60-354, Theory of Computation
Fall 2013

Asish Mukhopadhyay
School of Computer Science
University of Windsor

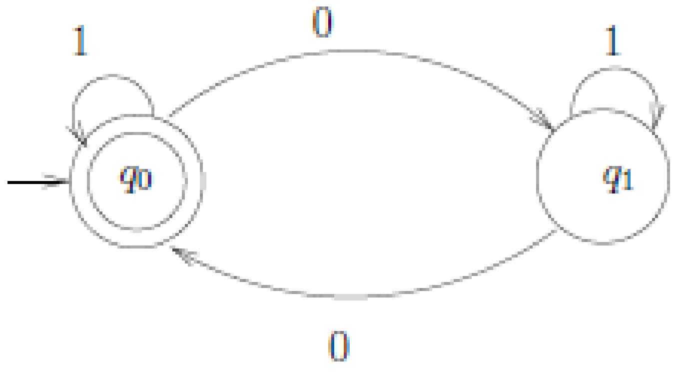
Formal Definition of a DFA

- A DFA is a 5-tuple : (Q, ∑, δ, q_0, F)
  - Q is the set of states
  - ∑ is the input alphabet
  - δ is the transition function
    - δ : Q X ∑ -> Q
  - F , a subset of Q, is the set of final states

Example

- Q = $\{q_0, q_1\}$
- ∑ = {0,1}
- F = {q0}

δ	0	1
q_0	q_1	q_0
q_1	q_0	q_1



Extended transition function

$$\hat{\delta}(q, e) = q$$

$$\hat{\delta}(q, aw) = \hat{\delta}(\delta(q, a), w)$$

Transition function Example

$$\begin{aligned}\hat{\delta}(q_0,001) &= \hat{\delta}(\delta(q_0,0),01)\\ &= \hat{\delta}(q_1,01)\\ &= \hat{\delta}(\delta(q_1,0),1)\\ &= \hat{\delta}(q_0,1)\\ &= \hat{\delta}(\delta(q_0,1),\varepsilon)\\ &= \hat{\delta}(q_0,\varepsilon)\\ &= q_0\end{aligned}$$

Language accepted by a DFA

- L = { w in Σ^* | $\hat{\delta}(q_0, w)$ is an accepting state of *A*}

Regular Language

- Language accepted by a DFA

DFA Constructions

- Example 2
  - Construct a DFA that accepts all strings over {0,1} such that the reverse of *w,* when evaluated in decimal, is divisible by 5 (or, multiple of 5)

Reverse of a string

- If $w = x_1 x_2 .. x_k$ is a string, then $w^r = x_k x_{k-1},...,x_1$
- Thus if w = 1101, w^r = 1011

Main Observation

- The contribution , modulo 5, of a current 1 bit is periodic with respect to its position from the left.
- That's because:
  - 2^0 mod 5 = 1 2^4 mod 5 = 1
  - 2^1 mod 5 = 2 2^5 mod 5 = 2 ……….
  - 2^2 mod 5 = 4 2^6 mod 5 = 4
  - 2^3 mod 5 = 3 2^7 mod 5 = 3

State description

- $q_{i,j}$
  - *i* is the current position in the string modulo 4
  - *j* is the cumulative reminder modulo 5 of the string seen so far

Transition function

- $\delta(q_{i,j,}\, 0) = q_{(i+1) \bmod 4,\, j}$
- $\delta(q_{i,j,}\, 1) = q_{(i+1) \bmod 4,\, (j + 2^{(i+1) \bmod 4}) \bmod 5}$
- Start state : $q_{-1,0}$

