Streamlining of DFA based Pattern Matchers

Ashish Kumar Pandey
Assistant Professor,
AIMT, Lucknow
Department of
Computer Science

Amit Sinha
Technical Lead
Analyze Infotech Pvt. Ltd

ABSTRACT
This paper presents an efficient algorithm for finding matches to a given regular expression in given text using optimization of DFA. To match a regular expression of length n, a serial machine requires 0(2^n) memory and takes 0(1) time per text character. The proposed approach requires only 0(n^2) space and still process a text character in 0(1) time (one clock cycle).The improvement is due to the optimization of DFA that means without converting it into the NFA, directly convert into the DFA. Finite Automaton (DFA) used to perform the matching. Furthermore, the paper presents a simple, fast algorithm that quickly constructs the DFA for the given regular expression.

Keywords
DFA, NFA

1. INTRODUCTION
A regular expression, often called a pattern, is an expression used to specify a set of strings required for a particular purpose. Regular expressions are widely supported in programming languages, text processing programs (particular lexers), advanced text editors, and some other programs. A regular expression processor translates a regular expression into a nondeterministic finite automaton (NFA), which is then made deterministic and run on the target text string to recognize substrings that match the regular expression. The pattern sequence itself is an expression that is a statement in a language designed specifically to represent prescribed targets in the most concise and flexible way to direct the automation of text processing of general text files, specific textual forms, or of random input strings.This paper presents an efficient algorithm for finding matches to a given regular expression in given text using optimization of DFA.

2. BACKGROUND
followpos(i) : is the set of positions which can follow the position i in the strings generated by the augmented regular expression.

followpos is just defined for leaves,it is not defined for inner nodes.

Computing Followpos:
A position of a regular expression can follow another position in two ways:

Rule 1: if n is a cat-nodec1c2

For every position in lastpos(c1) all positions in firstpos(c2) are in followpos(i).

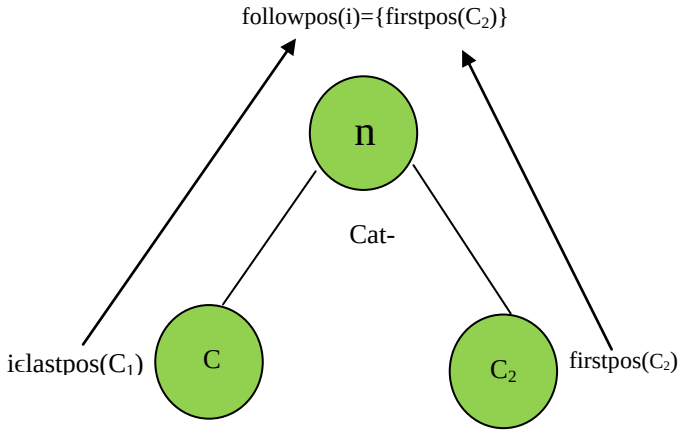

Figure 1: Cat -Node

Rule 2:if n is a star-node
If i is a position in lastpos(n) then all positions in firstpos(n) are in followpos(i).

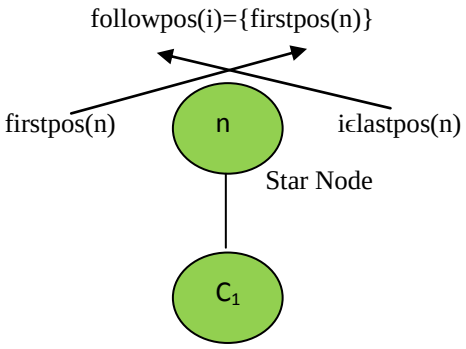

Figure 2: Star-Node

Evaluation of firstpos, lastpos, nullable

To evaluate followpos, three more functions are to be definedfor the nodes (not just for leaves) of the syntax tree.

nullable(n) true if the empty string is a member of strings generated by the sub-expression rooted by n false otherwise

firstpos(n)the set of the positions of the first symbols of strings generated by the sub-expression rooted at n.

lastpos(n) the set of the positions of the last symbols of strings generated by the sub-expression rooted at n.

