النظرية الاحتسابية

(Computational Theory)

المرحلة الثانية

م. كنار محمد سامي مصطفى أ.م. نغم ثروت

Lecture 1: Languages & Grammars

1. Introduction Languages

* Definition of formal languages
* Importance in computer science (e.g., compilers, parsers)

2. Alphabets and Strings

* Alphabet (Σ): a finite set of symbols
* String: a finite sequence of symbols from Σ
* Empty string (ε), length of a string

3. Languages

* A language is a set of strings over an alphabet
* Examples of languages (e.g., set of all binary strings with even number of 0s)

4. Grammars

* Grammar: a set of production rules for generating strings
* Components of a grammar:
  * Terminals (T)
  * Non-terminals (N)
  * Start symbol (S)
  * Production rules (P)

Chomsky Hierarchy (Preview)

* Type 0: Unrestricted grammars
* Type 1: Context-sensitive

* Type 2: Context-free
* Type 3: Regular

Derivations and Parse Trees

* Leftmost, rightmost derivations
* Parse tree structure

Lecture 2: Ambiguity

* A grammar is ambiguous if there exists a string with more than one parse tree

Applications of CFGs

* Used in programming languages, compiler design
* Example: arithmetic expressions

5. Examples

* CFG for balanced parentheses
* CFG for simple expressions:
E → E + E | E * E | (E) | id

Regular Grammar

1. Introduction

* Simpler than CFGs
* Equivalent to finite automata

Types of Regular Grammar

* Right-linear grammar
* Left-linear grammar

Production Rules

* Form: $A \rightarrow aB$ or $A \rightarrow a$ or $A \rightarrow \varepsilon$

Examples

* Grammar for binary strings ending with 1
* Grammar for even-length strings

Lecture 3: Finite Automata

Deterministic Finite Automaton (DFA)

* Definition: 5-tuple $(Q, \Sigma, \delta, q_0, F)$
* Transition function $\delta: Q \times \Sigma \rightarrow Q$
* Example with transition diagram

Non-Deterministic Finite Automaton (NFA)

* Can have multiple transitions for a symbol or ε-transitions
* Equivalence with DFA

Accepting Strings

* A string is accepted if it leads to a final state
* Language of an automaton: all accepted strings

Construction Examples

To optimize the DFA you have to follow the various steps. These are as follows:

Step 1: Remove all the states that are unreachable from the initial state via any set of the transition of DFA.

Step 2: Draw the transition table for all pair of states.

Step 3: Now split the transition table into two tables T1 and T2. T1 contains all final states and T2 contains non-final states.

Step 4: Find the similar rows from T1 such that:

1. δ (q, a) = p
2. δ (r, a) = p

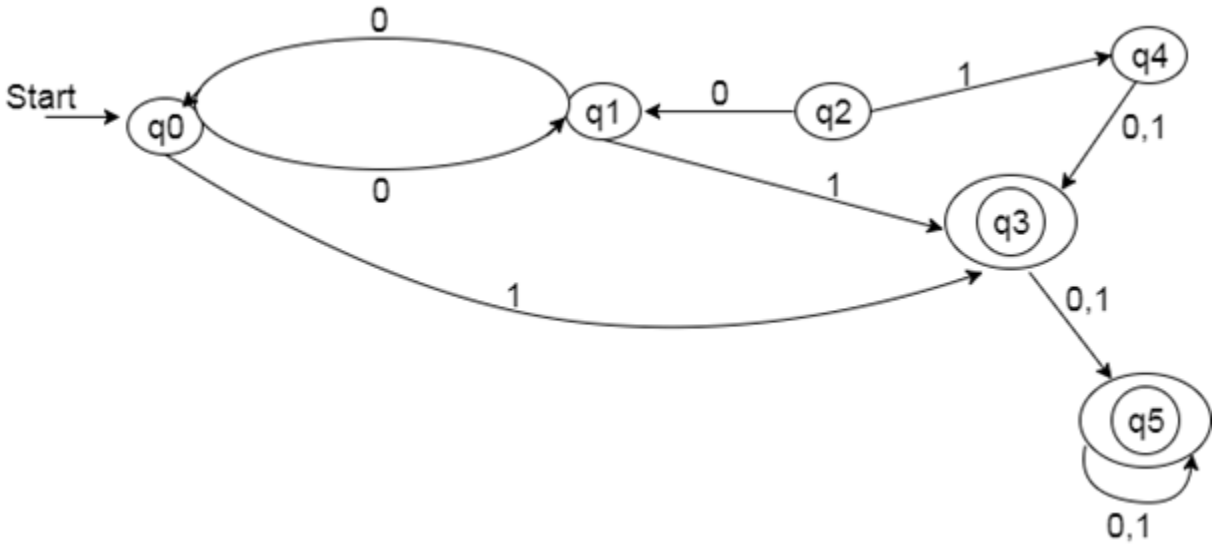
That means, find the two states which have same value of a and b and remove one of them.

