A Structural Construction of a Deterministic Position Automaton

O.V.Shanmuga Sundaram,
Asst.Prof in Mathematics
Sri Shakthi Institute of Engg & Tech,
L & T Byepass Road, Coimbatore, India – 641 062

N.Murugesan,
Asst.Prof in Mathematics
Govt. Arts College (Autonomous)
Coimbatore, TN, India – 641 018

ABSTRACT

Every regular expression can be transformed into a Non-deterministic Finite Automaton (NFA) with or without ε -transitions. A well known algorithm called subset construction technique is used for conversion of NFA into DFA. In this paper, initially, the construction of the position automaton is given for the same. Also, the algorithm to convert this Position Automaton into DFA using subset construction technique is discussed.

AMS MSC2010 Certification: 68Q45, 68Q70

Keywords

Regular expressions, Position automaton, subset construction, DFA and NFA.

1. INTRODUCTION

Let A be a finite alphabet and A^* denote the set of all finite strings of symbols in A. The empty string is denoted by ε. Any subset of A^* is a language over A. A regular expression is a declarative way of defining a regular language [9]. Because of the simplest and clear syntax, the regular expressions are the common choice to represent the regular languages. The regular expression can be built from the symbols in A and the special symbols "$\varnothing$", "ε", using the binary operations "+" and "." and the unary operation "*". Parentheses are used to indicate grouping, the operators "+" and "." are written in infix notation, "*" is written in postfix notation. Given a regular expression E over an alphabet A, we write $L(E)$ for the subset of A^*, that is denoted by E. The size of regular expression E, denoted $|E|$, is the number of symbols occurred from A in E.

A finite automaton is the simplest abstract computing device, and has mainly three variations, such as Deterministic finite automata (DFA), Non-deterministic finite automata (NFA) and NFA with ε-moves. In DFA, for every input symbol of the given alphabet, next move is uniquely determined. In NFA, at any stage, different choices may be given from the set of moves. In ε-NFA, automaton has the power to change the state without the change of head position.

Most importance given in the automata theory is regular expression. Regular expressions have become the basis of standard utilities such as scanner generators, editors, or programming languages. The recent researchers are using regular expressions as additional tool to reduce the programming effort. Implementations of regular expressions can be done using finite automata.

Glushkov [5] and McNaughton and Yamada [8] independently proposed Position Automaton. Their algorithm starts with linearized form of the regular expressions. The symbols are given as positions in the regular expression E. The number of states of the position automaton is the number of occurrences of symbols in E plus initial state. Many well known researchers have been proposed to obtain this position automaton with quadratic time complexity, like Champarnaud and Ziadi [2, 3], Chang and Paige [4] and Bruggermann–Klein [1]. Their NFA construction size was between linear and quadratic time.

In this paper, initially, a Glushkov position automaton is constructed, using Glushkov functions for a given regular expressions. Then, this Glushkov NFA is converted into DFA using subset construction. In this DFA, subset construction involves all subsets of the set of states of Glushkov NFA. The illustrations with some examples are discussed.

2. BASIC NOTIONS

The languages that are recognized by the various forms of automata are called regular languages. Here the discussion is about the regular expressions. This is a declarative way of defining regular languages [9]. Before giving the exact definition of regular expressions, an important notion is introduced. i.e., Kleene closure or simply closure L^* of a language L. L^* is nothing but the set of all strings that can be formed by taking any number of strings even with repetitions and taking concatenation of all them. For better undertaking of L^*, consider the language $L=\{0,1\}$. $L^0=\{\varepsilon\}$; And for i>1, L^i is the strings by getting after the concatenation of i copies of L are defined. For example,

$$L^1=\{0,1\};\ L^2=\{00,10,01,11\}\ \ and$$
$$L^3=\{000,001,010,100,110,101,011,111\}.....,and\ so\ on.$$

The Kleene closure is called by the notation $L^*=\bigcup_{i=0}^{\infty}L^i$

The regular expressions over the given set A^* of input symbols are described as follows:

1. The empty string ε is a regular expression
2. Any symbol of A is a regular expression.
3. If a is a regular expression, then, the occurrences of zero or more times of a and (a) are also regular expressions.
4. If a and b are regular expressions, then a + b, ab are also regular expressions.