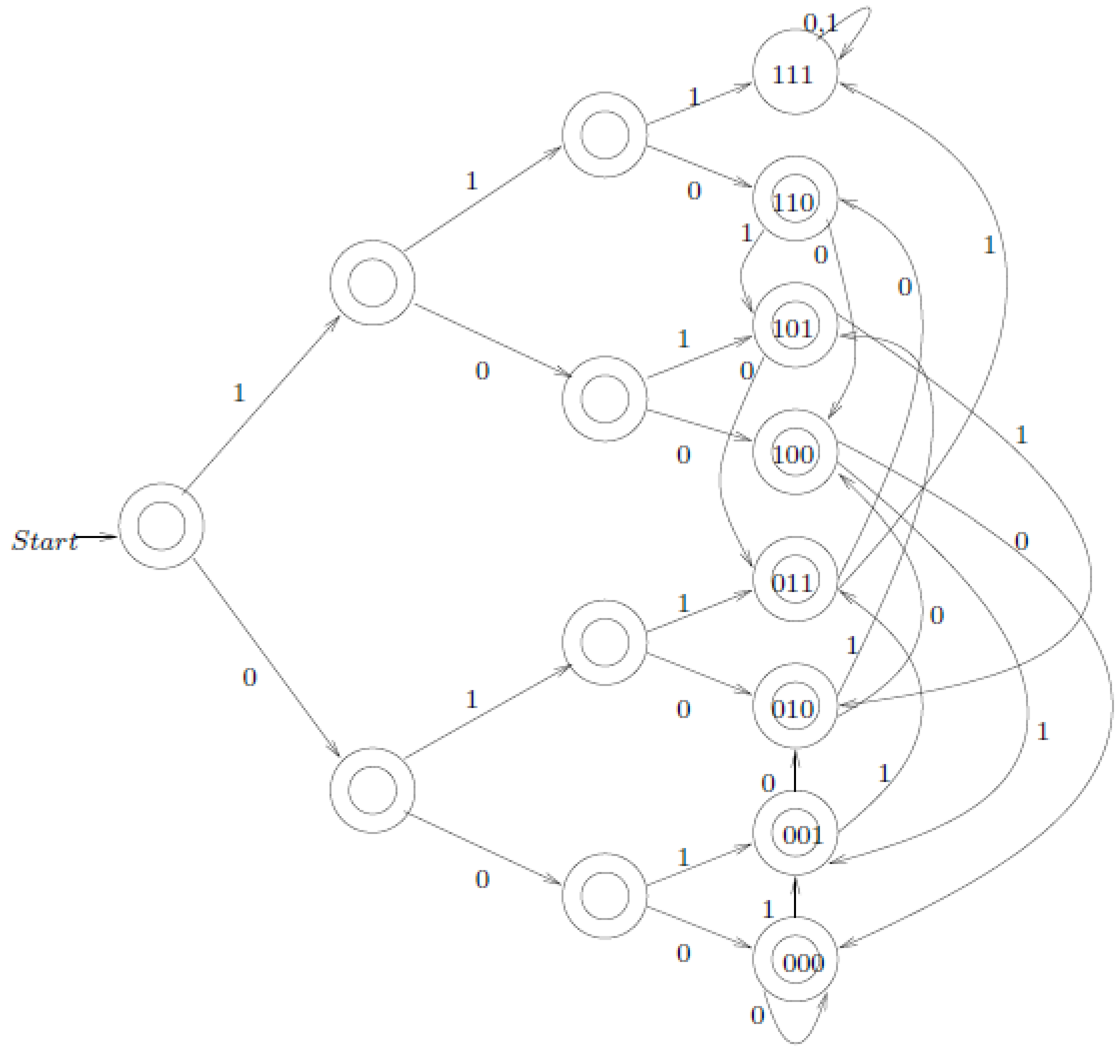
DFA constructions

- Example 3
  - The set of all strings such that each block of five consecutive symbols contains at least two 0’s.

