CS3063 Theory of Computing

Semester 4 (21 Intake), Jan – May 2024

Lecture 3

Regular Languages & Finite Automata – Session 2

Sanath Jayasena

Announcement on Quizzes

- Students must be present in the lab
- Quiz attempts must only be from the lab computers
- If there are N quizzes in the semester, the best N-2 quizzes will be counted for each student
  - 2 spare quizzes for each student

- Unexpected issues (e.g., computer getting stuck), absent due to sickness, …
  - 2 spare quizzes are meant to cater to such cases

Today's Outline

Lecture 3

- FA $\leftrightarrow$ Regular expressions: How?
- Distinguishing strings
- Set operations on regular languages
- Non-deterministic Finite Automata (NFA)
- Equivalence between NFA and FA (DFA)

PART 1

Today's Outline

Lecture 3

- FA $\leftrightarrow$ Regular expressions: How?
- Distinguishing strings
- Set operations on regular languages
- Non-deterministic Finite Automata (NFA)
- Equivalence between NFA and FA (DFA)

Review: Regular Languages and FA

- **Kleene's Theorem**
  - *A language $L \subseteq \Sigma^*$ is regular if and only if there is an FA with alphabet Σ that accepts L*
- This means:
  - If M is an FA, there is a regular expression corresponding to the language $L(M)$
  - Given a regular expression, there is an FA that accepts the corresponding language

Regular Languages and FA

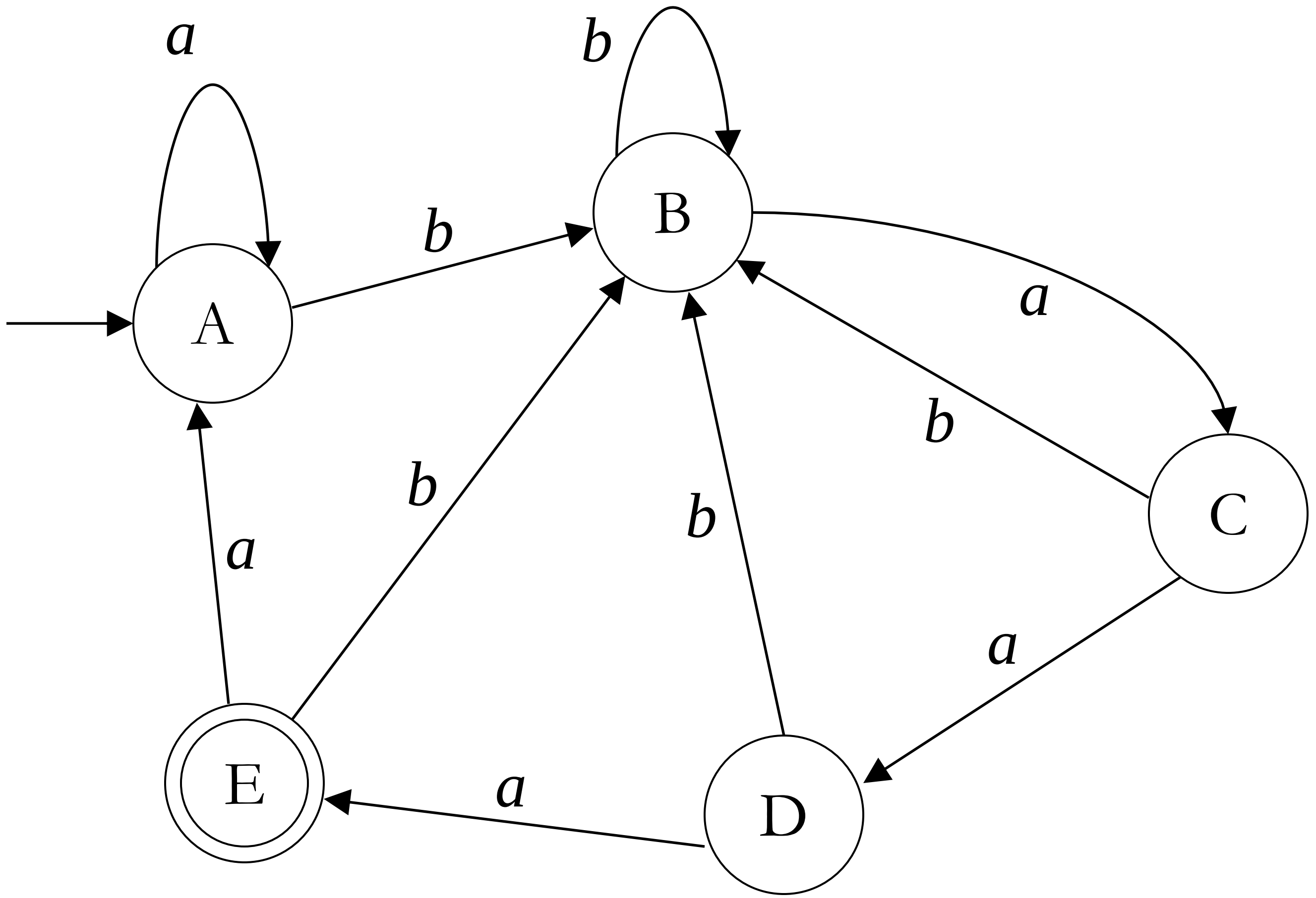
- How to get FA, given regular expression (and vice-versa)?
  - Will discuss later
  - When discussing proof of Kleene’s Theorem