The language for each regular expression is represented. In the case of ε, the language $L(\varepsilon)=\{\varepsilon\}$. Similarly, the language of a is $L(a)=\{a\}$. The languages of the regular expressions a + b and ab are $L(a)\cup L(b)$ and $L(a)L(b)$ respectively. The language of the regular expression a^* is $(L(a))^*$.

The following are some simple regular expressions and the language they represent as in the construction given.

(i). a^* – It denotes the language of strings of a's that have any number of a's. i.e, $L(a^*)=\{\varepsilon,a,aa,aaa,....\}$ the automaton accepting this language is in the fig .1

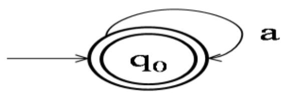


Fig.1. The automaton accepting a^* .

(ii). ab^* – It denotes the language of strings of a's and b's that begin with a single a followed by any number of b's. i.e., $L(ab^*)=\{a,ab,abb,abbb,....\}$. The automaton accepting this language is in the fig 2.

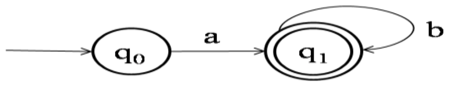


Fig.2. The automaton accepting ab^* .

(iii). a + b – It denotes the language consisting the strings a and b. $L(a+b)=L(a)\cup L(b)=\{a\}\cup\{b\}=\{a,b\}$. The automaton accepting this language is given in the following fig 3.

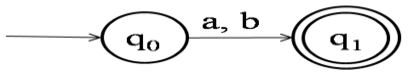


Fig.3. The automaton accepting a + b.

(iv). $(a+b)^*$ – It represents the language of strings of a's and b's consisting either a's or b's or both occur at any number of times. i.e., The corresponding automaton is given below in fig.4.

$$L\left((a+b)^*\right)=\begin{Bmatrix}\varepsilon,a,b,aa,ab,ba,bb,aaa,aab,aba,\\ baa,bba,bab,abb,bbb,...\end{Bmatrix}$$

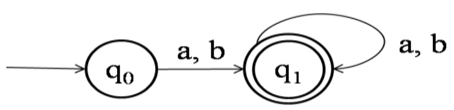


Fig.4. The automaton accepting $(a+b)^*$.

(v). $b^*\left(ab^*ab^*\right)^*$ – It represents the language of a's and b's consisting even number of as, because the occurrence of aa is must except in the case of ε and 1's may come at any number of times including zero times at any place in aa. The automaton accepting this language is in fig.5

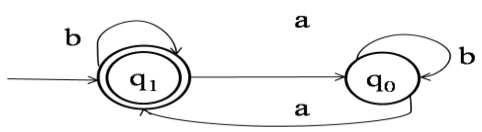


Fig.5. The automaton accepting $b^*\left(ab^*ab^*\right)^*$.

2. POSITION AUTOMATA

Position automata basically a non-deterministic finite state automaton discussed by Glushkov [5] and McNaughton and Yamada [8]. The basic principle behind this construction is that the occurrence of each symbol in a regular expression is considered as a different symbol by making with a unique index.

Let E be a regular expression over the set of symbols A. The set of positions of E is defined as $pos(E)=\{1,2,...,|E|_A\}$. $Pos_0(E)=Pos(E)\cup\{0\}$. The expressions obtained from E after rewriting with its position is denoted as $\overline{E}\in\overline{A},\ where\ \overline{A}=\{a_i\,|\,a\in A,1\le i\le|E|_A\}$.