Solution

- Build a DFA, maintaining 2 pieces of information
  - The decimal value of the current block of 5 bits
  - The number of 0's in it
- Updating the decimal value as we move one place right:
  - (oldValue * 2) mod 32 + decimal value of the new bit (this value tells us whether the leading bit of the block of 5 bits is a 1 or a 0)

DFA for modified Example 3



Example 4

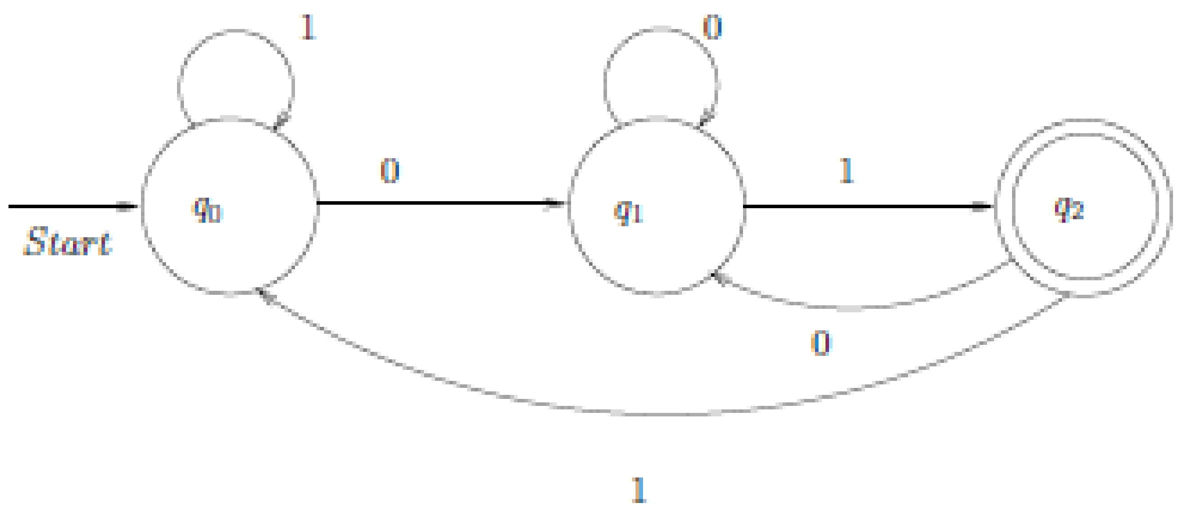


Figure 2.4: A DFA that accepts all strings that end in 01

