Closure Properties of Prefix-free Regular Languages

Meenu Lochan
Thapar University
Patiala, Punjab, India

Sunita Garhwal
Thapar University
Patiala, Punjab, India

Ajay Kumar
Thapar University
Patiala, Punjab, India

ABSTRACT

Regular languages are closed under union, intersection, complementation, Kleene-closure and reversal operations. Regular languages can be classified into infix-free, prefix-free and suffix-free. In this paper various closure properties of prefix-free regular languages are investigated and result shows that prefix-free regular languages are closed under union and concatenation. Under complementation, reverse, Kleene-closure and intersection operations prefix-free regular languages are not closed.

General Terms

Theoretical Computer Science

Keywords

Regular expressions; State complexity; Prefix-free

1. INTRODUCTION

Regular languages are used in various fields of computer science like compilers, data compression, text processing, software engineering and pattern matching. Regular languages can be represented by finite-state automata (FAs) or regular expressions. Regular languages can be classified as infix free, prefix free, suffix free. A regular language is prefix-free [1] if and only if its minimal DFA *M* has only one final state and the final state has no out-transitions whose target state is not a sink state. In this paper various properties of prefix-free regular languages are investigated.

2. BASIC DEFINITION AND NOTATION

Regular languages can be represented by regular expressions or finite automata. Finite automata can be classified into deterministic finite automata and non-deterministic finite automata. Non-deterministic finite automata are generalizations of DFA.

Def. 2.1: A deterministic finite automaton [10] M is a quintuple (Q, Σ, δ, q_0, F), where Q is the finite set of states, Σ is the finite set of symbols called the alphabet , δ is a transition function mapping $Q \times \Sigma \rightarrow Q$, q_0 is the starting state ($q_0 \in Q$), F is the set of accepting states ($F \subseteq Q$).

Def. 2.2: A non-deterministic finite state automaton (NDFA) [7, 8] is same as DFA except the transition relation is defined as $QX\Sigma \longrightarrow 2^Q$.

A word is accepted by the NFA if some choice of transitions takes the machine to a final state [6]. Some other choices may lead to a non-final state, but the word is accepted as long as there exist at least one accepting computation path in the automaton.

Def. 2.3: A regular expression [10] is a pattern that describes some set of strings.

Example 2.1: Considering input alphabets $A = \{a, b\}$ then $(a+b)^*$ represents a regular expression that includes all strings over a and b.

Def. 2.4: A regular language L is prefix-free [4] if, for all distinct strings $x, y \in \Sigma^*$ and $x, y \in L$ mean that x and y are not prefixes of each other. A regular language is prefix-free if and only if its minimal DFA A has only one final state and the final state has no out transitions whose target state is not a sink state [1].

Def. 2.5: A regular language L is suffix free [4] if, for all distinct strings $x, y \in \Sigma^*$, and $x, y \in L$ mean that x and y are not suffixes of each other. Given two strings x and y over Σ, x is a suffix of y if there exists $z \in \Sigma^*$ such that $zx = y$. A regular language is suffix-free if and only if its minimal DFA have a unique sink state and its starting state does not have any in-transitions [2].

Def. 2.6: A language L is infix free [4] if, for all distinct strings $x, y \in \Sigma^*$, and $x, y \in L$ mean that x and y are not substrings of each other. x is said to be substring or an infix of y if there are two strings u and v such that $uxv = y$. A regular language is infix-free [5] if a DFA A is minimal and is non-returning and non-exiting.

Regular languages are closed under union, concatenation, Kleene closure, intersection, complement and reverse operations.

3. CLOSURE PROPERTIES OF PREFIX-FREE REGULAR LANGUAGES

State complexity [9] is the number of states that are necessary and sufficient in the worst-case for the minimal deterministic finite state automaton that accepts the language obtained from the operation. There is a lot of research work which has been done on state complexity but very few on closure properties of regular languages. So, it is natural to investigate the closure properties of various forms of regular languages.

Theorem 3.1: Prefix-free regular languages are closed under union operation.

Proof: Suppose L_1 and L_2 are prefix-free regular languages represented by DFA M_1 $(Q_1, \sum_1, \delta_1, q_{01}, F_{01})$ and M_2 $(Q_2, \sum_2, \delta_2, q_{02}, F_{02})$.Since both M_1 and M_2 contains a single final state and no out-going transition from their final states. $L_1 \, U \, L_2$ can be found by adding a new starting state q_0 to $q_{01}$and q_{02}. Final states of M_1 and M_2 are combined and there is no transition from the final state. After removing null transitions, we obtain a DFA representing a single final state and no out-going transition from the final state. Hence, the resulting language L is prefix-free regular language. We can say that prefix-free regular languages are closed under union.