For example, if $E=aa(ba+aba^*)$, then, $\overline{E}=a_1a_2(b_3a_4+a_5b_6a_7{}^*)$. For a regular expression E, the three mappings, First, Last and Follow are defined as follows:

$$First(E)=\left\{i\,\middle|\,a_iw\in L(\overline{E})\right\},$$

$$Last(E)=\left\{i\,\middle|\,wa_i\in L(\overline{E})\right\},$$

$$Follow(E,i)=\left\{j\,\middle|\,ua_ia_jv\in L(\overline{E})\right\}.$$

Let h be the alphabetic mapping from $pos(E)$ to A such that $h(x_i)=x,\ \ 1\le i\le|E|_A$, and $h(E^{'})=E$.

In position automaton, the subscripted symbols after applying this alphabetic mapping would become as follows:

$$h(a_1)=h(a_2)=a,\ and\ \ h(b_3)=h(b_4)=b\ \ etc..,$$

Position automaton of a regular expression E is the automaton

$P_E=(Pos_0(E),A,\delta,Last(E))$ such that

$$\delta(0,a)=\{x\in First(E)\,|\,h(x)=a\},\ \forall a\in A$$

$$\delta(x,a)=\{y \mid y \in Follow(E,x)\, and\, h(y)=a\}, \quad \forall x \in Pos_0(E), \forall a \in A .$$

2.1 Example:

For regular expression $E=(aa+b)^*(bb+a)^*$, the state diagram of position automaton is constructed as in fig. 6. The linearized expression of this regular expression is $E'=(a_1a_2+b_3)^*(b_4b_5+a_6)^*$. The position sets are calculated as below:

$$E=(aa+b)^*(bb+a)^*$$
$$\Rightarrow E'=(a_1a_2+b_3)^*(b_4b_5+a_6)^*$$
$$First(E')=\{a_1,b_3,b_4,a_6\};$$
$$Last(E')=\{a_2,b_3,b_5,a_6\};$$
$$Follow\{E',a_1\}=\{a_2\};$$
$$Follow\{E',a_2\}=\{a_1,b_3,b_4,a_6\};$$
$$Follow\{E',b_3\}=\{a_1,b_3,b_4,a_6\};$$
$$Follow\{E',b_4\}=\{b_5\};$$
$$Follow\{E',b_5\}=\{b_4,a_6\};$$
$$Follow\{E',a_6\}=\{b_4,a_6\};$$

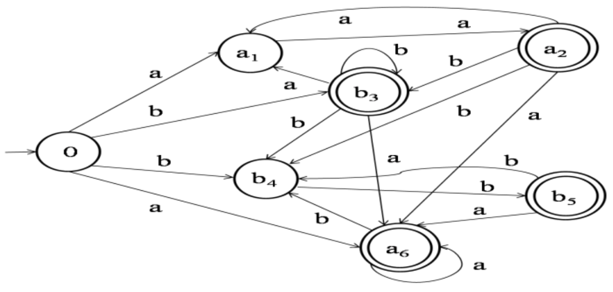


Fig.6 Position automaton for $E=(aa+b)^*(bb+a)^*$

3. DETERMINISTIC POSITION AUTOMATA

In general, the construction of NFA is simpler than the construction of DFA. For every NFA, there is an equivalent DFA that accepts the same language as the language acceptance of NFA. For this equivalence of these two automata, many authors were using one general algorithm, called the subset construction algorithm. Using this algorithm, one can construct DFA with the help of NFA states. Initially, all the subsets of all states of NFA are found and computed. Suppose if there are n states of NFA, then the DFA may have larger number of 2^n states in worst case. Otherwise the number of states of DFA may be very smaller than 2^n-1 (empty set avoided), because some inaccessible states from starting state may be eliminated.

In this paper, the Deterministic Position Automata (DPA) is constructed that is of NFA type, the modified definition of Glushkov position automaton is given.

In the definition of position automaton, initially, it has $Pos_0(E)=Pos(E)\cup\{0\}$. As the convenience, {0} is taken as $\{q_0\}$, which is defined as starting state in DPA.