Example 4

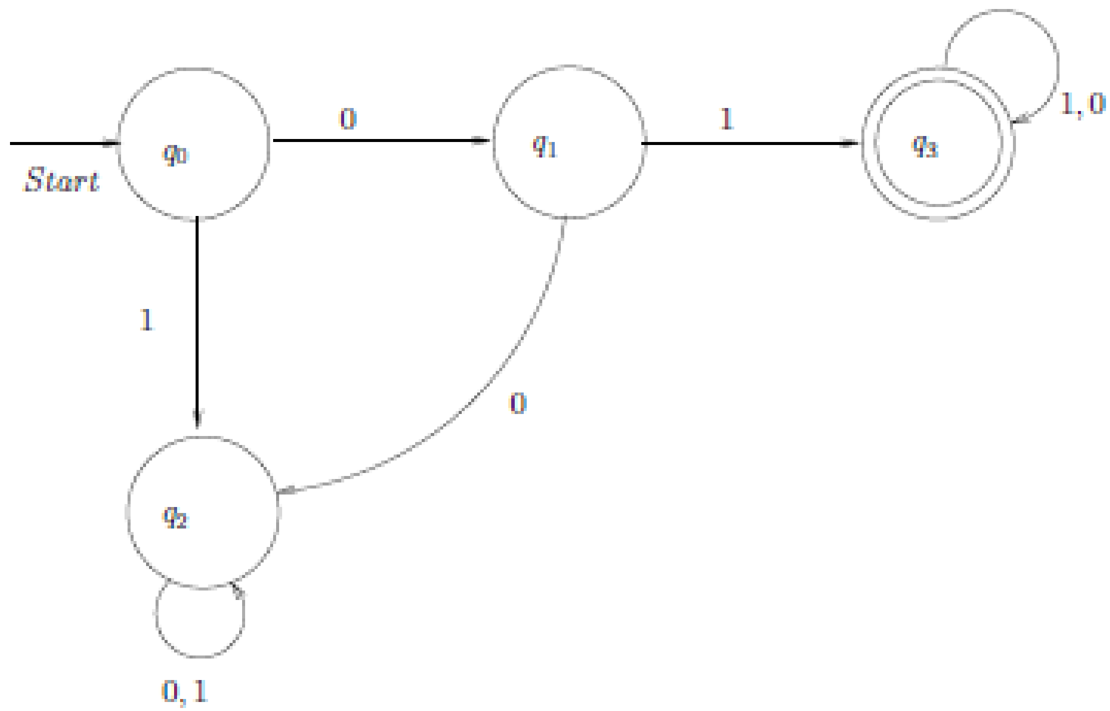


Figure 2.5: A DFA that accepts all strings that begin with 01

Product automaton

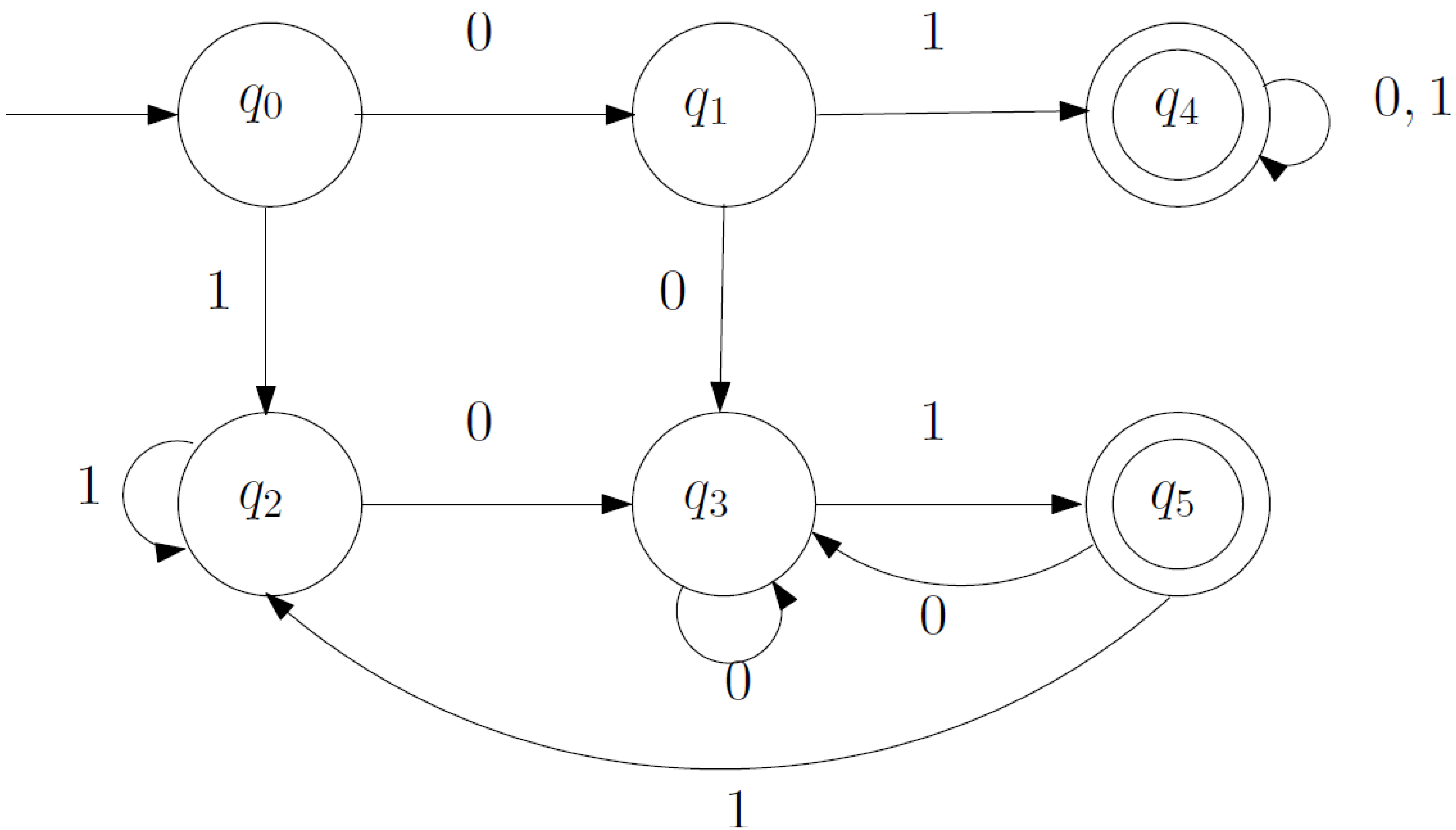
- Let DFA $A_i = (Q_i, \sum, \delta_i, q_{i0}, F_i)$, for i = 1, 2, recognize language L_i over $\sum$.
- Then the automaton
  - $A_1 X A_2 = (Q_1 X Q_2, \sum, \delta, (q_{01}, q_{02}), F_1 X F_2)$, where
    $\delta((q_{1i}, q_{2j}), a) = (\delta_1(q_{1i}, a), \delta_2(q_{2j}, a))$, for all a in $\sum$