Step 5: Repeat step 3 until there is no similar rows are available in the transition table T1.

Step 6: Repeat step 3 and step 4 for table T2 also.

Step 7: Now combine the reduced T1 and T2 tables. The combined transition table is the transition table of minimized DFA.

Example



Solution:

Step 1: In the given DFA, q2 and q4 are the unreachable states so remove them.

Step 2: Draw the transition table for rest of the states.

State	0	1
→q0	q1	q3
q1	q0	q3
*q3	q5	q5
*q5	q5	q5

Step 3:

Now divide rows of transition table into two sets as:

1. One set contains those rows, which start from non-final sates:

State	0	1
q0	q1	q3
q1	q0	q3

2. Other set contains those rows, which starts from final states.

State	0	1
q3	q5	q5
q5	q5	q5

Step 4: Set 1 has no similar rows so set 1 will be the same.

Step 5: In set 2, row 1 and row 2 are similar since q3 and q5 transit to same state on 0 and 1. So skip q5 and then replace q5 by q3 in the rest.

State	0	1
q3	q3	q3

Step 6: Now combine set 1 and set 2 as:

State	0	1
→q0 q1 *q3	q1 q0 q3	q3 q3 q3

Now it is the transition table of minimized DFA.

Transition diagram of minimized DFA:

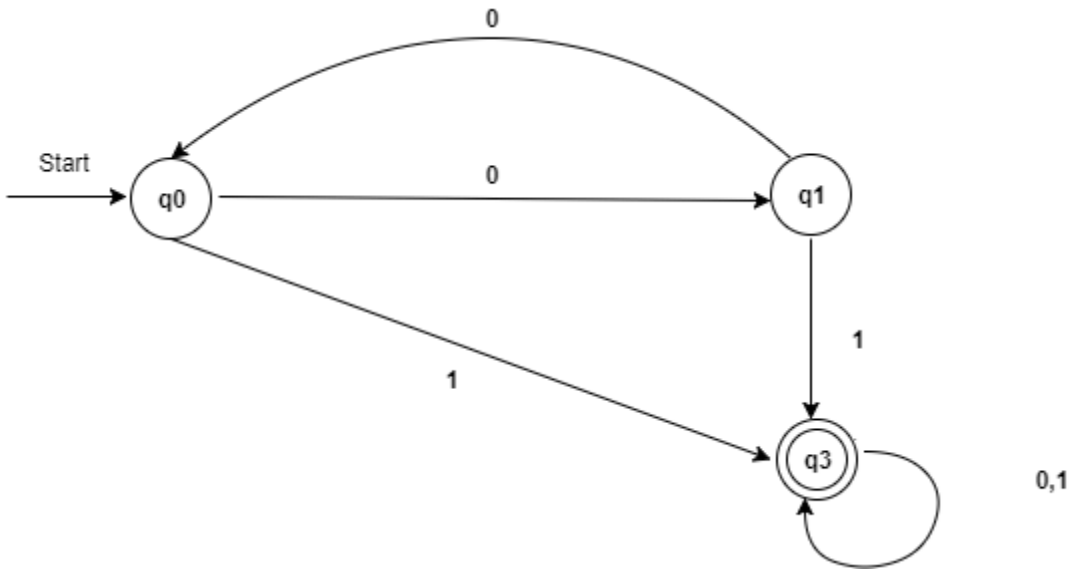


Fig: Minimized DFA

Lecture 4 :Conversion from NFA to DFA

In this section, we will discuss the method of converting NFA to its equivalent DFA. In NFA, when a specific input is given to the current state, the machine goes to multiple states. It can have zero, one or more than one move on a given input symbol. On the other hand, in DFA, when a

specific input is given to the current state, the machine goes to only one state. DFA has only one move on a given input symbol.

Let, M = (Q, ∑, δ, q0, F) is an NFA which accepts the language L(M). There should be equivalent DFA denoted by M' = (Q', ∑', q0', δ', F') such that L(M) = L(M').

