AUTOMATA FORMAL LANGUAGES AND LOGIC

Lecture notes on Properties of Regular Language

Prepared by:
Prof.Divya S J
Assistant Professor

Department of Computer Science & Engineering

PES UNIVERSITY
(Established under Karnataka Act No.16 of 2013)
100-ft Ring Road, BSK III Stage, Bangalore – 560 085

Table of contents

Properties of Regular Languages

Regular language is formal language for which we can able to construct:

- a DFA, or
- a NFA, or
- a RegEx, or
- a right-linear grammar or a left-linear grammar.

These constructions help us to accept ,match, parse or generate all the strings in the language.

Closure properties on regular languages are defined as certain operations on regular languages which are guaranteed to produce another regular language as output.

Closure refers to some operation on a language, resulting in a new language that is of the same "type" as originally operated on, here we are talking about regular languages being closed under an operation.

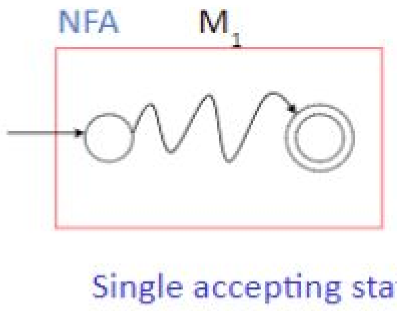
We will prove that regular languages are closed under

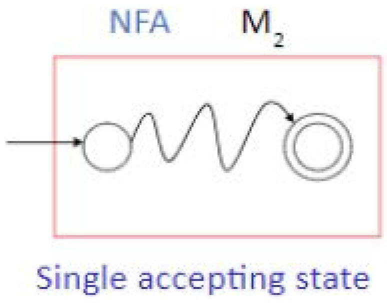
Union , Concatenation , Star closure, Complementation, Intersection and Reversal operation.

For each of these operations, we just need to prove that there is some way of either constructing a Finite Automata or Regular Expression or Regular Grammar, hence making the resultant language regular.

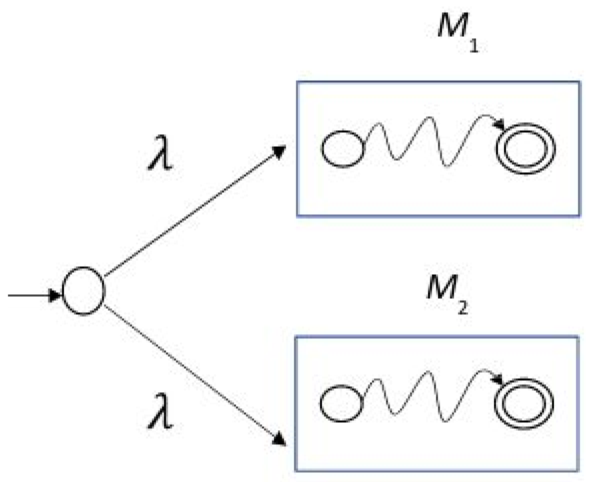
Union of Regular Languages

Consider two regular languages L_1 and L_2 which have machine M1 and M_2 respectively and each machine has a single accepting state as shown below.