  - Until then, try to do this without an algorithm

Obtaining RE, given the FA?

- Intuitive approach (brute force)
- Can study the set of states and inputs on the transition diagram
- Start with simple strings
- Consider all paths/cases
- But some FAs can be difficult, experience will help

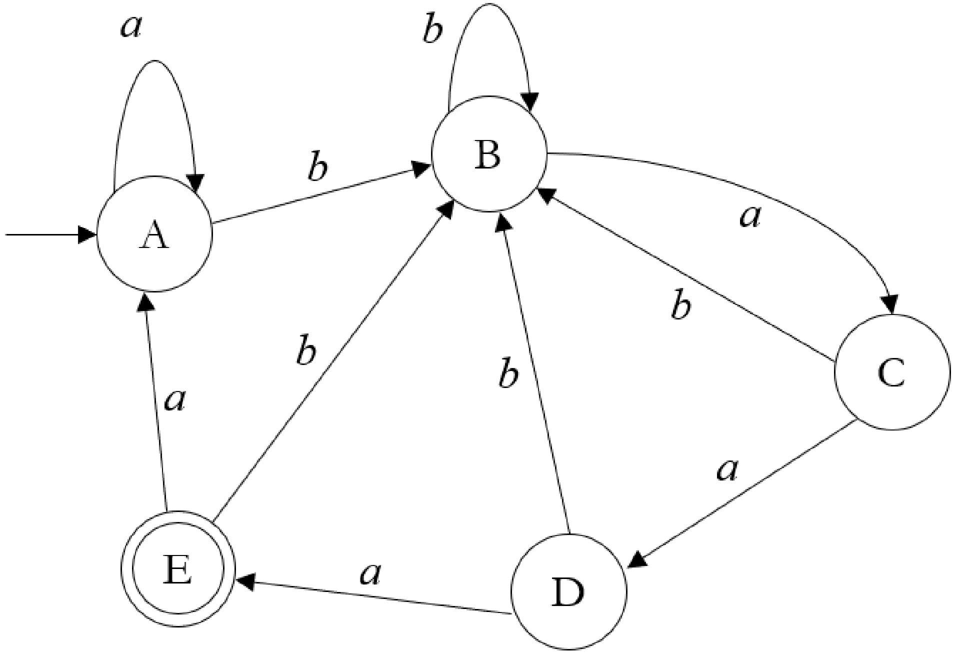
Example



What are the strings accepted by this FA?

Solution

- For any string ending in $b \Rightarrow$ go to state B
- The only way to get to state:
  - E is from state D with input a
  - D is from state C with input a
  - C is from state B with input a
  - B is with input b from any state
- The language accepted
  - Set of all strings ending in $baaa \Rightarrow (a\,|b)^*baaa$