Now, Glushkov NFA (Position automaton) is defined as

$$G=(Q_G,A,\delta_G,q_0,F_G),$$

Where

Q_G - the set of states (positions of E) plus start state $\{q_0\}$ i.e., $Pos(E)\cup\{q_0\}$

A - the set of alphabet symbols.

δ_G - the transition function

$$\delta_G(q_0,a)=\{x \in First(E) \mid h(x)=a\}, \ \forall a \in A$$
$$\delta_G(x,a)=\{y \mid y \in Follow(E,x)\, and\, h(y)=a\}, \quad \forall x \in Pos_0(E), \forall a \in A .$$

F_G - Final states

$$= F_G=\begin{cases}\{Last(E)\}\cup(q_0) & if\ \varepsilon \in L(G)\\ \{Last(E)\}, & otherwise\end{cases}.$$

Procedure of subset construction:

1. Start with NFA $G=(Q_G,A,\delta_G,q_0,F_G)$. Using this NFA, the construction of the DPA $P=(Q_P,A,\delta_P,\{q_0\},F_P)$ is computed such that $L(P)=L(G)$.
2. In this procedure, the components of DPA are specified as below:
   a. Q_P is the set of all subsets of all states of Q_G.
   b. F_P is the final set of all subsets, called S of Q_G such that $S\cap F_G=\varnothing.$ which contains the set of states of F_P. i.e., F_P is all sets of G's states that include at least one accepting state of G.
   c. For every set $S\subseteq Q_P$ and each symbol a in A, it is defined as $\delta_P(S,a)=\bigcup_{q\in S}\delta_G(q,a)$. From this, it can be shown as $L(P)=L(G)$.

i.e., for computing $\delta_P\left(S,a\right)$, come across all the states q in S, for which states N goes to from q on input a, and combine all those states as union.

3. The input symbols of these two automata are the same and the starting state of DPA is the set containing only the start set of NFA.
4. After reading sequence of input symbols w, the built DPA is in one state of the set of NFA states as it reads w. Also the accepting states of the DPA are those sets that included in any one of the accepting states of NFA. The NFA also accepted if it in any one of its accepting states, we can conclude that the DPA and NFA accept exactly the same strings, and therefore accept the same language.

For the equivalence, the following theorems based on the subset construction are used:

Theorem 1 [Hopcroft J.E, Rajeev Motwani & Ullman J.D] [7]

If $P=\left(Q_P, A, \delta_P, \{q_0\}, F_P\right)$ is the DPA constructed from NFA $G=\left(Q_G, A, \delta_G, q_0, F_G\right)$ by the subset construction, then $L\left(P\right)=L\left(G\right)$.

Theorem 2 [Hopcroft J.E, Rajeev Motwani & Ullman J.D] [7]

A language L is accepted by some DPA if and only if L is accepted by some NFA.

4. Examples

4.1 Example

For the regular expression $E=\left(aa+b\right)^*\left(bb+a\right)^*$ of the position automaton have constructed in the figure 6. Now the DFA of this position automaton is below. Firstly, all the subsets of the position automaton of E are found and computed. This regular expression contains 6 symbols. Therefore there are 7 states in the position automaton including the starting state {0}. Using this, from the above state diagram of position automaton, a transition table is formed and computed as follows:

Table 1: Transition table

	a	b
q0	{q1, q6}	{q3, q4}
q1	{q1, q2}	$\varnothing$
q2	{q6}	{q3, q4}
q3	{q1, q6}	{q3, q4}
q4	$\varnothing$	{q5}
q5	{q6}	{q4}
q6	{q6}	{q4}

For the Glushkov position automaton, it is given that

$$Q_G = pos\left(E\right)\cup\{q_0\} \\ =\{q_0, q_1, q_2, q_3, q_4, q_5, q_6\}$$

$A=\{a,b\}$ and $F_G = Last\left(E\right)=\{a_2, b_3, b_5, a_6\}$.