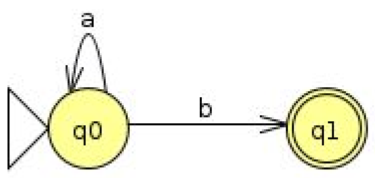


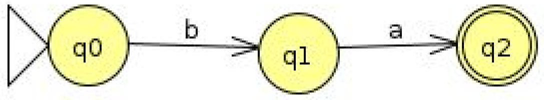


NFA for the union of two regular languages

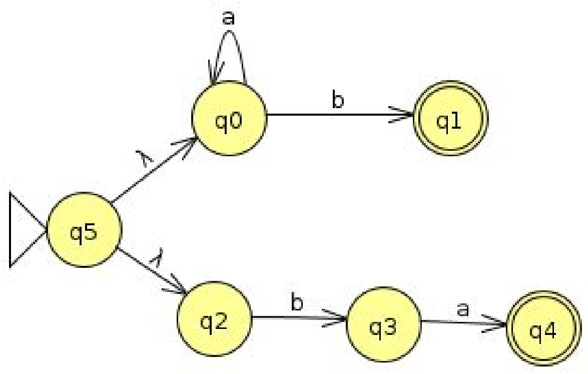


For example, Consider the language $L_1 = \{a^n b\} \mid n \geq 1\}$ and $L_2 = \{ba\}$ and its corresponding finite automata





Now we will combine both the automata by introducing a new start state with lambda transition on M_1 and M_2 and resulting is also NFA.



The resulting NFA accepts either n a's followed by b or b followed by a.

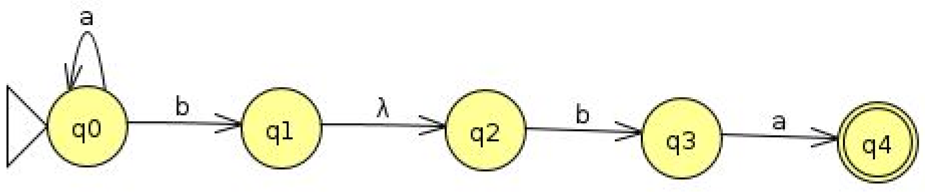
Even we can consider an regular expression where L_1 represents strings starts with **a** where ={a,b}.Regular expression R_1 is **a(a+b)*** L_2 represents strings ends with **b** where ={a,b}.Reg Ex R_2 is **(a+b)*** **b.**The union of two languages is also regular since it is represented by the RegEx is **a(a+b)*** **+ (a+b)*** **b** is also a regular language.

Concatenation of Regular Languages

If L_1 and L_2 are regular languages,then $L_1.L_2$ is also regular.

The two finite automata can be combined into an NFA by introducing a lambda transition from the final state of machine 1 to the start state of machine 2. We must change the M1 machine final state to non-final as we must continue searching for strings that belong to L2. .Such an NFA would accept any string that is a concatenation of a string from the first language and a string from the second language , thereby making the concatenation of the two languages also regular.

For example, Consider the language $L_1 = \{a^n b\} \mid n \geq 1\}$ and $L_2 = \{ba\}$ .The concatenation of two regular languages are $L_1\ L_2 = \{a^n b\}\{ba\} = \{a^n bba\}$ and the resulting automaton is

