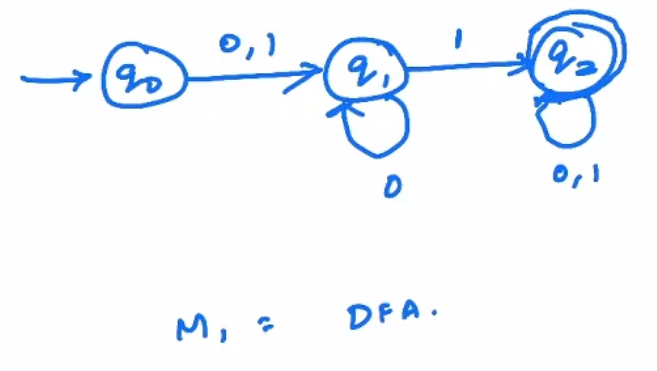
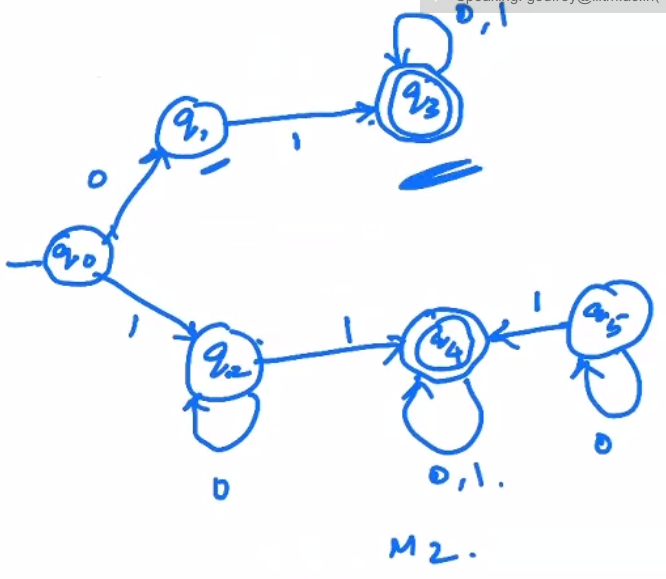
Minimization of a DFA

Consider the following automaton



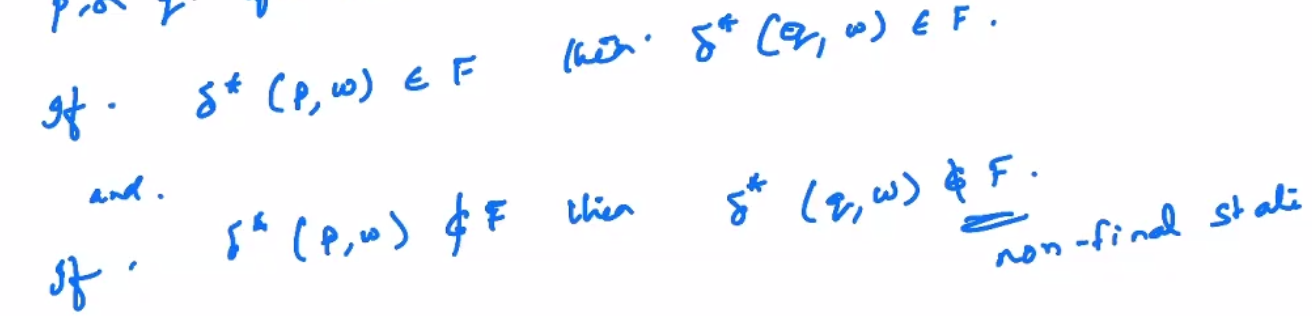
The language, L = {01,11,01^n}

The automaton below also has the same language as the automaton above.



State q5 is unreachable, so we can remove it. Then note that the branch above has the exact same configuration as the one below. So we can merge these 2 branches into 1. Then this automaton becomes equivalent to automaton M1 above.

2 states p and q of a DFA are said to be indistinguishable or equivalent if



Types of equivalence

1. Zero equivalence

When the length of the input string is 0, two states A and B are indistinguishable, then the states A and B are zero equivalent.