Example 3.1: Given L_1= *{ab}* and L_2= *{aabb}* are prefix-free regular languages, then $L=L_1UL_2$ is also a prefix-free regular languages. L_1 and L_2 can be represented by fig. 1 and fig. 2 respectively.q_2 and q_7 represent the final states of the DFA as shown in fig. 1 and fig. 2

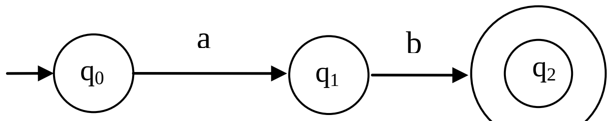

Fig.1: DFA for L_1 = *{ab}*

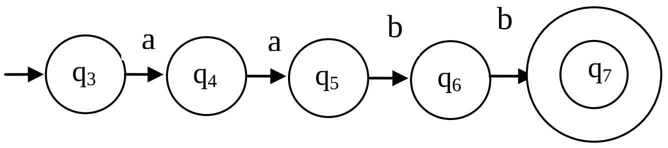

Fig.2: DFA for L_2 = *{aabb}*

Union of L_1 and L_2 after removal of ε-transition and minimization of DFA is shown in fig. 3.state q_2 represents the final state of the fig. 3.

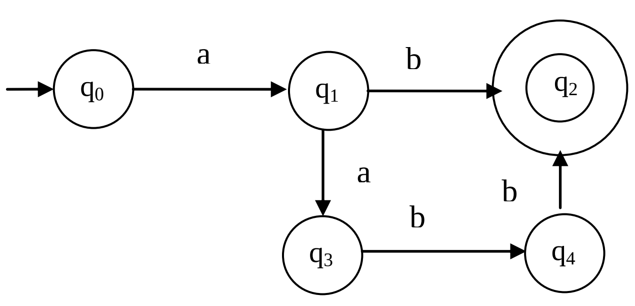

Fig.3: Union after ε-transition

Clearly, $L = L_1 \; U \; L_2$ is a prefix-free regular language as there is no out-going transition from the final state.

Theorem 3.2: Prefix-free regular languages are closed under concatenation operation.
Proof: Suppose L_1 and L_2 are prefix-free regular languages represented by DFA M_1 $(Q_1, \sum_1, \delta_1, q_{01}, F_{01})$ and M_2 $(Q_2, \sum_2, \delta_2, q_{02}, F_{02})$.Since both M_1 and M_2 contains a single final state and no out-going transition from their final states. L_1L_2 are obtained by merging the final state of M_1 and starting state of M_2. It will not cause any problem as final state of M_1 is not having any out-going transition. *DFA obtained for L_1L_2* contains a single final state with no out-going transitions. Hence prefix-free regular languages are closed under concatenation.

Example 3.2: Given L_1 = *{ab}* and L_2 = *{aabb}* are prefix-free regular languages, then
$L=L_1L_2$ is also a prefix-free regular language. L_1 and L_2 are represented by fig.1 and fig.2 respectively.

Concatenation of L_1 and L_2 can be represented by a DFA (with q_6 as final state) as shown in fig.4.

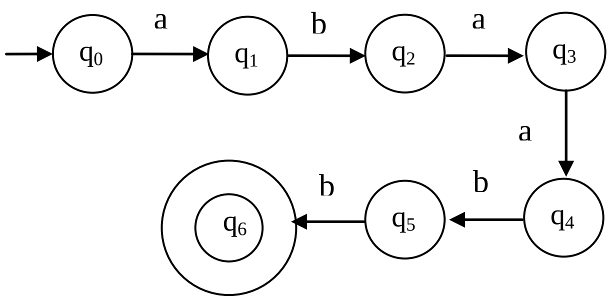

Fig.4: L_1L_2 = *{abaabb}*

Clearly, $L = L_1.L_2$ is a prefix-free regular language as there is no out-going transition from the final state.

Theorem 3.3 Prefix-free regular languages are not closed under complementation operation.
Proof: We prove by an example that prefix-free regular languages are not closed under complementation. Let L= ab^*a is a prefix-free regular languages.

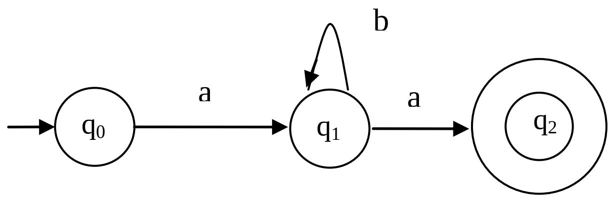

Fig.5: L = {ab^*a}

For performing complement operation, all the strings which are accepting in original language become non-accepting and all non-accepting strings become accepting strings.
For complement, following steps are carried out:

1. Add an additional state called dead state (q_d).
2. Repeat for each state (q) of the DFA except q_d .
   - 2.1 Find the alphabet which is not having out-transition from q.
   - 2.2 Add an edge from q to q_d with input alphabets found from step 2.1.
3. Convert accepting state into non-accepting state and non- accepting state into accepting state.
4. Add a self loop on q_d with labeled all input alphabets.

