# Dual Finite Automata to Scan Regular Languages

**Pooja Sharma[1], Deepinder Kaur[2], Rajwinder Kaur[3]**
*[1]A.P, Applied Science (IET Bhaddal, Ropar)*
*[2]Lecturer (Govt. College , Ropar)*
*[3]Lecturer (DWIET , Mohali)*
*pssharmapuja143@gmail.com, rubal_08@yahoo.co.in, rajbanwait88@gmail.com*

**Abstract:** *A regular language is generally accepted by a single finite automaton. But when to increase the efficiency, we use Dual Finite Automata, An input string is scanned by two deterministic finite automata (DFA's): reading from the string's head and tail respectively. One of them accepts the regular language itself; the other accepts the language's reversal. Whether a string is accepted depends on the states of both automata, when their reading heads meet. Dual finite automata can be applied in compiler generation and parallel computing.*

## 1. Introduction

Regular languages (sets) are generally used in compiler design, text editor, and pattern scanning language. A regular language can be accepted by a single finite automaton scanning with single reading head. But in case of Dual Finite Automata, An input string is scanned by two deterministic finite automata (DFA's): reading from the string's head and tail respectively. One of them called ***obverse*** DFA accepts the regular language; the other called ***reverse*** DFA accepts the language's reversal. The reverse DFA is constructed by reversing obverse DFA: the initial state and final states are swapped, with transition function reversed. A reverse DFA's state is represented by a set of obverse DFA's states: a set of possible states which may come back to the start state of obverse DFA. An input string ($W_1W_2$) is processed as follows: $W_1$ is processed left to right by obverse DFA, and $W_2$ right to left by reverse DFA. A string

is accepted when both reading heads meet (i.e. between $W_1$ and $W_2$) and states both are *joinable,* the state of obverse one is included in that of reverse one. The reverse DFA constructed after removing inaccessible states has minimum number of states. It is shown that the space complexity of this approach is $O(|L|+|L^R|)$, linear to the size of the regular language and its reversal. Scanning string by dual finite automata can improve efficiency if both automata are implemented as units of parallel computing. Dual finite automata are used to recognize the sender and receivers of message path during bottom up parsing. The use of Dual Finite Automata will increase the process of pattern matching of Finite Automata with the introduction of two different scanning heads working in opposite direction. This will decrease the time complexity factor while scanning a string to check that the given string belongs to automata or not. This process will take less time in checking a string than the Finite Automata.

## 2. THE PROCESS OF SCANNING BY DUAL DETERMINISTIC FINITE AUTOMATA

### 2.1 Constructing Reverse DFA

Let M and M' respectively be the obverse DFA and reverse DFA for a regular language. Let obverse DFA M =(Q, ", δ, $q_0$ F) with no inaccessible states.
Let reverse DFA *M'* = ($2^Q$, ", δ', $q_0'$ *F'),* where δ'*(q',*a) ={*q*|δ(*q*,a) ε *q',q*ε Q*},* $q_0'$*=F, F'={ q'*| *q'* ⊃ *{ q*$_0$*}, q'*ε$2^Q$}. *q'* here is a state of *M',*and *q is a state of M.Let q' be denoted a* ε *[q'*$_{1.....}$ *q'*$_j$*].q* ε *q' if*
*q* ε{*q'*$_{1.....}$ *q'*$_j$*}.*

***Lemma2.2***

δ*'(q',w)={q*|δ(q, *w*$^R$) ε *q',q* ε *Q}, q'* ε$2^Q$,w ε "*.

*Proof:*

*Basis.* | w|=0, δ*'(q',* ε)={q|δ(q, ε$^R$) ε *q'}= q' hold.*

*Assume*| w|= n hold. δ*'(q',w)={* q | δ(*q, w*$^R$ *)* ε *q'}.*

*Induction. Let α=aw, α* ε", | *α*| = n + 1.

δ*'(q', α)=* δ*'(q'* , *αw)=* δ*'(*δ*'(q'* , *α),w*).

Let *q''=* δ*'(q', α)={q*| δ(q, *α)* ε *q'}.*

δ*'(q'* , *αw)=* δ*'(q'', w)= {q*| δ(q, *w*$^R$*)* ε *q''}*

= *{q*| δ (δ (q, *w*$^R$*), α* )ε *q'}={q*| δ(q, (*αw*)$^R$*)* ε *q'}.*

δ*'(q'* , *α)= {q*| δ(q, *α*$^R$*)* ε *q'}.*

**Lemma 2.3 :** *M' accepts L(M)$^{R.}$*

*Proof:*

$w \varepsilon L(M)$.

$\delta(q_{0,} w) \varepsilon F = q_0'$.

$q_0 \varepsilon \delta'(q_0', w^R) = (q | \delta(q, w) \varepsilon q_0', q \varepsilon Q\}$

$\delta'(q_0', w^R) \varepsilon F'$.

$w^R \varepsilon L(M)$.

## 2.2 Scanning by Obverse and Reverse DFA's

A string here is scanned by the obverse DFA starting from the string's head and by reverse DFA from the tail. The states of both are *joinable* if the states of obverse one is included in that of reverse one. Theorem 2.4 shows that a string is accepted if the state of obverse one and that of reverse one are joinable.