Obtaining FA, given the RE

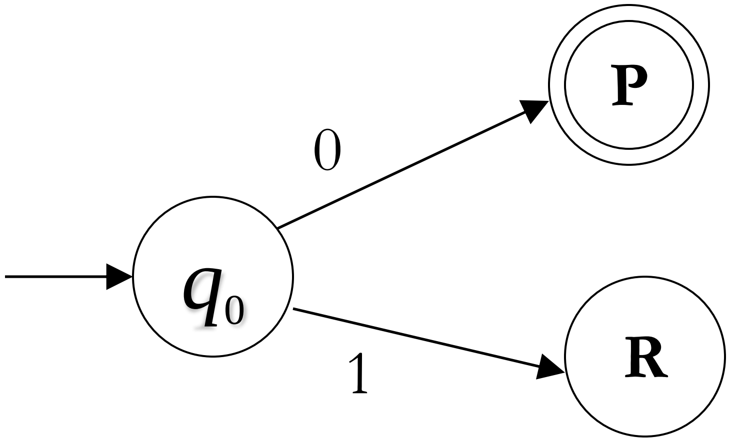
- Observe the given set for key patterns
- May be able to identify the states quickly
  - Depends on the given regular set
- Example
  - Construct the FA that accepts the language *L* corresponding to (11 | 110)*0

- [Note: easier method using NFAs discussed later]

Solution: the FA for $L \equiv (11|110)^*0$

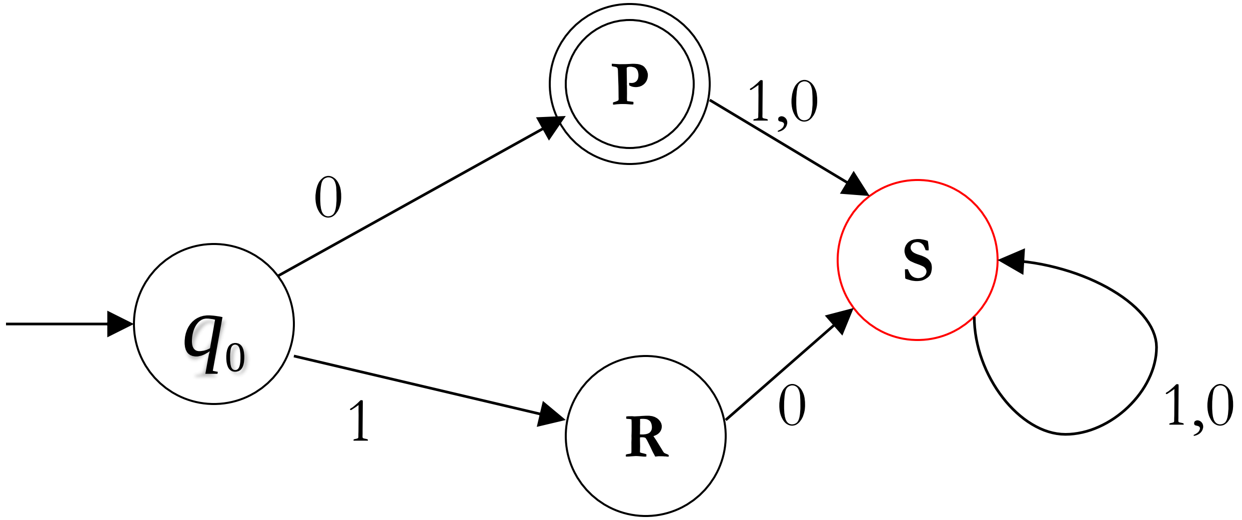
- Λ is not in $L \Rightarrow q_0$ is not an accepting state
- 0 is in $L \Rightarrow$ from q_0 input 0 takes us to an accepting state
- 1 is not in L; 1 and Λ needs to be distinguished
  – 110 is in L, but 1110 is not in L

Need at least these states



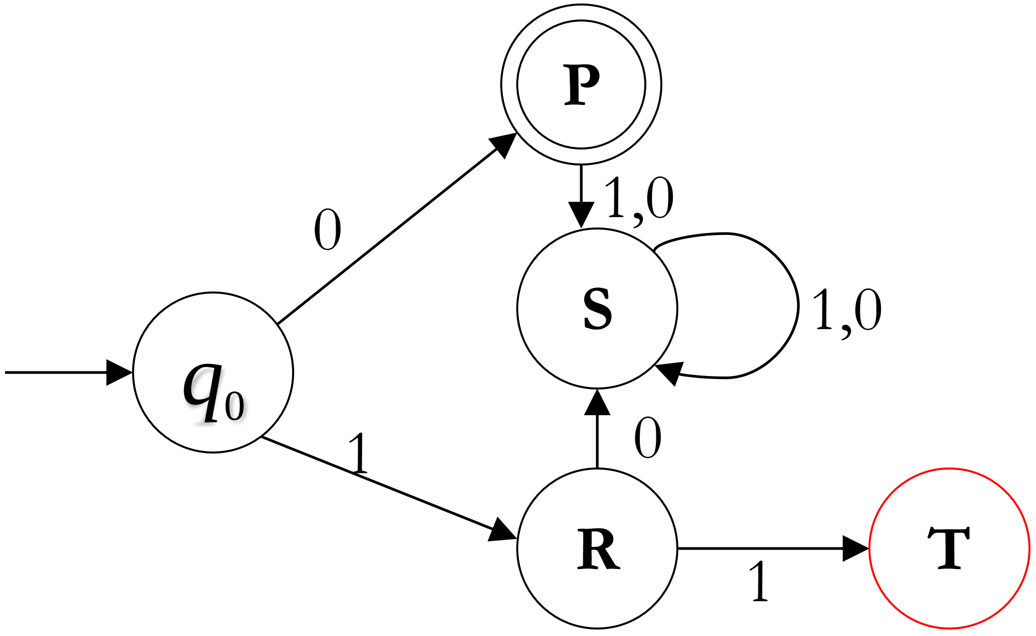
Solution: the FA for $L \equiv (11|110)^*0$