Above steps are carried out on the language L = {ab^*a}.After applying above steps, we obtain the DFA as shown in fig. 6 and fig. 7.

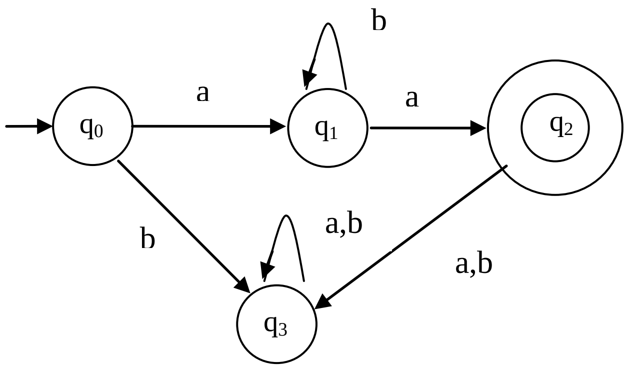

Fig.6: *L* with a dead state and q_2 as final state

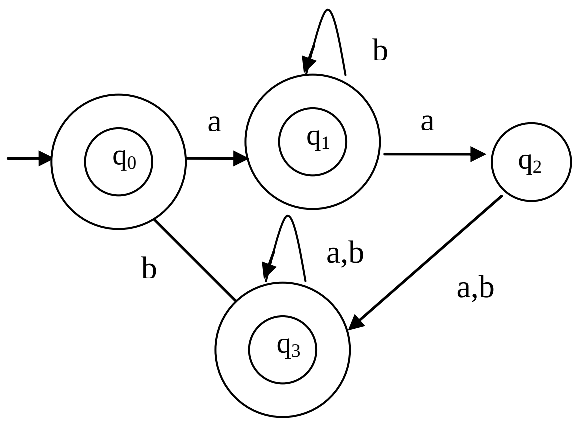

Fig.7: L^c with a dead state

Clearly, DFA representing L^c is not a prefix-free regular language.

Theorem 3.4: Prefix-free regular languages are not closed under intersection operation.
Proof: We proof it by contradiction. Let us assume intersections of L_1 and L_2 prefix-free regular language are closed. Then, $L_1 \cap L_2 = (L_1^C + L_2^C)^C$ where L_1^C represent complement of L_1. In Theorem 3.3, we have proved that a prefix-free regular language is not closed under complement. It implies that $L_1 \cap L_2$ may or may not be a prefix-free regular language. Hence prefix-free regular languages are not closed under intersection operation.

Theorem 3.5: Prefix-free regular languages are not closed under Kleene closure operation.
Proof: Let L be a prefix-free regular language represented by a DFA $M\ (Q, \sum, \delta, q_0, F)$. We prove by an example that prefix-free regular languages are not closed under Kleene closure operation.
Consider $L_1 = \{ab\}$ represented by DFA M as shown in fig.1.
Kleene closure of M can be obtained as shown in fig. 8.
ε-closure (q_3) = {q_3, q_0, q_4} = A
On reading a on state q_0 from A, we reach to q_1. ε-closure (q_1) = {q_1} = B
Reading of b is not possible on state A, as there is no out transition from q_0, q_3 and q_4 labeled b.
Reading of a on state B is not possible as there is no out transition from q_1 labeled a.
In reading b on state B, we reach to q_2. ε- Closure (q_2) = {q_0, q_2, q_4} = C
Similarly, on reading a on state C we reach to state B.

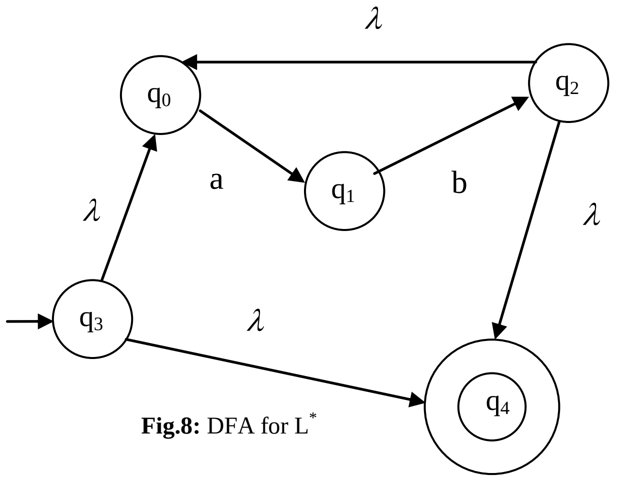

Fig.8: DFA for L^*

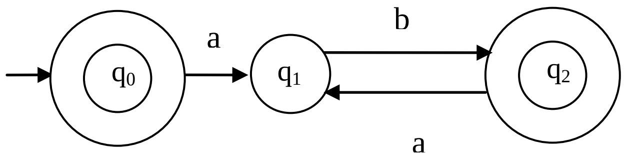

Fig.9: DFA without ε- Closure

Clearly, under Kleene-closure prefix-free regular languages are not closed.