Steps for converting NFA to DFA:

Step 1: Initially Q' = ϕ

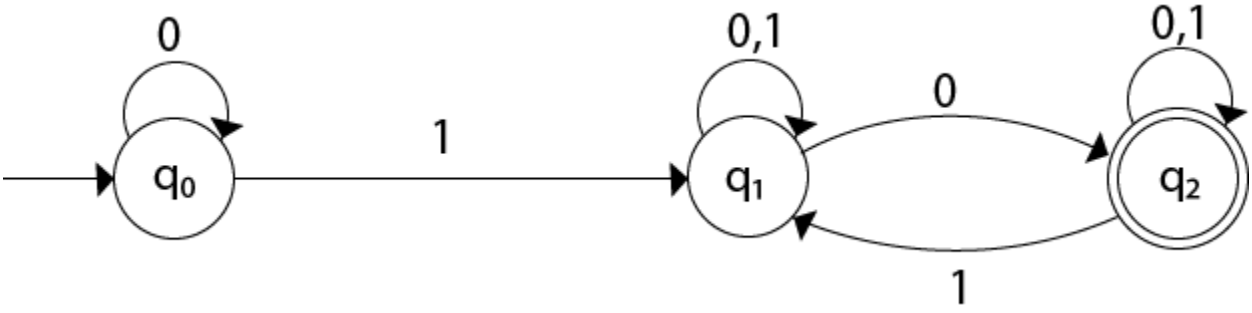
Step 2: Add q0 of NFA to Q'. Then find the transitions from this start state.

Step 3: In Q', find the possible set of states for each input symbol. If this set of states is not in Q', then add it to Q'.

Step 4: In DFA, the final state will be all the states which contain F(final states of NFA)

Example 1:

Convert the given NFA to DFA.



Solution: For the given transition diagram we will first construct the transition table.

State	0	1
→q0	q0	q1

q1	{q1, q2}	q1
*q2	q2	{q1, q2}

Now we will obtain δ' transition for state q0.

1. δ'([q0], 0) = [q0]
2. δ'([q0], 1) = [q1]

The δ' transition for state q1 is obtained as:

1. δ'([q1], 0) = [q1, q2] (**new** state generated)
2. δ'([q1], 1) = [q1]

The δ' transition for state q2 is obtained as:

1. δ'([q2], 0) = [q2]
2. δ'([q2], 1) = [q1, q2]

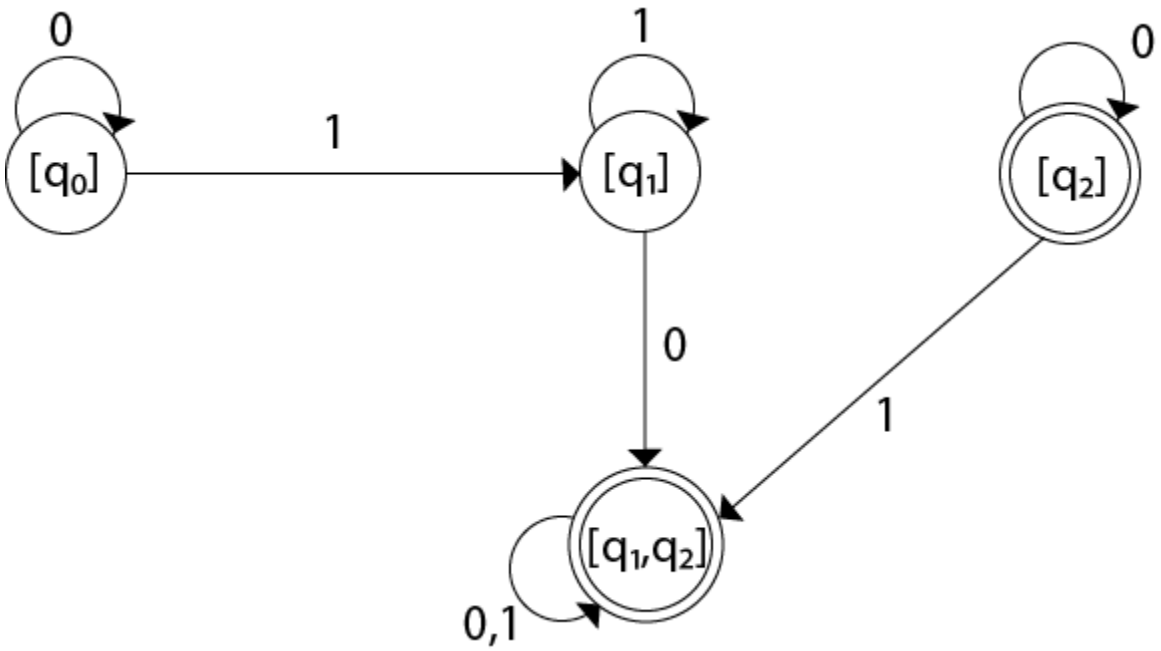
Now we will obtain δ' transition on [q1, q2].