- L contains 0 but no string of the form $0x$
- L contains no string of the form $10x$
- Can add another state S to represent above 2 unaccepted forms; when you go there, you never leave



Solution: the FA for $L \equiv (11|110)^*0$

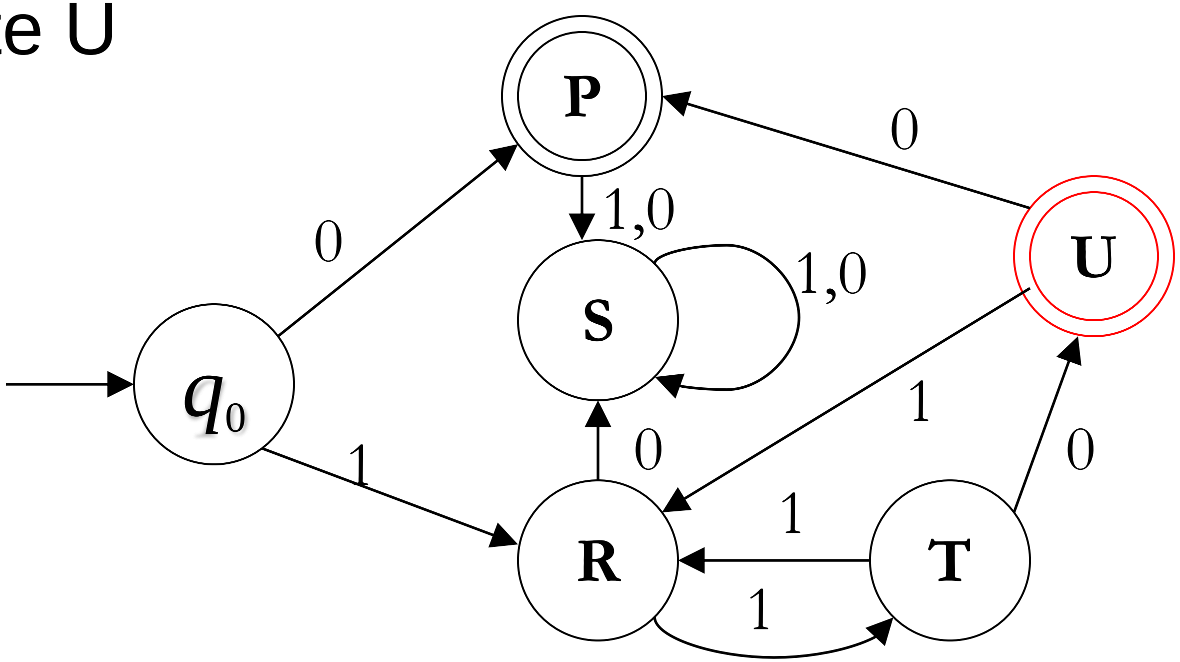
- What happens at R for input 1?
  - Shouldn't stay at R (1 and 11 to be distinguished)
  - Shouldn't return to q_0 (Λ and 11 to be distinguished)
  - Need a new state, T



Solution: the FA for $L \equiv (11|110)^*0$

- What happens at T?
  - As we did so far, consider all inputs
  - Need an accepting state U

- Final solution



Obtaining FA, given the RE

- May not be able to identify the states quickly

- As we saw in the example, can keep on adding states
  - Eventually ends because language is regular
  - If language is not regular, with the same approach, we will continue forever