Theorem 3.6: Prefix-free regular languages are not closed under reversal operation.
Proof: Let L be a prefix-free regular language represented by a DFA $M\ (Q, \sum, \delta, q_0, F)$.We proof it by giving an example in which reversal of a prefix-free regular language is not a prefix-free regular language. Let *Language L represented by fig.9.*

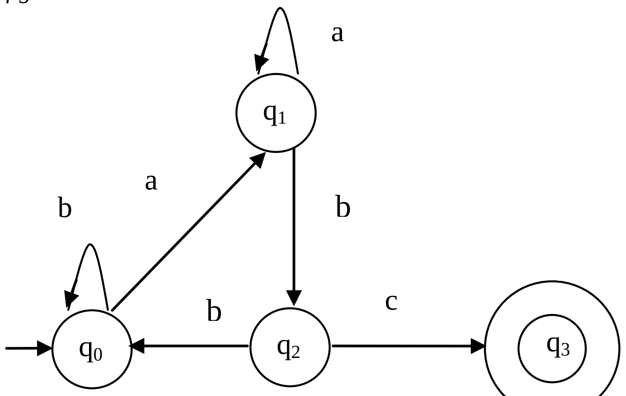

Fig.9: DFA for language L
Reversal of M can be obtained as shown in fig. 10.

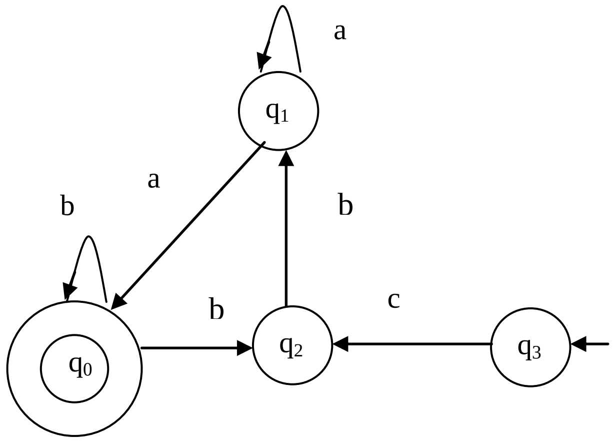

Fig. 10: Reversal of $(L_1)^R$
Hence under reversal operation, prefix-free regular languages are not closed.

4. Conclusion and future directions

Prefix-free regular languages are closed under union and concatenation operation. Prefix-free regular languages are not closed under complementation, intersection, Kleene closure and reversal operations. In future work can be carried out on closure properties of suffix-free and infix-free regular languages.

5. REFERENCES

[1] Han Y.S., Salomaa K., Wood D., 2007, " State complexity of Prefix-free Regular Languages", HKUST Theoretical Computer Science Center Research Report HKUST-TCSC-2006-02.

[2] Han Y.S., Salomaa K, 2009, "State complexity of basic operations on suffix-free regular languages", Theoretical Computer Science 410, 2537-2548.

[3] Han Y.S., Wang Y., Wood D., 2005, "Prefix-Free Regular-expression Matching" Springer-Verlag Berlin Heidelberg, LNCS 3537, 298-309.

[4] Han Y.S., Wood D., 2007, "Outfix-free Regular Languages and Prime Outfix-free decomposition", Fundamenta Informaticae XX, 1-17.

[5] Han Y.S. ,Wood D., " Overlap-free Regular Languages", Springer-Verlag, LNCS 4112, Berlin Heidelberg, COCOON 2006, 469-478.

[6] Kari J., 2011, "Automata and formal languages", University of Turku.

[7] Mishra K.L.P. and N. Chandrasekaran, 1998, "Theory of Computer Science (Automata Language and. Computation) ", PHI, Second edition.

[8] Peter Linz, 2009, "An Introduction to Formal Languages and Automata", Narosa publishers, fourth edition.

[9] Sheng Yu, 2001, "State Complexity of Regular Languages", Journal of automata, Languages and Combinatorics, 6(2).

[10] Ullman, J., J. E. Hopcroft and R. Motwani, 2001, "Introduction to Automata Theory, Languages, and Computation", Pearson Education Inc.