1. δ'([q1, q2], 0) = δ(q1, 0) ∪ δ(q2, 0)
2. = {q1, q2} ∪ {q2}
3. = [q1, q2]
4. δ'([q1, q2], 1) = δ(q1, 1) ∪ δ(q2, 1)
5. = {q1} ∪ {q1, q2}
6. = {q1, q2}
7. = [q1, q2]

The state [q1, q2] is the final state as well because it contains a final state q2. The transition table for the constructed DFA will be:

State	0	1
→[q0]	[q0]	[q1]
[q1]	[q1, q2]	[q1]
*[q2]	[q2]	[q1, q2]
*[q1, q2]	[q1, q2]	[q1, q2]

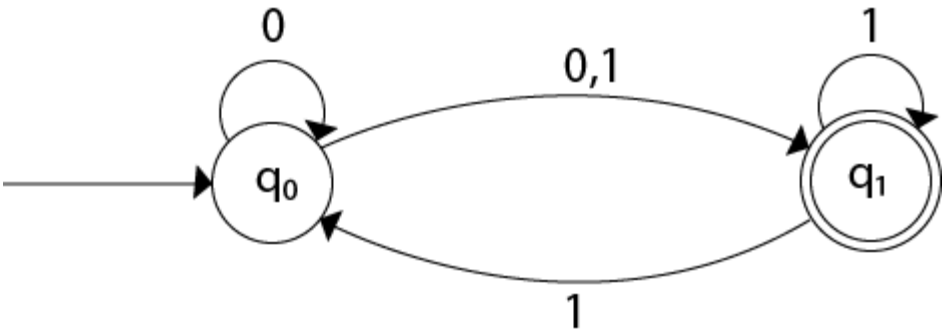
The Transition diagram will be:



The state q2 can be eliminated because q2 is an unreachable state.

Example 2:

Convert the given NFA to DFA.



Solution: For the given transition diagram we will first construct the transition table.

State	0	1
→q0	{q0, q1}	{q1}
*q1	ϕ	{q0, q1}

Now we will obtain δ' transition for state q0.

1. δ'([q0], 0) = {q0, q1}
2. = [q0, q1] (**new** state generated)
3. δ'([q0], 1) = {q1} = [q1]

The δ' transition for state q1 is obtained as:

1. δ'([q1], 0) = ϕ
2. δ'([q1], 1) = [q0, q1]

Now we will obtain δ' transition on [q0, q1].

1. δ'([q0, q1], 0) = δ(q0, 0) ∪ δ(q1, 0)
2. = {q0, q1} ∪ ϕ
3. = {q0, q1}

4. = [q0, q1]