## 2.3 Dual Finite Automata

1) The reverse DFA from its Definition has exponential size.
2) merging states by applying minimization process may lose the information necessary for state matching between two automata. An intuitive improvement is to construct the minimum DFA's for L and $L^R$ first. It then applies its definition to construct the reverse of the smaller one. However, the space complexity is *exp( min* (|L|, $|L^R|$) ), still inefficient. Most of the states in reverse DFA are observed inaccessible. The DFA resulted from removing inaccessible states seems to be a minimum. The space complexity is then *O(*(|L|, $|L^R|$), linear to the size of the regular language and its reversal. *Reverse(M),* or $M^R$, is the DFA resulting after removing inaccessible states from M'. $M^R$ is a minimum DFA for *L(M)$^R$.* M and $(M^R)^R$ are isomorphic. *Reverse* function is invertible up to an isomorphism (a renaming of the states), and its inverse function is itself. It is interesting that a DFA can be minimized by applying *Reverse* function twice: first constructing its reverse DFA and then transforming back. *Reverse* function can be applied any times on a minimum DFA *M.* We then call M and $M^R$ as *dual finite automata* (under *Reverse* operator).

## 3. An Example:

The obverse and Reverse DFA's for regular expression "01*+10*" are given as:

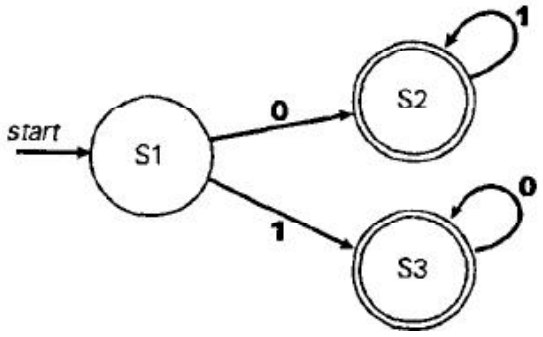


*Figure 1: Observer DFA for 01*+10**

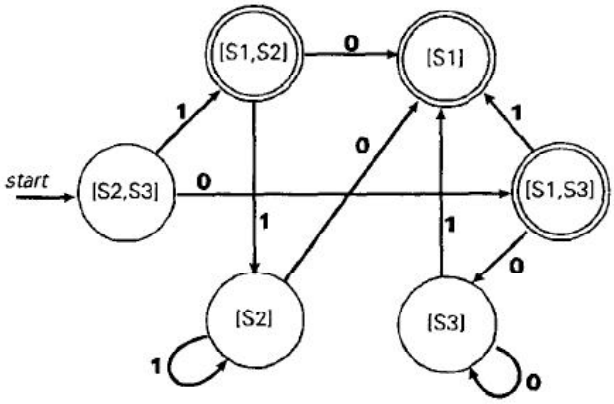


*Figure 2: Reverse DFA for 01*+10**

Table1,2,3: shows the transition functions of M and M'. A start state has an asterisk * on its left; a final state is bold-faced; X means dead state. M" is reverse of M' by renumbering states of M' ( [S2, S3] =1, …, [S2] =6) and constructing reverse of M'. The transition function of M" is δ", equivalent to δ as shown. The reverse DFA constructed from M' is isomorphic to M.

| Figure1: Transition function for M | | |
|---|---|---|
| δ | **0** | **1** |
| *S1 | **S2** | **S3** |
| **S2** | X | S2 |
| **S3** | **S3** | X |

| Table 2: Transition function for M' | | |
|---|---|---|
| δ' | **0** | **1** |
| *[s2,s3] | **[s1,s3]** | **[s1,s2]** |
| [s1,s3] | [s3] | **[s1]** |
| [s1,s2] | **[s1]** | [s2] |
| [s3] | [s3] | **[s1]** |
| [s1] | X | X |
| [s2] | **[s1]** | [s2] |

| Table 3: Transition function for M'' | | |
|---|---|---|
| δ" | 0 | 1 |
| *[235] | [136] | [124] |
| [136] | X | [136] |
| [124] | [124] | X |

## 4. Applications:

Scanning regular languages by dual finite automata can improve the efficiency in scanning strings if each automaton is implemented by a computing unit. A daily use system command directory listing, e.g. "ls dfa??.c" in UNIX, can be executed more faster: each file name in current directory can be scanned with double speed. A scanner of dual reading head has no difficulties in cooperating with a traditional parser. The tokens scanned by the reverse DFA can be stacked for the later usages of parser. It is interesting whether a dual reading head parser can be constructed in

cooperating with a scanner. Regular expressions are used in modeling (specifying) propagations paths of messages in parse trees. Each path is divided into two parts processed by dual finite automata. The sender and receiver(s) of a path is recognized at their youngest common ancestor, when both the states of obverse and reverse automata are joinable. The recognitions of propagation paths are done during bottom-up parsing.

We are also interested in extending the modeling language of propagation paths from regular languages to other category, e.g. context-free languages.

## 5. References

[1] Aho, A. V., Sethi, R., and Ullman, J. D., *Compilers - Principles, Techniques, and Tools,* Addison-Wesley, 1986.

[2] [Con97] *Constructively Formalizing Automata Theory.* R. Constable, P. Jackson, P.Naumov and J. Uribe, 1997. <citeseer.nj.nec.com constable97constructively.html>

[3] Wu, P.-C. and Wang, F.-J., "Message Passings on a Parse Tree," submitted to *IEEE Computer Society 1992 Int'l Conf. on Computer Languages.*

[4] NEVEN, F. AND SCHWENTICK, T. 2002. Query automata on finite trees. *Theoret. Comput. Sci. 275*, 633–674.

*[5].* HROMKOVIC, J. 2000. *Communication Complexity and Parallel Computing*. Texts in Theoretical Computer Science—An EATCS Series. Springer-Verlag, Berlin, Germany.

.