PART 2

Today's Outline

Lecture 3

- FA ↔ Regular expressions: How?
- Distinguishing strings
- Set operations on regular languages
- Non-deterministic Finite Automata (NFA)
- Equivalence between NFA and FA (DFA)

Distinguishing Strings

- Consider an FA recognizing a language L
  - There are *groups of strings* where strings within the same group need not be distinguished from each other by FA
  - Remembering which group a string belongs to is enough when it is reading a string
  - The *number of distinct states the FA needs* to recognize L is related to the *number of distinct strings to be distinguished from each other*

Distinguishable Strings

- **Definition** (Definition 3.5, p. 105 in text)
  - Let L be a language in Σ^* and x and y be any strings in Σ^*. The set L $/x$ is defined as

    L $/x$ = {z in Σ^* | xz is in L}

  - Two strings x and y are distinguishable with respect to L if L $/x \neq$ L $/y$. Any z that is in one of the two sets but not the other is said to distinguish x and y w.r.t. L
  - If L $/x$=L $/y$, x and y are indistinguishable with respect to L

Two Important Properties

- Property 1 (Theorem 3.2 in text, p. 106)
  - Suppose $L \subseteq \Sigma^*$ and for some +ve integer ***n***, there are ***n*** strings in Σ*, any two of which are distinguishable w.r.t. L. Then every FA recognizing L must have at least ***n*** states
- Property 2 (Theorem 3.3 in text, p. 108)
  - The language ***pal*** of palindromes over the alphabet {0,1} cannot be accepted by any FA and therefore not regular

Shows a lower bound on the memory requirements of an FA to recognize a language

Set Operations and Languages

- Suppose $M_1=(Q_1, \Sigma, q_1, A_1, \delta_1)$ and $M_2=(Q_2, \Sigma, q_2, A_2, \delta_2)$ accept languages L_1 and L_2
- Let $M=(Q, \Sigma, q_0, A, \delta)$ where

$$Q = Q_1 \times Q_2$$

$$q_0 = (q_1 , q_2)$$

$$\delta((p, q), a)= (\delta_1(p, a), \delta_2(q, a))$$

- Then the following hold

Set Operations & Languages ...contd

1. If A={(p, q) | p is in A_1 or q is in A_2}, then M accepts the language $L_1 \cup L_2$

2. If A={(p, q) | p is in A_1 and q is in A_2}, then M accepts the language $L_1 \cap L_2$

3. If A={(p, q) | p is in A_1 and q is not in A_2}, then M accepts the language $L_1 - L_2$

PART 3

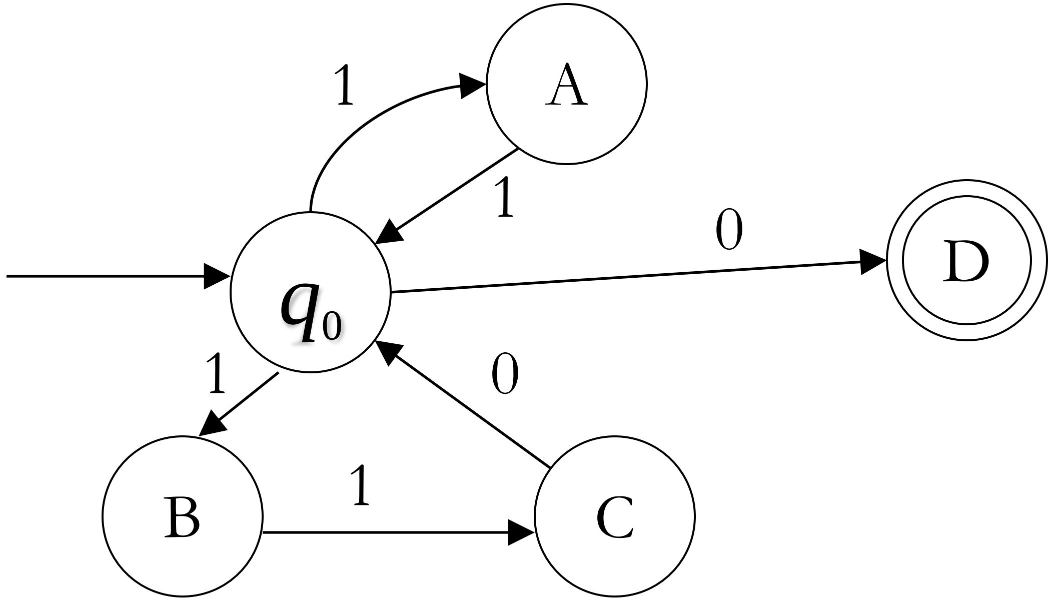
Today's Outline

Lecture 3

- FA ↔ Regular expressions: How?
- Distinguishing strings
- Set operations on regular languages
- **Non-deterministic Finite Automata (NFA)**
- Equivalence between NFA and FA (DFA)