Similarly,

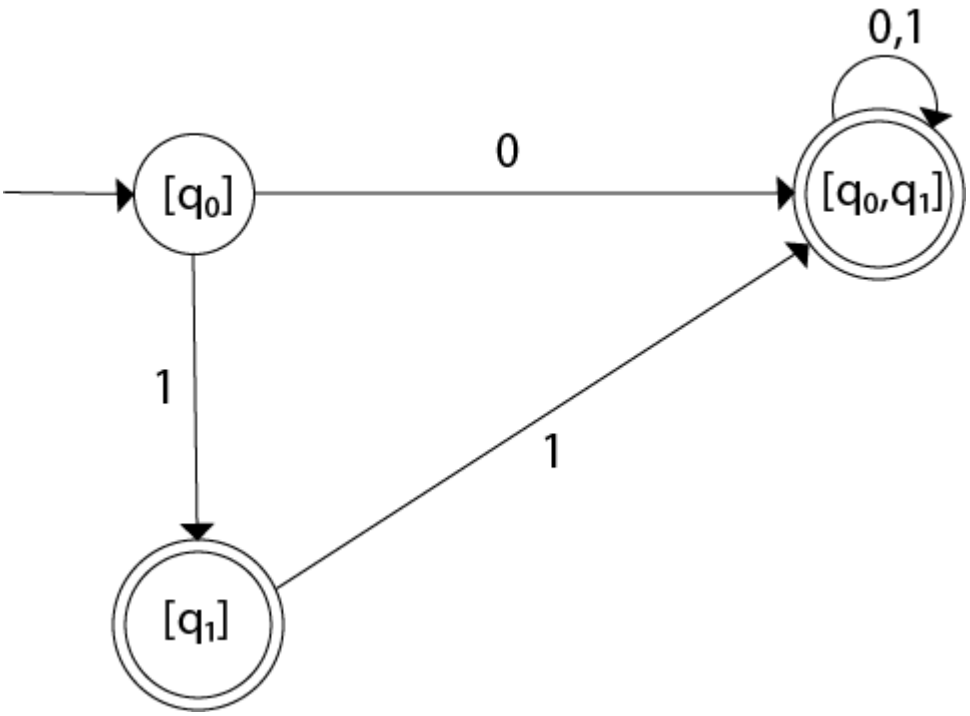
1. δ'([q0, q1], 1) = δ(q0, 1) ∪ δ(q1, 1)
2. = {q1} ∪ {q0, q1}
3. = {q0, q1}
4. = [q0, q1]

As in the given NFA, q1 is a final state, then in DFA wherever, q1 exists that state becomes a final state. Hence in the DFA, final states are [q1] and [q0, q1]. Therefore set of final states F = {[q1], [q0, q1]}.

The transition table for the constructed DFA will be:

State	0	1
→[q0]	[q0, q1]	[q1]
*[q1]	ϕ	[q0, q1]
*[q0, q1]	[q0, q1]	[q0, q1]

The Transition diagram will be:

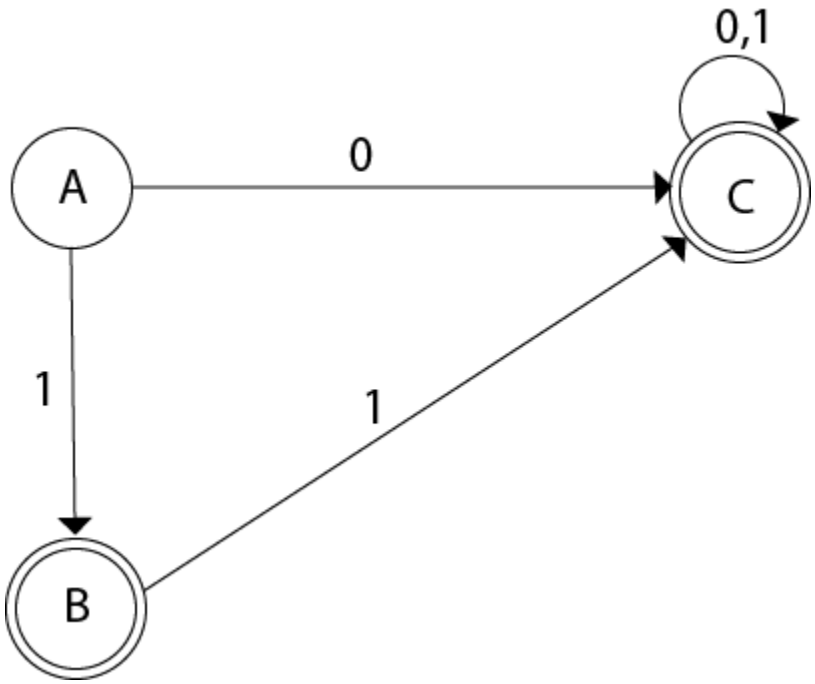


Even we can change the name of the states of DFA.

Suppose

1. A = [q0]
2. B = [q1]
3. C = [q0, q1]

With these new names the DFA will be as follows:



Conversion from NFA with ε to DFA

Non-deterministic finite automata(NFA) is a finite automata where for some cases when a specific input is given to the current state, the machine goes to multiple states or more than 1 states. It can contain ε move. It can be represented as M = { Q, ∑, δ, q0, F}.

Where

1. Q: finite set of states
2. ∑: finite set of the input symbol
3. q0: initial state
4. F: **final** state
5. δ: Transition function

NFA with ∈ move: If any FA contains ε transaction or move, the finite automata is called NFA with ∈ move.

ε-closure: ε-closure for a given state A means a set of states which can be reached from the state A with only ε(null) move including the state A itself.