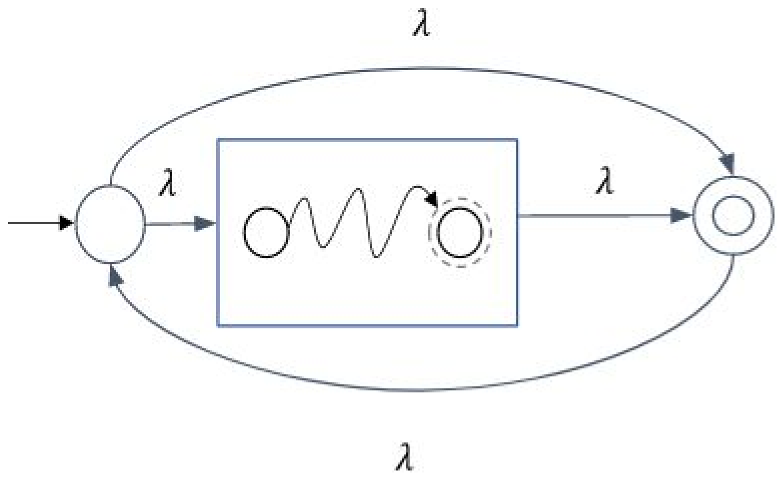


Star closure

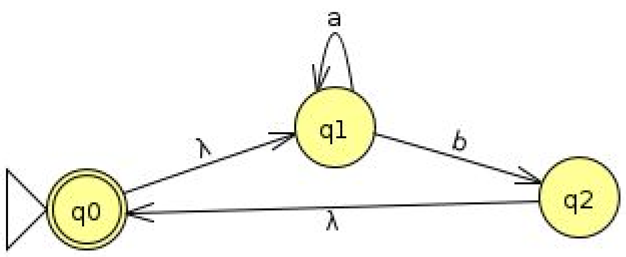
If L is regular languages,so L^* is also regular.

The star closure can be obtained by creating loops with lambda transitions. We introduce a new start and final state on lambda transition to the existing Finite automata. Then creating a loop ensuring that the sub-automata can be repeated any no. of times.

Such an NFA would accept any string that is obtained by repeatedly concatenating strings from the given language,thereby making the closure of the regular language also regular.



The result of the * closure for the language L = $\{a^nb \mid n \geq 1\}$ is also regular i.e., L* = $\{a^nb \mid n \geq 1\}^*$,the resulting automaton is



Reversal of regular language

If L is regular its reversal L^R is also a regular language.

Assume the existence of a finite automaton M with a single final state that accepts L

Given the transition graph for M, to construct a FA M^R that accepts L^R:

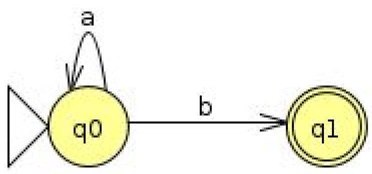
•The start state in M becomes the final state in M^R

•The final state in M becomes the start state in M^R

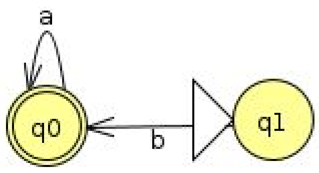
•The direction of all transition edges in M is reversed

If there is more than one final state in the machine M , a new start can be introduced with lambda transitions leading from it to these multiple final states.

The language L =$\{a^nb \mid n \geq 1\}$



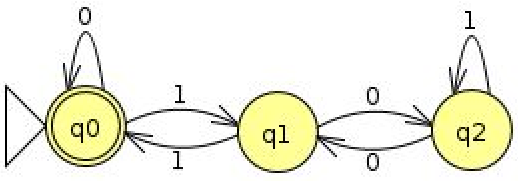
and its reversal i.e., L^R is obtained by applying the above procedure



Given language L,L^R is the set of whose reversal is in L.
Example: L{0,01,100}
L^R = {0,10,001}
Let E = $01^*+ 010^*$
$E^R = 0^*10 + 1^*0$

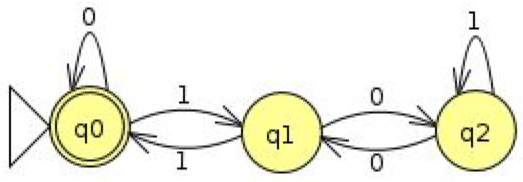
Let's look at an interesting example for reversal

Here we have a finite automata which accepts the binary Strings whose decimal equivalent is divisible by 3



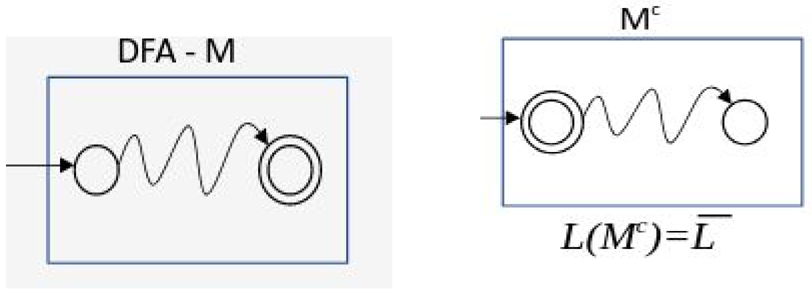
If we reverse the automata by making final to non final and vice versa and also reversing the direction of arrows that gives the same automata.