For the DFA conversion, all the subsets of all states of Q_G is given as below.

$$Q_G=\left\{\begin{array}{l}\varnothing,\{q_0\},\{q_1\},\{q_2\},\{q_3\},\{q_4\},\{q_5\},\\ \{q_6\},\{q_0,q_1\},\{q_0,q_2\},\{q_0,q_3\},\{q_1,q_2\},\\ \{q_1,q_3\},......,\{q_0,q_1,q_2,q_3,q_4,q_5,q_6\}\end{array}\right\}$$

= 128 different states.

$$F_G=\left\{\begin{array}{l}\{q_2\},\{q_3\},\{q_5\},\{q_6\},\\ ......,\{q_0,q_1,q_2,q_3,q_4,q_5,q_6\}\end{array}\right\}$$

The reachable states are in table 2.

Table 2: Transition table

	a	b
{q0}	{q1}	{q2, q3}
{q1}	{q1}	{q2, q3}
{q1, q4}	{q1, q4}	{q2, q3, q5}
{q2, q3}	{q1, q4}	{q2, q3, q6}
{q2, q3, q5}	{q1, q4}	{q2, q3, q6}
{q2, q3, q6}	{q1, q4}	{q2, q3, q5, q6}
{q2, q3, q5, q6}	{q1, q4}	{q2, q3, q5, q6}

The construction of the DFA is

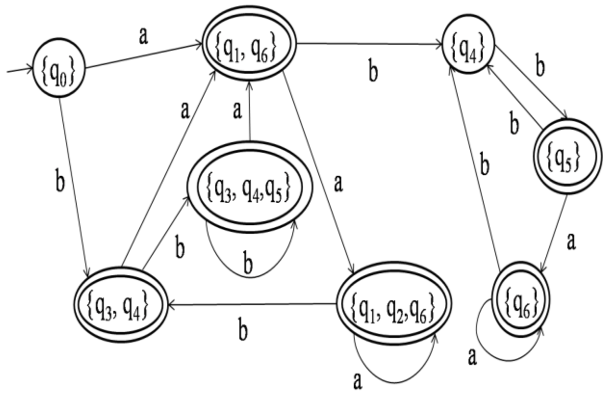
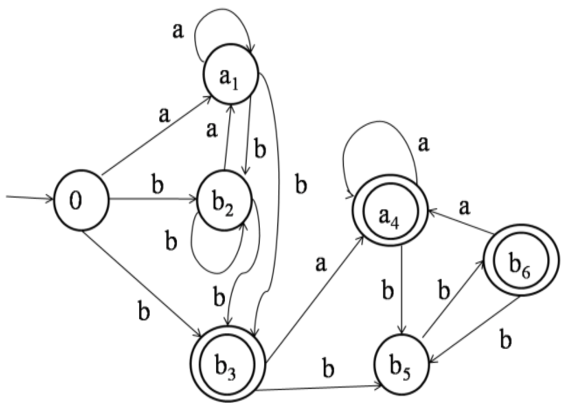


Fig.7 The DFA for the position automaton of $E=\left(aa+b\right)^*\left(bb+a\right)^*$.

4.2 Example

For the regular expression $E=(a+b)^{*}b(a+bb)^{*}$, the construction of DFA containing 7 states including the starting state {0}.



The reachable states are in table 3

Table 3: Transition table

	a	b
$\{q_0\}$	$\{q_1, q_6\}$	$\{q_3, q_4\}$
$\{q_4\}$	$\varnothing$	$\{q_5\}$
$\{q_5\}$	$\{q_6\}$	$\{q_4\}$
$\{q_6\}$	$\{q_6\}$	$\{q_4\}$
$\{q_1, q_6\}$	$\{q_1, q_2, q_6\}$	$\{q_4\}$
$\{q_3, q_4\}$	$\{q_1, q_6\}$	$\{q_3, q_4, q_5\}$
$\{q_1, q_2, q_6\}$	$\{q_1, q_2, q_6\}$	$\{q_3, q_4\}$
$\{q_3, q_4, q_5\}$	$\{q_1, q_6\}$	$\{q_3, q_4, q_5\}$