Steps for converting NFA with ε to DFA:

Step 1: We will take the ε-closure for the starting state of NFA as a starting state of DFA.

Step 2: Find the states for each input symbol that can be traversed from the present. That means the union of transition value and their closures for each state of NFA present in the current state of DFA.

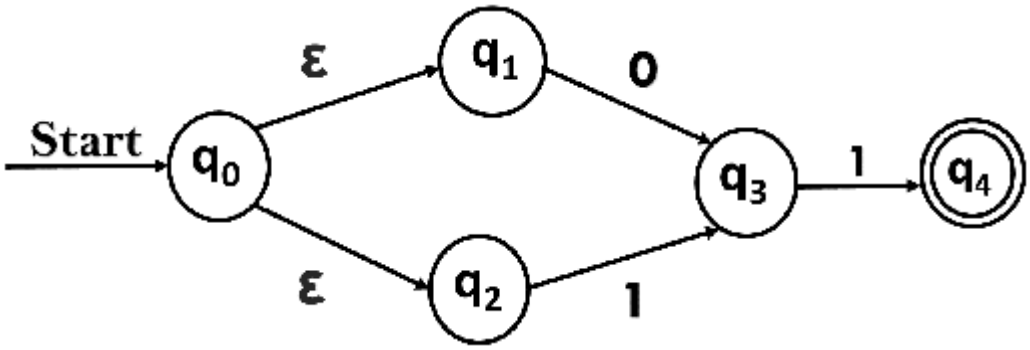
Step 3: If we found a new state, take it as current state and repeat step 2.

Step 4: Repeat Step 2 and Step 3 until there is no new state present in the transition table of DFA.

Step 5: Mark the states of DFA as a final state which contains the final state of NFA.

Example 1:

Convert the NFA with ε into its equivalent DFA.



Solution:

Let us obtain ε-closure of each state.

1. ε-closure {q0} = {q0, q1, q2}
2. ε-closure {q1} = {q1}
3. ε-closure {q2} = {q2}
4. ε-closure {q3} = {q3}

5. ε-closure {q4} = {q4}

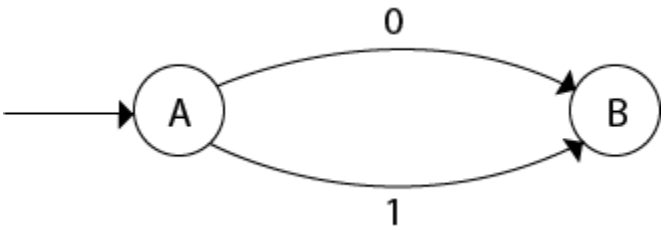
Now, let ε-closure {q0} = {q0, q1, q2} be state A.

Hence

δ'(A, 0) = ε-closure {δ((q0, q1, q2), 0) }
= ε-closure {δ(q0, 0) ∪ δ(q1, 0) ∪ δ(q2, 0) }
= ε-closure {q3}
= {q3} **call it as state B**.

δ'(A, 1) = ε-closure {δ((q0, q1, q2), 1) }
= ε-closure {δ((q0, 1) ∪ δ(q1, 1) ∪ δ(q2, 1) }
= ε-closure {q3}
= {q3} = B.

The partial DFA will be



Now,

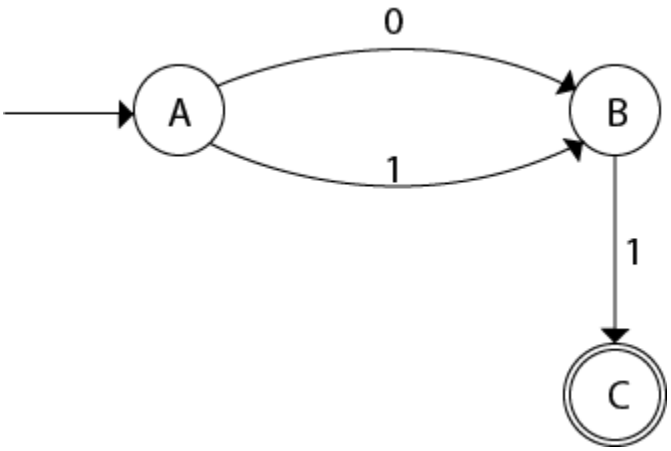
δ'(B, 0) = ε-closure {δ(q3, 0) }
= ϕ
δ'(B, 1) = ε-closure {δ(q3, 1) }
= ε-closure {q4}
= {q4} **i.e. state C**