Nondeterministic Finite Automata (NFA)

- An NFA differs from a deterministic FA (or DFA) on δ
  - Allows zero, one or more transitions from a state on the same input symbol
  - So, the value of δ is a *set of states*



This NFA accepts (11 | 110)*0 as the DFA on slide 14

Definition of NFA

- An NFA is a 5-tuple $(Q, \Sigma, q_0, A, \delta)$, where:
  - Q, Σ, q_0 and A have the same meaning as for a DFA, but…
  - δ, the transition function, maps $Q \times \Sigma$ to 2^Q
  - (2^Q is the power set of Q, the set of all subsets of Q)

Extended Transition Function δ*

- We can extend δ, as with DFA, to describe the status of an NFA on an input string x
- Definition: The function δ^* : $Q \times \Sigma^* \rightarrow 2^Q$ is such that:
  - For any $q \in Q$, $\delta^*(q, \Lambda) = \{q\}$
  - For any $q \in Q$, $y \in \Sigma^*$ and $a \in \Sigma$,

$$\delta^*(q, ya) = \bigcup_{r \in \delta^*(q,y)} \delta(r, a)$$

Properties of NFAs

- Any language accepted by an NFA is also accepted by a DFA
- Constructing an NFA for a regular expression is often simpler
- NFA are useful for proving theorems
- There is a procedure to convert an NFA into an equivalent DFA
- DFA (Deterministic FA) is a special case of NFA

Acceptance by an NFA

- A string is accepted by an NFA if there is a sequence of transitions for it leading from the initial state to an accepting state

- That is, an NFA, M, accepts a string x if the set of states M can end up after processing x contains at least one accepting state

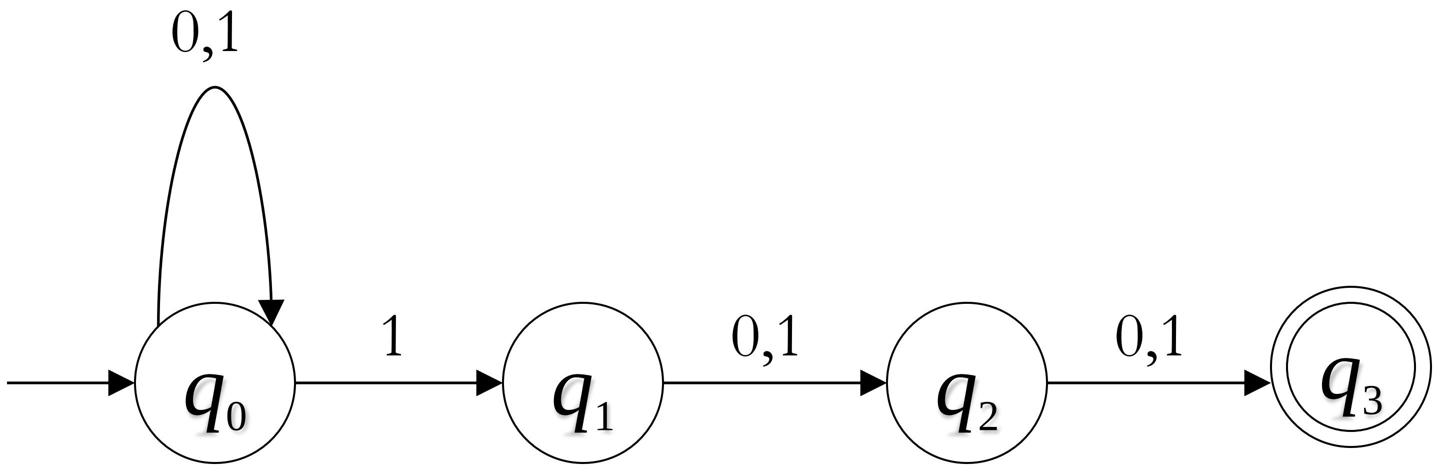
Example

- Suppose the NFA, M=(Q, Σ, q_0, A, δ) where Q={q_0, q_1, q_2, q_3}, Σ={0,1}, A={q_3} and δ specified as follows is given.

