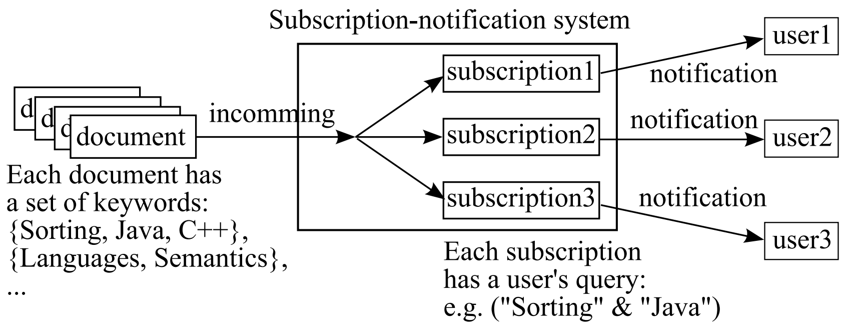

Figure 1: Basic Processing Flow of Subscription-Notification Service.

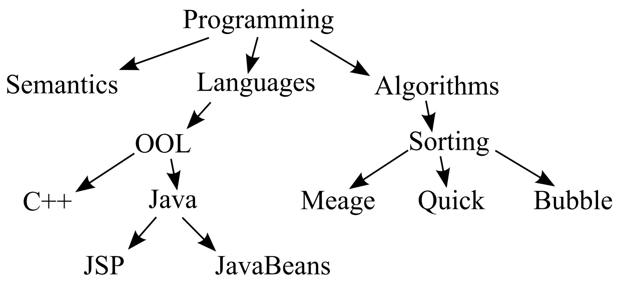

Figure 2: An example of taxonomy graph.

3 BDD-Based Document Filtering

In the subscription-notification system, each document is determined whether it matches or not to respective user subscriptions. If we employ the keywords based on the hierarchical taxonomy, the matching procedure becomes complicated. In order to accelerate the matching procedure, we propose a BDD-based method to generate a complete document filter for each user under a structural taxonomy.

3.1 BDD Generation for a Keyword Query

Under fixed variable ordering, a BDD can be generated from the Boolean expression of the user's query. The algorithm is completely same as one to generate BDDs from given logic circuits, which has been developed in the area of VLSI CAD. For example, the query:

```
( ("Languages" & "Sorting") | "Java" )
 & (! "C++")
```

produces a BDD as shown in Fig. 4. In this BDD, each path from the root node to 1-terminal node corresponds to a possible keyword pattern to be notified to the user. When we check a document to be notified, we may traverse the BDD downward from the root node to the terminal nodes, by choosing one of subgraph as not contradicting the keyword set of the document. Eventually we will arrive at either 0- or 1-terminal node and it tells us the result. The computation time is linearly bounded by the height of the BDD, not directly depends on the number of BDD nodes. In other words, the Boolean expression of the user's query is compiled into a logic circuit with an efficient BDD-based structure.

3.2 Expansion of BDDs based on Taxonomy Graph

Employing the hierarchical taxonomy, we expand the BDD for the user query to take into account all implications of multiple keywords. The method

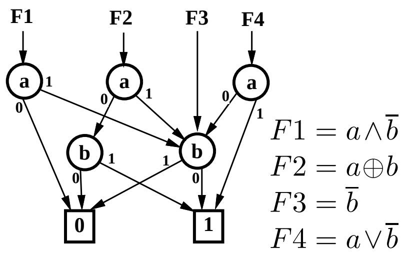

Figure 3: Shared multiple BDDs.

