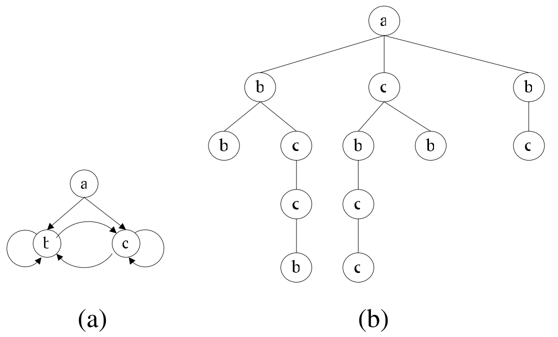
Section 3 and Section 4 propose the HFilter strategy, discuss its two-tier and three-tier implementations. Section 5 shows the performance analysis and comparison. Section 6 surveys related works. Section 7 presents our conclusions and discusses future works.

2 Background

Fig. 2 (a) is an example of a recursive DTD. Fig. 2 (b) shows an instance of this DTD. Fig. 2 (c) gives a sample query set, *Q*.

In this paper, we define *location level* of an element as its position in a path from root. For example, in the path {*/a/b/b*}, the *location level* of root element */a* is 1, the *location level* of the first */b* in this path is 2, and that of the second */b* is 3.

In XML stream processing systems, queries, instead of the data, are usually indexed. Two typical finite automaton based filtering engines are widely adopted, which are NFA based [1, 2, 3, 4, 9, 10, 13] and lazy DFA based [5, 6, 7, 8, 12].



Q:
/a//b
/a//c
/a/*/b/c/c
/a/*/c/c/b

(c)

Fig. 2. (a) DTD graph, (b) an XML instance of (a), (c) a query set *Q*

2.1 NFA Based Filtering Engine

Considering the prefix commonality in the query set, an NFA based filtering engine constructs the NFA as the query index from the query set. For each query, an NFA will be constructed, and all these NFAs will be combined into a single NFA [2, 4]. Fig. 3 shows an example of a combined NFA for the query set *Q* shown in Fig. 2(c).

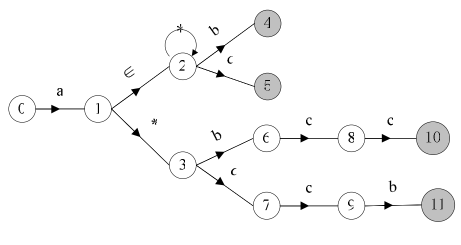


Fig. 3. The combined NFA for *Q* shown in Fig. 2 (c)

For the NFA based filtering engine, a special runtime stack is maintained. Since the NFA may have many active states at the same time, the stack should be able to track the multiple active paths.

Example 1. To process the path {*/a/b/b*} in the XML instance shown in Fig. 2 (b), the initial top state set of the runtime stack is {0}. After the "start-of-element" event of the root */a* has been processed, the top state set is {1, 2}. Then after the "start-of-element" event of the first */b* has been processed, the top state set is {2, 3, 4}. And after processing the second */b*, the top state set is {2, 4, 6}. Then the following events are three "end-of-element" events and the top three state sets are popped.

The characteristics of NFA based approaches can be described as follows:

- **Advantages:** The NFA consumes a very small amount of main memory.
- **Disadvantages:** This mechanism always needs to compute the top set of the active states. Many of the state sets are repeatedly computed, which costs a lot of execution time.

2.2 Lazy DFA Based Filtering Engine

In general, for a DFA based filtering engine, constructing the eager DFA will result in exponential number of states and lead to memory overflow. Therefore, the DFA is constructed in a lazy way in most cases and called the lazy DFA [5].

For the lazy DFA based filtering engine, a special query index is maintained, while a special runtime stack is maintained in NFA. Every DFA state of lazy DFA contains an NFA state table and thus new DFA states can be computed at the runtime other than pre-computed before filtering and save the physical memory. The runtime stack of lazy DFA always pushes or pops only one DFA state.

The filtering engine memorizes all sets of the active NFA states which have been encountered by inserting new DFA states into the lazy DFA. When the same set of active NFA states are needed next time, the engine can get the target DFA state immediately which leads to a high throughput.

Example 2. When processing the path {*/a/b/b*} using lazy DFA three sets of active states will be generated: {1, 2}, {2, 3, 4} and {2, 4, 6}. Fig. 4 shows an example of a lazy DFA: the lazy DFA after processing the whole XML instance shown in Fig. 2 (b). The NFA state ID sets in the eclipses in Fig. 4 are NFA tables.

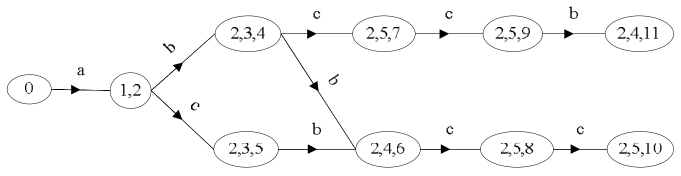


Fig. 4. An example of lazy DFA

The lazy DFA engine has a warming up phase at the beginning of the filtering and enters a stable phase later. During the warming up phase, the engine spends a lot of time in expanding new lazy DFA states. Therefore, the runtime filtering performance is much worse than that during the stable phase.