q	$\delta(q, 0)$	$\delta(q, 1)$
q_0	$\{q_0\}$	$\{q_0, q_1\}$
q_1	$\{q_2\}$	$\{q_2\}$
q_2	$\{q_3\}$	$\{q_3\}$
q_3	$\varnothing$	$\varnothing$

1. Draw the transition diagram
2. Determine the language accepted by M

Solution



Language accepted? $(0 \mid 1)^*1(0 \mid 1)^2$

PART 4

Today's Outline

Lecture 3

- FA ↔ Regular expressions: How?
- Distinguishing strings
- Set operations on regular languages
- Non-deterministic Finite Automata (NFA)
- **Equivalence between NFA and FA (DFA)**

Equivalence Between NFA & DFA

- **Theorem**: For an NFA, $M=(Q, \Sigma, q_0, A, \delta)$ accepting a language $L \subseteq \Sigma^*$, there is a DFA, $M_1=(Q_1, \Sigma, q_1, A_1, \delta_1)$ that accepts L
- M_1 can be defined such that:

  $Q_1 = 2^Q$, $q_1=\{q_0\}$,

  for $q \in Q_1$ and $a \in \Sigma$, $\delta_1(q,a) = \bigcup_{r \in q} \delta(r,a)$

  $A_1 = \{q \in Q_1 \mid q \cap A \neq \varnothing\}$

DFA Equivalent to an NFA: How?

- From the theorem on equivalency
  - Proof gives a method to obtain equivalent DFA
  - Proof by induction (on length of input string)

- Method based on *subset construction*
  - *A set of states in the NFA* is considered as *a state in the DFA*
  - DFA keeps track of all states that the NFA could be in after reading the same input as the DFA has read

Example

- Consider the NFA (Example on slide 28): $M=(Q, \Sigma, q_0, A, \delta)$ where $Q=\{q_0, q_1, q_2, q_3\}$, $\Sigma=\{0,1\}$, $A=\{q_3\}$ and δ specified as follows;

q	$\delta(q, 0)$	$\delta(q, 1)$
q_0	$\{q_0\}$	$\{q_0, q_1\}$
q_1	$\{q_2\}$	$\{q_2\}$
q_2	$\{q_3\}$	$\{q_3\}$
q_3	$\emptyset$	$\emptyset$

Find an equivalent DFA

Solution: Approach

- Subset construction could produce a DFA with 16 (2^4) states
  - Because the NFA has 4 states
- But we may get fewer states if we consider only states reachable from initial state
  - Start from q_0
  - Each time a new state (subset) S appears, then compute new state from S for each input

Solution ...contd

q	$\delta_1(q, 0)$	$\delta_1(q, 1)$
$\{q_0\}$	$\{q_0\}$	$\{q_0, q_1\}$
$\{q_0, q_1\}$	$\{q_0, q_2\}$	$\{q_0, q_1, q_2\}$

Solution ...contd

q	$\delta_1(q, 0)$	$\delta_1(q, 1)$
$\{q_0\}$	$\{q_0\}$	$\{q_0, q_1\}$
$\{q_0, q_1\}$	$\{q_0, q_2\}$	$\{q_0, q_1, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_3\}$	$\{q_0, q_1, q_3\}$
$\{q_0, q_1, q_2\}$	$\{q_0, q_2, q_3\}$	$\{q_0, q_1, q_2, q_3\}$
$\{q_0, q_3\}$	$\{q_0\}$	$\{q_0, q_1\}$
$\{q_0, q_1, q_3\}$	$\{q_0, q_2\}$	$\{q_0, q_1, q_2\}$
$\{q_0, q_2, q_3\}$	$\{q_0, q_3\}$	$\{q_0, q_1, q_3\}$
$\{q_0, q_1, q_2, q_3\}$	$\{q_0, q_2, q_3\}$	$\{q_0, q_1, q_2, q_3\}$

Conclusion

- We discussed today
  - FA $\leftrightarrow$ Regular expressions
  - Distinguishing strings
  - Set operations on regular languages
  - Non-deterministic FA (NFA)
  - NFA$\leftrightarrow$DFA Equivalency
  - Finding an equivalent DFA for a given NFA