This happens because strings that belong to the language are all palindromes



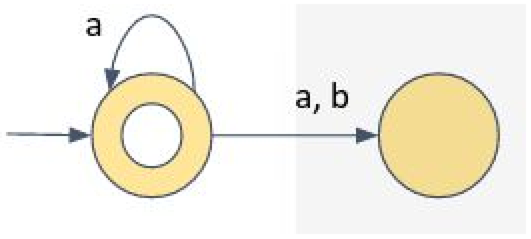
Complement of Regular Languages

The complement of a language is the set of all strings that do not belong to the language.Consider an automaton for the given language.Its final states accept all the strings in the language. Whenever an input string does not belong to the language, the automaton halts in a non final state which might be a reject state or any other non final state.

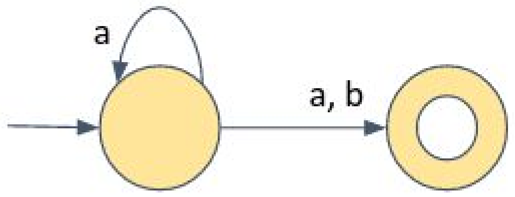


Hence if we toggle the final and the non-final states we will get an automata which accepts the complement of the original language.
We must note that NFAs cannot be used for complement.

Because if we apply the same procedure to nfa, the resulting nfa may or may not result in the complement of the language.
Consider an example here this automata accepts the language a*

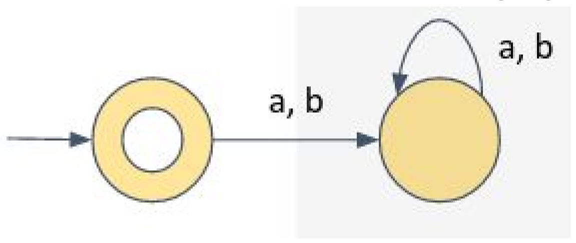


Taking the complement of this NFA by toggling the final and the non-final states we get an automata which accepts the language a* ending with either a or b.



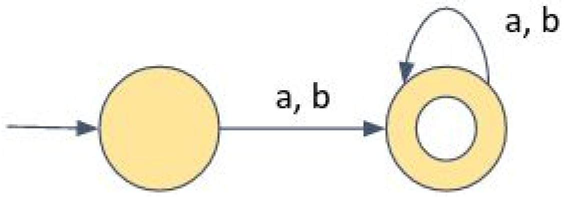
Now, we see that we have string a or even more strings in common. Hence complementing this NFA does not complement the language.

Consider another automata for the language which accepts only lambda.



L = { λ }

Now if we complement the NFA we get an automata which accepts the language containing strings of length > 1 framed over the alphabet a and b.



L = {a+b}*

In this case complement of the automata resulted in complement of the language.

Hence complement of NFA is not guaranteed to complement the language hence we must always work with a DFA whenever we want to complement a regular language.

Closure under Intersection

If L_1 and L_2 are regular languages,so $L_1 \cap L_2$ is also regular.

Slide 20

There are two ways of showing the intersection of two regular languages is also regular.

One method is by taking a cross product of two machines i.e., M_1 and M_2.