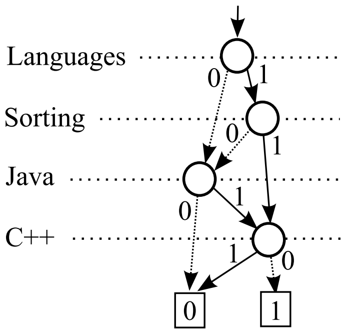

Figure 4: A BDD representing user's query.

is illustrated in Fig. 5. First we look at the top keyword *Programming* in the taxonomy graph. *Programming* has three children *Semantics, Languages*, and *Algorithms*, so we may accept a document with either one of three keywords instead of *Programming*. This can be performed by logic operations as follows. We divide the query function F into the two subfunction F_{off} and F_{on}, as:

```
F = (! "Programming" & F_off )
  | ("Programming" & F_on )
```

and then we replace "Programming" with the set of four keywords as:

```
F = (! "Programming" & F_off )
  | ( ("Programming" | "Semantics"
    | "Languages" | "Algorithms") & F_on ).
```

In this way, the keyword patterns related to *Programming* are expanded. Similarly we expand the BDD for all the other keywords from the top to the bottom in the taxonomy graph. Finally we can obtain the BDD to determine the documents considering all implications of keywords under the taxonomy graph. If we can generate such a BDD in a feasible memory size, the information of taxonomy structure is already compiled into the BDD-style logic circuit. It will be a very powerful data structure.

3.3 Combining Multiple Users' Queries

Our subscription-notification system assumes multiple users. We may combine a number of users' queries into a single BDD as follows. To handle total m users, we prepare the special variables as many as $\lfloor \log_2 m \rfloor$, to distinguish respective users by using a binary code. For example, we prepare the three variables $u1, u2$, and $u3$ for $m = 8$, and we combine the logic expression for the queries $F0, F1, \ldots F7$ as:

```
F = ( !u1 & !u2 & !u3 & F0 )
  | ( !u1 & !u2 &  u3 & F1 )
  | ( !u1 &  u2 & !u3 & F2 )
 ...
  | (  u1 &  u2 &  u3 & F7 ) .
```

The BDD for this logic expression is shown in Fig. 6.

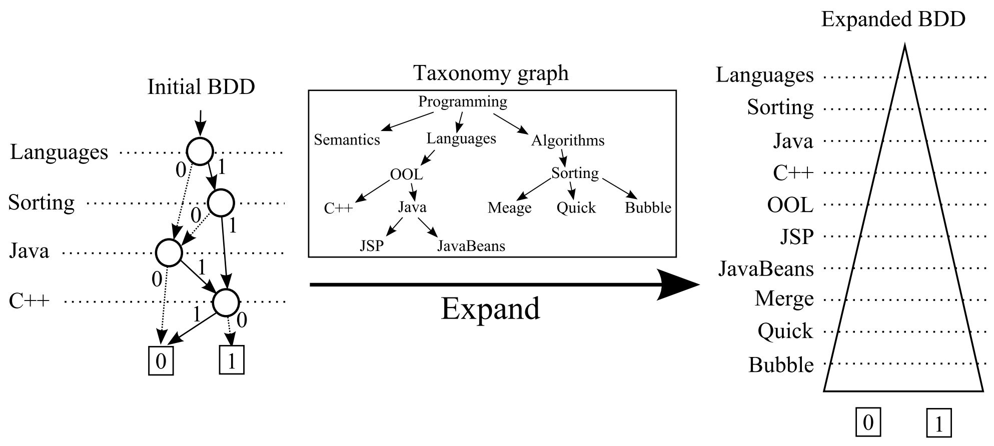

Figure 5: BDD expansion based on a taxonomy graph.

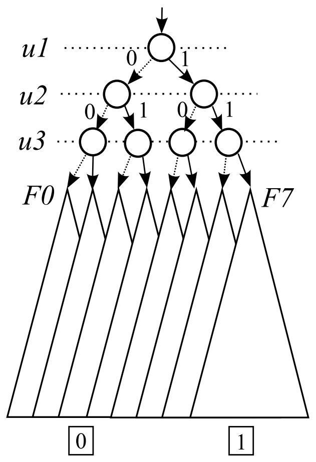

Figure 6: BDD for multiple users queries.

We can obtain various information by manipulating this BDD. For example, when we assign 0 or 1 for all keyword variables using the description of a document, then the variable $u1, u2$, and $u3$ remain in the BDD. Such a BDD represents the set of users who accept the document with the keyword pattern assigned now.