The construction of the DFA is

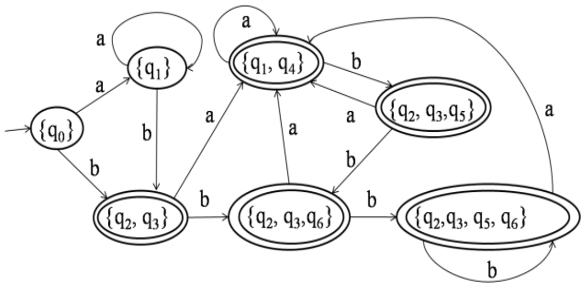


Fig.8. The DFA for the position automaton of $E=(a+b)^{*}b(a+bb)^{*}$.

The position and C-Continuation automata are isomorphic [2, 3]. The relation that exists between the Glushkov position functions First, Follow and Last sets and the C-continuations is enlightened. The construction of C-Continuation automaton may be constructed as a DFA.

5. CONCLUSION

Normally, NFA are useful for representing a pattern matcher that scans a large body of text for one or more keywords. These automata are either simulated directly in software or first converted to a DFA, which is then simulated. Here, the construction of the given various regular expressions and constructed the position automaton of the regular expression is illustrated. Then, the modified algorithm is given to implement existing Glushkov Position automaton (NFA) reduction into DFA equivalences.

Future work involves evaluating the impact of using Glushkov NFA reduction to C-Continuation automaton, even to equation automaton, which is the quotient of the C-Continuation automaton instead of applying state elimination method.

6. REFERENCES

[1] Bruggermann-Klein A., *Regular expressions into finite automata*, Theoretical Comput.Sci., 120:197 – 213, 1993.

[2] Champarnaud, J.M., Quardi F., and Ziadi D., *Normalized expressions and finite Automata*, Intern. Journ. Of Alg. And Comp., 17(1):141-154, 2007.

[3] Champarnaud J.M., and Ziadi D., *Canonical Derivatives, Partial Derivatives and Finite Automaton Constructions*, Theoretical Computer Science, 289:137-163, 2002.

[4] Chang C.H., and Paige R., *From Regular Expressions to DFA's using Compressed NFA's*, Theoretical Computer Science, 178, 1997, 1 – 36.

[5] Glushkov V.M., *The Abstract Theory of Automata*, Russian Mathematical Surveys 16(1961), 1-53.

[6] Hromkovic J., Seibert S., and Wilke T., *Translating Regular Expressions into Small Epsilon-free Nondeterministic Automata*, Journal. Computer. System Sci., 62(4):565-588, 2001.

[7] Hopcroft J.E., Rajeev Motwani and Ullman J.D., *Introduction to Automata theory, Languages and Computation,* Narosa Publishing House, New Delhi, 1987.

[8] Hugo Gouveia, Nelma Moreira and Rogerio Reis, *Small NFAs from Regular Expressions: Some Experimental Results*, arXiv: 1009.3599v1

[9] McNaughton R., and Yamada H., *Regular expressions and state graphs for automata,* IEEE Trans. on Electronic Computers, 9(1):39-47, 1960

[10] Murugesan N., Principles of Automata theory and Computation, 2004, Sahithi Publications.

[11] Murugesan N., and Shanmugasundaram O.V., Construction of State diagram of a regular expression using Derivatives, *www.m-hikari.com/ams/ams-2012/ams.../sundaramAMS21-24-2012.pdf,* Applied Mathematical Sciences, Vol. 6, 2012, no. 24, 1173 – 1179.

[12] Saradhi Varma G.P., and Thirupathi Rao B., *Theory of Computation – Formal Languages and Automata Theory,* Scitech Publications (India) Pvt. Ltd., Chennai, 2005, 2.18 – 2.19.