Table:1Evaluation of firstpos, lastpos, nullable

Node n	Nullable(n)	Firstpos(n)	Lastpos(n)
A leaf labeled ε	True	Ø	Ø
A leaf with position i	False	{i}	{i}
An or-node n=c1|c2	Nullable(c1) or nullable(c2)	Firstpos(c1) U firstpos(c2)	Lastpos(c1) U lastpos(c2)
A cat-noden=c1c2	Nullable(c1) and nullable(c2)	If (nullable(c1)) Firstpos(c1)U firstpos(c2) Else firstpos(c1)	If(nullable(c2)) Lastpos(c2) U lastpos(c1) Elselastpos(c2)
A star-noden=c1*	True	Firstpos(c1)	Lastpos(c1)

Need Of Pattern Matching

Pattern matching is the process of checking a perceived sequence of string for the presence of the constituents of somepattern. In contrast to pattern recognition, the match usually has to be exact. The patterns generally have the form sequencesof pattern matching include outputting the locations of a pattern within a string sequence, to output some component of thematched pattern, and to substitute the matching pattern with some other string sequence (i.e., search and replace). Patternmatching concept is used in many applications Following figure shows the different applications.

In pattern matching I focused on the regular expression amongst others application.Pattern matching will help to find right and appropriate result. Hence I proposed Streamlining of DFA Based Pattern Matchers algorithm to match the Pattern.

Figure 3: Applications of Pattern Matching

3. PROPOSED ALGORITHM

Algorithm to make RE to DFA:

1. Create syntax tree with augmented rerular expression.
2. Assign number to each terminal from left to right.If there is any epsilon as a terminal,do not assign any number.
3. Calculate the functions: followpos, firstpos, lastpos, nullable.
4. Put S(start node)= firstpos (root of augmented syntax tree) of DFA as an unmarked state.
5. Mark S and make it start state.
6. Now we give the input to the start state which is already marked.
7. Input is provided according to numbers which are in given states.
8. If terminals are defined in more than one position then find the followpos of those terminals on different positions and do the union.
9. Now we find out the next state which is move(S,a) = S'(next state which can also be final state).
10. If the next state is new state then go to step[7].
11. If the next state is repeated state and no new state is found then stop process.
12. To make DFA according to move fuction.
13. The accepting states of DFA are all states containing the position of #.

Example: Augmented Regular Expression

(a | b) * a #

red – firstpos, blue – lastpos

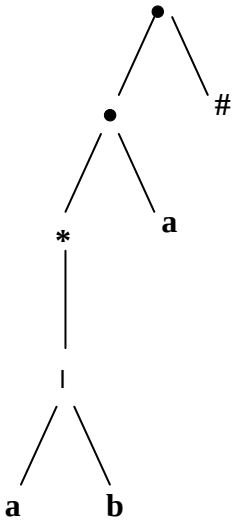

Figure 4: Syntax tree with augmented regular expression

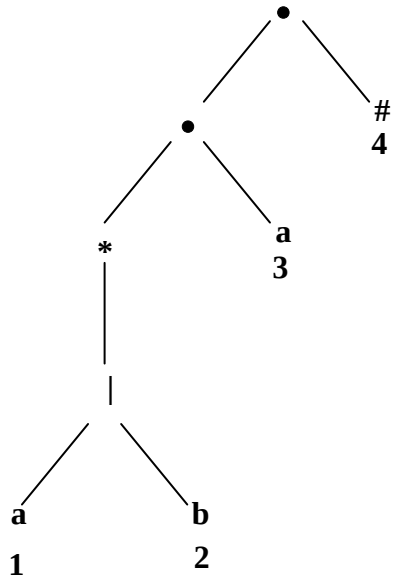

Figure 5: Assign number to each terminal from left to right

Calculate the functions: followpos, firstpos, lastpos, nullable

Calculate firstpos(i) and lastpos(i) for leaf i:

For i=1,

firstpos(1): {1}

lastpos (1): {1}

For i=2,

firstpos(2): {2}

lastpos (2): {2}

For i=3,

firstpos(3): {3}

lastpos (3): {3}

For i=4,

firstpos(4): {4}

lastpos (4): {4}

Calculate firstpos(i) and lastpos(i) for or, cat and star node:

For or node(|):

firstpos(|): {1,2}

lastpos(|): {1,2}

For star node(*):

firstpos(*): {1,2}

lastpos(*): {1,2}

For cat node(.):