1. N-equivalence

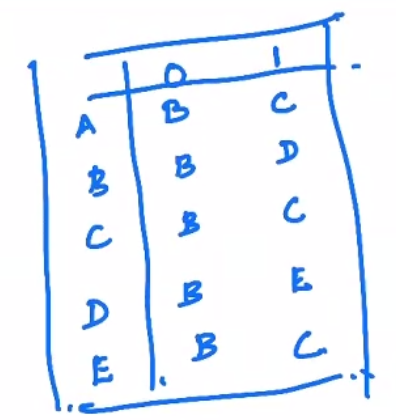
When the length of the input string is n, two states A and B are indistinguishable, then the states A and B are n-equivalent.

Steps to minimize a given DFA

1. Obtain transition table
2. Compute equivalences of states
3. Construct a reduced DFA by combining the equivalent states

Eg

Transition table



NF = {A, B, C, D} and F = {E}

Checking for 1-equivalence

A on 0 goes to B, so does B. Both of these are final. Similarly on 1, both go to non final states. Hence A and B are equivalent.

Checking for A and C. They are equivalent again. Now since B and A are equivalent, we can say A,B,C are equivalent.

Checking for D and E. They are not equivalent.

So we have {A,B,C} , {D}, {E}

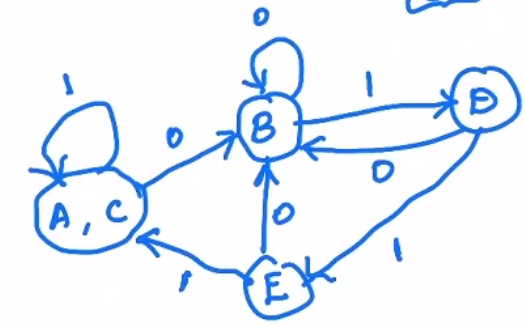
Checking for 2-equivalence

A and B are not equivalent now, consider the input 11. However, A and C are 2-equivalent.

Checking for 3-equivalence

A and C are 3-equivalent. For all further values of n, A and C will always be n-equivalent.

Now we construct the reduced DFA, with only 4 states.



Regular Expressions

They are a way of representing a regular language. A regular language can be represented using a finite state automaton, or a regular grammar.

If sigma is the alphabet, then phi (empty set), lambda and any a belonging to sigma are primitive regular expressions.

If r1 and r2 are regular expressions then r1 + r2, r1.r2 (concat), r1\* (repeating), (r1) (precedence) are all regular expressions.

A string is a regular expression iff it can be derived from the primitive regular expressions in a finite number of operations

Eg.

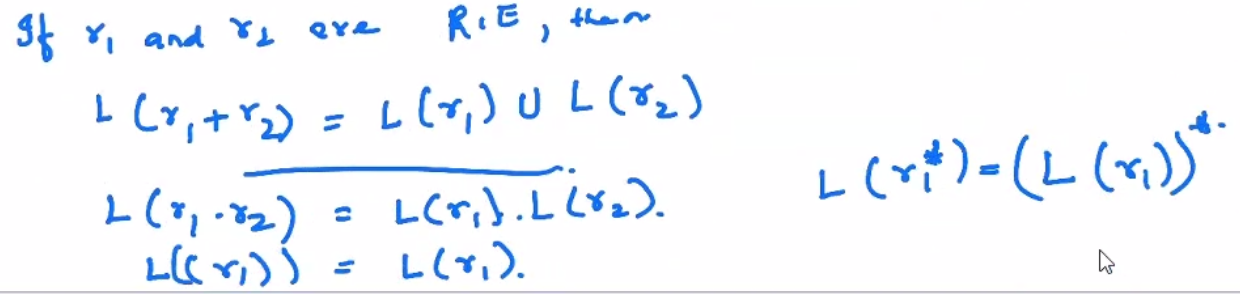
Sigma = {a,b,c}

Then (a\*)(c + phi), (a + b.c)(c + phi) are regular expressions.

Language defined by a regular expression

Rules

1. Phi is a RE representing empty set
2. Lambda is an RE representing the set {lambda}
3. If a belongs to Sigma, a is an RE representing set {a}



Eg

Consider an RE

a\*.(a+b)