On the other hand, if we compute a union (logical OR) of the two subgraph of the top node with the variable $u1$, the result of BDD represents a filtering circuit to know the keyword patterns which are accepted no matter how is the value of $u1$. By applying the same operation to all the user variables, we can obtain a BDD to show the set of keyword patterns such that the document is accepted by at least one user.

4 Experimental Result

To evaluate our new method, we conducted preliminary experiments to generate BDDs for given user queries under a small taxonomy graph. We assume the taxonomy graph shown in Fig. 2, which consists of 13 keywords. In this experiments, we used BEM-II[4], a Boolean expression manipulator based on our own BDD package. This program can manipulate up to 30,000,000 nodes of BDDs on a Linux PC with 2GB memory.

At first we generate a BDD for the simplest user query:

```
F = "Programming" .
```

The BDD has only one decision node with the vari-

```
symbol prog sem lang ool cpp java jsp jb alg sort merg qk bbl

F = ( (lang & sort) | java ) & (! cpp)

F = (!prog & F[!prog]) | ((prog | sem | lang | alg) & F[prog])
F = (!lang & F[!lang]) | ((lang | ool) & F[lang])
F = (!alg & F[!alg]) | ((alg | sort) F[alg])
F = (!ool & F[!ool]) | ((ool | cpp | java) & F[ool])
F = (!java & F[!java] | ((java | jsp | jb) & F[java])
F = (!sort & F[!sort]) | ((sort | merg | qk | bbl) & F[sort])

print /size F
```

Figure 7: BEM-II script for generating expanded BDD.

able for the keyword. We then apply expansion using the taxonomy graph. The expanded BDD includes 13 decision nodes, and it can accept the set of 8,191 keyword patterns, that means all documents with at least one keyword are accepted in this BDD.

Next, we generate a BDD for the query:

```
F = ( ("Languages" & "Sorting") | "Java" )
  & (! "C++") .
```

Before expansion, the BDD for F consists of 6 decision nodes and it accepts 2,560 patterns of keyword combinations. After expansion with the taxonomy, The BDD grows as 14 decision nodes and accepts 3,944 keyword patterns. Total computation time is too small to evaluate. (less than 0.1 second.)

The script of BEM-II program to generate the expanded BDD is shown in Fig. 7. The total number of logic operations (i.e. the length of the script) is linearly bounded by the size of taxonomy graph.

5 Summary

In this paper, we presented a new method of processing keyword queries using BDDs. The number of possible keyword sets can be exponentially large, but the compiled BDD gives a compact representation, and matching process will become faster. In addition, our method can deal with any Boolean combination of keywords, while the previous result[3] considered only a conjunctive keyword set.

We presented preliminary experimental results that thousands of possible keyword patterns can be represented compactly by a BDD-based filtering circuit. Current examples are too small to evaluate the computation time of our method, so, future we will apply our method to real-life document sets with large-scale keyword domain under a common taxonomy structure such as ACM Computing Classification System[1]. Our future work also includes to consider more advanced keyword expansion, for example, to define *distance* of keywords in taxonomy graph and provide a limited keyword expansion. It will be also interesting work to provide *sorted* output of documents under some preference of users to avoid “spam” notifications.

References

[1] The ACM computing classification system, 1999. `www.acm.org/class`.

[2] R. E. Bryant. Graph-based algorithms for Boolean function manipulation. *IEEE Transactions on Computers*, C-35(8):677–691, 1986.

[3] H. Belhaj Frej, Ph. Rigaux, and N. Spyratos. User notification in taxonomy based digital libraries (invited paper). In *Proc. of ACM SIG-DOC Conference on the Design of Communi-*

cation, Myrtle Beach SC, U.S.A., pages 18–20, 2006.

[4] S. Minato. BEM-II: An arithmetic boolean expression manipulator using BDDs. *IEICE Trans. Fundamentals*, E76-A(10):1721–1729, 1993.