For leaf a:

firstpos(.): {1,2,3}

lastpos(.): {3}

For leaf #:

firstpos(.): {1,2,3}

lastpos(.): {4}

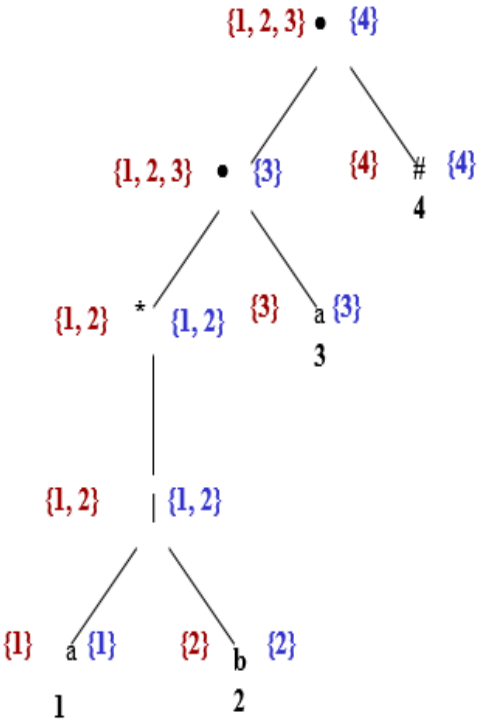

Figure 6: Augmented ST with computed values

followpos(1) = {1,2,3}

followpos(2) = {1,2,3}

followpos(3) = {4}

followpos(4) = {}

S_1=firstpos(root)={1,2,3}

⇓ mark S_1

a: followpos(1) ∪ followpos(3)={1,2,3,4}= S_2, move(S_1,a)=S_2

b: followpos(2)={1,2,3}=S_1, move(S_1,b)=S_1

⇓ mark S_2

a: followpos(1) ∪ followpos(3)={1,2,3,4}= S_2, move(S_2,a)=S_2

b: followpos(2)={1,2,3}=S_1move(S_2,b)=S_1

start state: S_1

accepting states: {S_2}

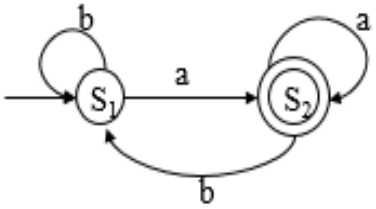

Figure 7: Deterministic finite automata

4. CONCLUSION

In this paper, we proposed an efficient regular expression algorithm to match the regular expression without converting into NFA.Which results the reduced area, better performance, less number of resources.To match a regular expression of length n, a serial machine requires 0(2^n) memory and takes 0(1) time per text character. The proposed algorithmrequires only 0(n^2) space and still process a text character in 0(1) time (one clock cycle).The improvement is due to the optimization of DFA that means without converting it into the NFA, directly convert into the DFA. The future scope of the work is to improve the efficiency of the finite state machine.

5. REFERENCES

[1] J. E. Hopcroft and J. D. Ullman, Introduction to automata theory,languages, and computation," *Addison Wesley*, 1979.

[2] Y. Sun, H. Liu, V. Valgenti, and M. S. Kim, "Hybrid regular expression matching for deep packet inspection on multi-core architecture," in *Proceedings of the 19th International onference on Computer Communications and Networks, ICCCN'10*, Aug. 2010, pp. 1–7.

[3] P.Jayaprabha and Rm. Somasundaram ,"Content Based Image Retrieval Methods Using Graphical Image Retrieval Algorithm (GIRA)"., Computer Science And Application, Vol. 1, No. 1, January, 2012.

[4]1997,http://www.igm.univmlv.fr/~lecroq/string/index.html C. Charras and T. Lecroq: Exact String MatchingAlgorithms. Univ. de Rouen.

[5] Srikanthan, Sharanyan ,"Implementing the dynamic time warping algorithm in multithreaded environments for realtime and unsupervised pattern discovery"., Computer and Communication Technology (ICCCT), 2011 2nd International Conference on 394 – 398, 15-17 Sept. 2011.

[6] Stan Salvador & Philip Chan,"Fast DTW: Toward Accurate Dynamic Time Warping in Linear Time and Space".KDDWorkshop on Mining Temporal and Sequential Data, pp. 70-80, 2004. (http://cs.fit.edu/~pkc/papers/tdm04.pdf).

[7] Chu, S., E. Keogh, D. Hart & Michael Pazzani. Iterative, Deepening Dynamic Time Warping for Time Series. In Proc.of the Second SIAM Intl. Conf. on Data Mining. Arlington, Virginia, 2002.

[8] Arcangel, Cory. "On Compression". Retrieved 6 March 2013.

[9] Jaiswal, R.C. Audio-Video Engineering. Pune, Maharashtra: Nirali Prakashan. p. 3.41. (2009).ISBN 9788190639675.

[10] "Video Coding". Center for Signal and Information Processing Research. Georgia Institute of Technology. Retrieved 6March 2013.