For state C:

1. δ'(C, 0) = ε-closure {δ(q4, 0) }

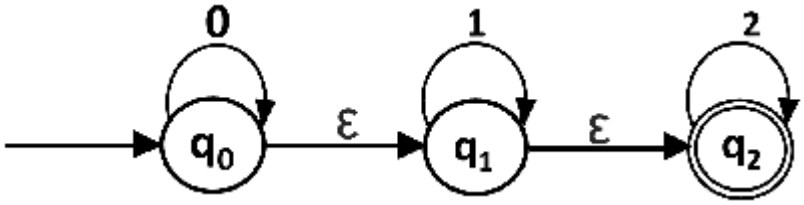
2. = ϕ
3. δ'(C, 1) = ε-closure {δ(q4, 1) }
4. = ϕ

The DFA will be,



Example 2:

Convert the given NFA into its equivalent DFA.



Solution: Let us obtain the ε-closure of each state.

1. ε-closure(q0) = {q0, q1, q2}
2. ε-closure(q1) = {q1, q2}
3. ε-closure(q2) = {q2}

Now we will obtain δ' transition. Let ε-closure(q0) = {q0, q1, q2} call it as **state A**.

δ'(A, 0) = ε-closure{δ((q0, q1, q2), 0)}
= ε-closure{δ(q0, 0) ∪ δ(q1, 0) ∪ δ(q2, 0)}

= ε-closure{q0}
= {q0, q1, q2}

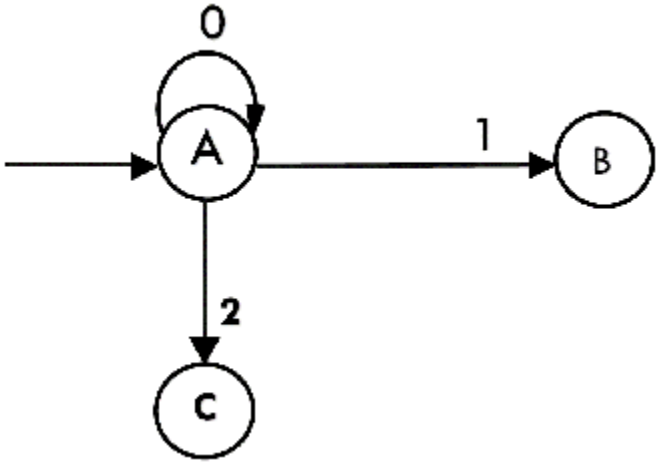
δ'(A, 1) = ε-closure{δ((q0, q1, q2), 1)}
= ε-closure{δ(q0, 1) ∪ δ(q1, 1) ∪ δ(q2, 1)}
= ε-closure{q1}
= {q1, q2} **call it as state B**

δ'(A, 2) = ε-closure{δ((q0, q1, q2), 2)}
= ε-closure{δ(q0, 2) ∪ δ(q1, 2) ∪ δ(q2, 2)}
= ε-closure{q2}
= {q2} **call it state C**

Thus we have obtained

1. δ'(A, 0) = A
2. δ'(A, 1) = B
3. δ'(A, 2) = C

The partial DFA will be:



Now we will find the transitions on states B and C for each input.

Hence

δ'(B, 0) = ε-closure{δ((q1, q2), 0)}
= ε-closure{δ(q1, 0) ∪ δ(q2, 0)}
= ε-closure{ϕ}
= ϕ

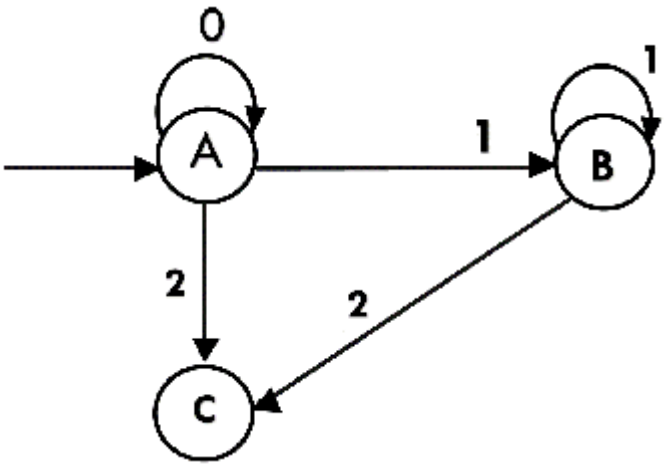
δ'(B, 1) = ε-closure{δ((q1, q2), 1)}
= ε-closure{δ(q1, 1) ∪ δ(q2, 1)}
= ε-closure{q1}
= {q1, q2} **i.e. state B itself**