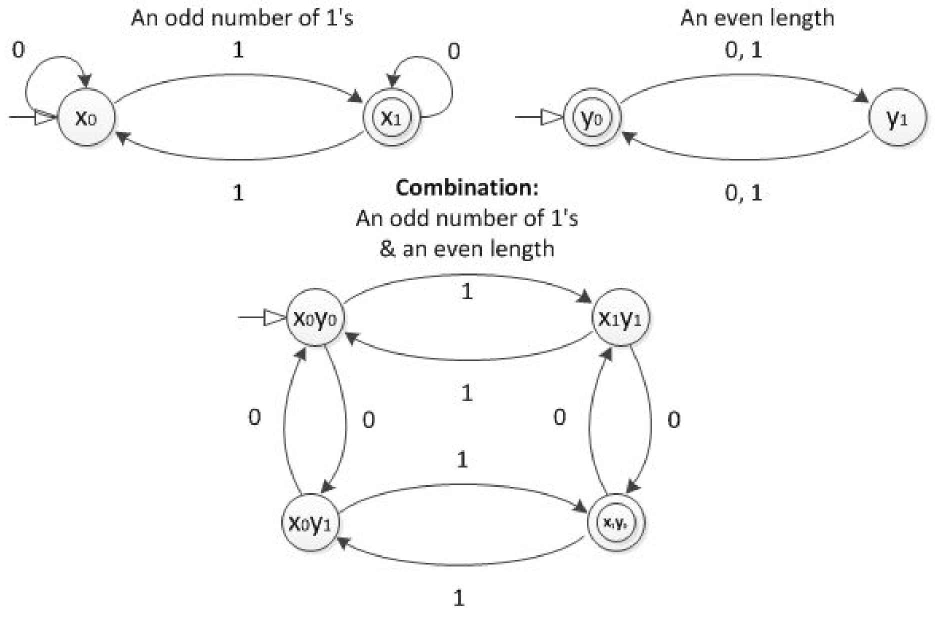
Each state of the new machine M is represented as a pair i.e.,q_ip_j where q_i is the one of the state of M_1 and p_j is the one of the state of M_2.

The initial state of new automata is the initial state of machine M_1 and initial state of machine M_2

The final state of new automata is the pair of final states of machine M_1 amd machine M_2

Consider the example where M1 accepts strings from the language where we have odd number of a's and M2 accepts the language where length of the string is even.

When we take cross product we get 4 states. Start state is x_0y_0 and final state is x_1y_0 This ensures that only we accept strings that belong to both the sets. We find the transition of each pair representing a new state on each input symbol and we get the resultant automata that accepts the intersection of two languages.



Another method is by applying DeMorgan's law

$L_1 \cap L_2 = \overline{\overline{L1} \cup \overline{L2}}$

$\Rightarrow$ L1 , L2 regular ,regular

$\Rightarrow \overline{L1}\ ,\ \overline{L2}$ regular ,regular

$\Rightarrow \overline{L1} \cup \overline{L2}$ regular

$\Rightarrow \overline{\overline{L1}\ U \overline{L2}}$ regular

$\Rightarrow L_1 \cap L_2$ regular

Therefore regular languages are closed under intersection too.

Decidable Properties of Regular Language

We say a property is decidable iff we can write a program which when given an input will always give the right answer. Yes for a yes, no for a no**.**

Let's look at the first question:

1. Does a given string w belong to the regular language L?
   We say Yes, if an automaton for the language accepts the string .No otherwise .Similarly if regular grammar for the language derives the string or a RegEx for the language matches the string ,then it belongs to the language ,else it does not.
2. Can we answer whether a given regular language is empty?
   The language is empty if there is no path from the start state of its automaton to any of the final states.
3. Can we answer whether a given regular language accepts everything?
   As we know regular languages are closed under complement. We can take a complement of the given language. If the complement accepts nothing that means the original language accepts everything.
4. Can we answer If a given regular language is finite or infinite?11
   We say A regular language is finite if its automaton is acyclic.It is infinite if there is a cycle or a loop among the states in its automaton that can be traversed in going from the start state to one of the final states of the automaton.
5. Given two regular languages L_1 and L_2 how can we check if L_1 is equal to L_2?
   We can construct a minimal DFA for both the languages. We know that for a given language there exists a unique DFA known as the minimal DFA that accepts the language. Hence if both the languages are equal, they will result in the same minimal DFA.