δ'(B, 2) = ε-closure{δ((q1, q2), 2)}
= ε-closure{δ(q1, 2) ∪ δ(q2, 2)}
= ε-closure{q2}
= {q2} **i.e. state C itself**

Thus we have obtained

1. δ'(B, 0) = ϕ
2. δ'(B, 1) = B
3. δ'(B, 2) = C

The partial transition diagram will be



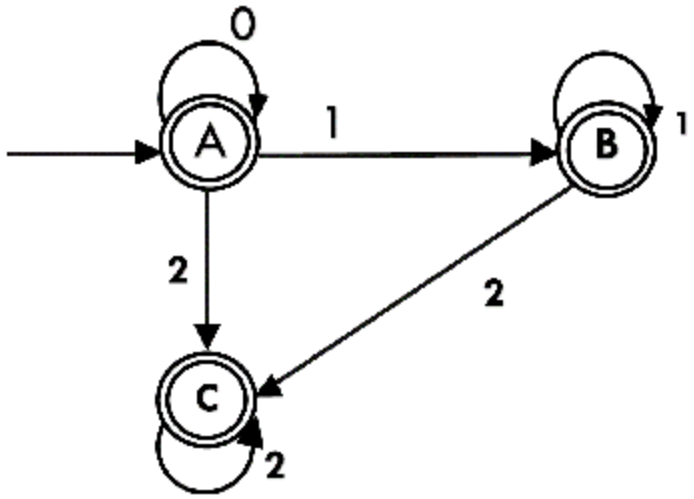
Now we will obtain transitions for C:

δ'(C, 0) = ε-closure{δ(q2, 0)}
= ε-closure{ϕ}
= ϕ

δ'(C, 1) = ε-closure{δ(q2, 1)}
= ε-closure{ϕ}
= ϕ

δ'(C, 2) = ε-closure{δ(q2, 2)}
= {q2}

Hence the DFA is



As A = {q0, q1, q2} in which final state q2 lies hence A is final state. B = {q1, q2} in which the state q2 lies hence B is also final state. C = {q2}, the state q2 lies hence C is also a final state.

Lecture 5: Regular Expressions

1. Introduction

* Describe patterns in strings
* Equivalence with finite automata

2. Syntax

* Operators:

  * Concatenation: ab
  * Union: a|b

  * Kleene star: a*
  * Parentheses for grouping

3. Examples

* Strings over {a, b} with even number of a's: (b*ab*ab*)*
* Binary strings divisible by 3

4. Converting Between FA and RE

* Thompson’s construction (NFA from RE)
* State elimination (FA to RE)

Regular expression	output(set of strings)
λ	{λ}
λ*	{λ}
a	{ a }
aa	{ aa }
a*	{λ, a, aa, aaa, ….}
aa*	{ a, aa, aaa, ... }
a+	{ a, aa, aaa, ...}
ba+	{ ba, baa, baaa, ...}
(ba)+	{ ba, baba, bababa, ...}
(alb)	{ a, b }
alb*	{ a, λ, b, bb, bbb, ... }
(alb)*	{ λ, a, b, aa, ab, ba, bb, ... }
aa(ba)*bb	{ aabb, aababb, aabababb, ... }
(a + a)	{a}
(a + b)	{a, b}
(a + b)2	(a + b)(a + b) == {aa, ab, ba, bb}
(a + b + c)	{a, b, c}
(a + b)*	{λ, a, b, aa, bb, ab, ba, aaa, bbb, aab, bba, ….}
(abc)	{abc}
(λ + a) bc	{bc, abc}
ab*	{a, ab, abb, abbb, …}
(ab)*	{λ, ab, abab, ababab, …}
a + b*	{a, λ, b, bb, bbb, …}
a (a + b)*	{a, aa, ab, aaa, abb, aba, abaa, … }
(a + b)* a (a + b)*	{a, aaa, aab, baa, bab, …}
(a + λ)*	(a)* = {λ, a, aa, aaa, ….}
x* (a + b) + (a + b)	x* (a + b)
x* y + y	x* y
(x + λ)x*	x* (x + λ) = x*
(x + λ)(x + λ)* (x + λ)	x*