  is called the *product automaton* of A_1 and A_2
- The language accepted by $A_1 X A_2$ is the set of all strings in $L_1 \cap L_2$

Solving the last problem

- Now use the idea of a product automaton to construct an automaton with 12 states that accepts all strings the begin with *and* end in 01.

A DFA accepting strings beginning with 01 *or* ending in 01



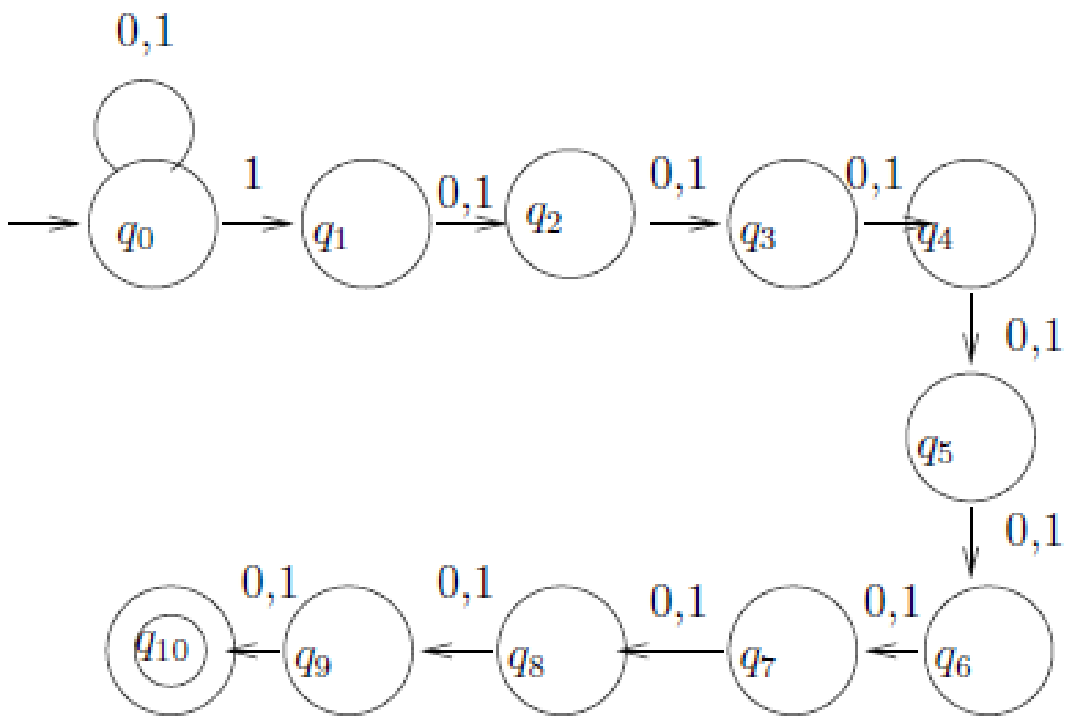
Nondeterministic Finite Automata

- NFA, for short
- Allow transitions from a given state on a given input to any one of a finite number of states or no state at all

Problem

- Construct an automaton that accepts all strings over $\Sigma = \{0, 1\}$ such that the tenth (10^{th}) symbol from the right end is a 1

Example 5



An NFA that accepts strings such that the tenth symbol from the right end is a 1

Designing a DFA

- Not straightforward
- Simpler problem
  - Construct a DFA that accepts strings whose second digit from the right is a 1

Solution

- Compute modulo 4 the decimal value of the string seen thus far
- If the value is 2 or 3 when we come to the end of the string the second bit from the right is 1
- Update value as we advance one place right:
  - Multiply previous value by 2, add the current bit and compute the result modulo 4

Transition table

	0	1
$\rightarrow q_0$	q_0	q_1
q_1	q_2	q_3
$*q_2$	q_0	q_1
$*q_3$	q_2	q_3

The index j of q_j is the remainder modulo 4

Formal definition of an NFA

- A 5-tuple $(Q, \sum, \delta, q_0, F)$ where

$$\delta : Q \times \sum \rightarrow 2^Q$$

Extended transition function

$$\hat{\delta}(q, \varepsilon) = \{q\}$$

$$\hat{\delta}(q, wa) = \bigcup_r \delta(r, a), \text{ where } r \in \hat{\delta}(q, w)$$

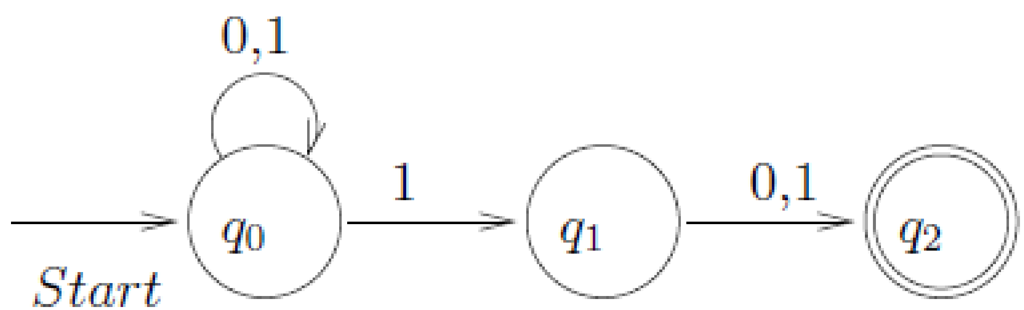
Language accepted by NFA

- L = {w in $\sum^*$ | $\hat{\delta}(q_0, w)$ intersection F is not empty}

NFA to DFA reduction

- Subset construction technique
  - The states of the DFA are all possible subsets of the states of the NFA
  - The start state is: $\{q_0\}$
  - Final states:
    - All subsets that contain at least one of the accepting states of the NFA

Construction by example (1)



Construction by example (2)

- If $\delta_D()$ is the transition function of the DFA, and S is a subset of the states of the NFA then

$$\delta_D(S,a) = \bigcup_q \delta_N(q,a),\ q \in S$$

Equivalent DFA

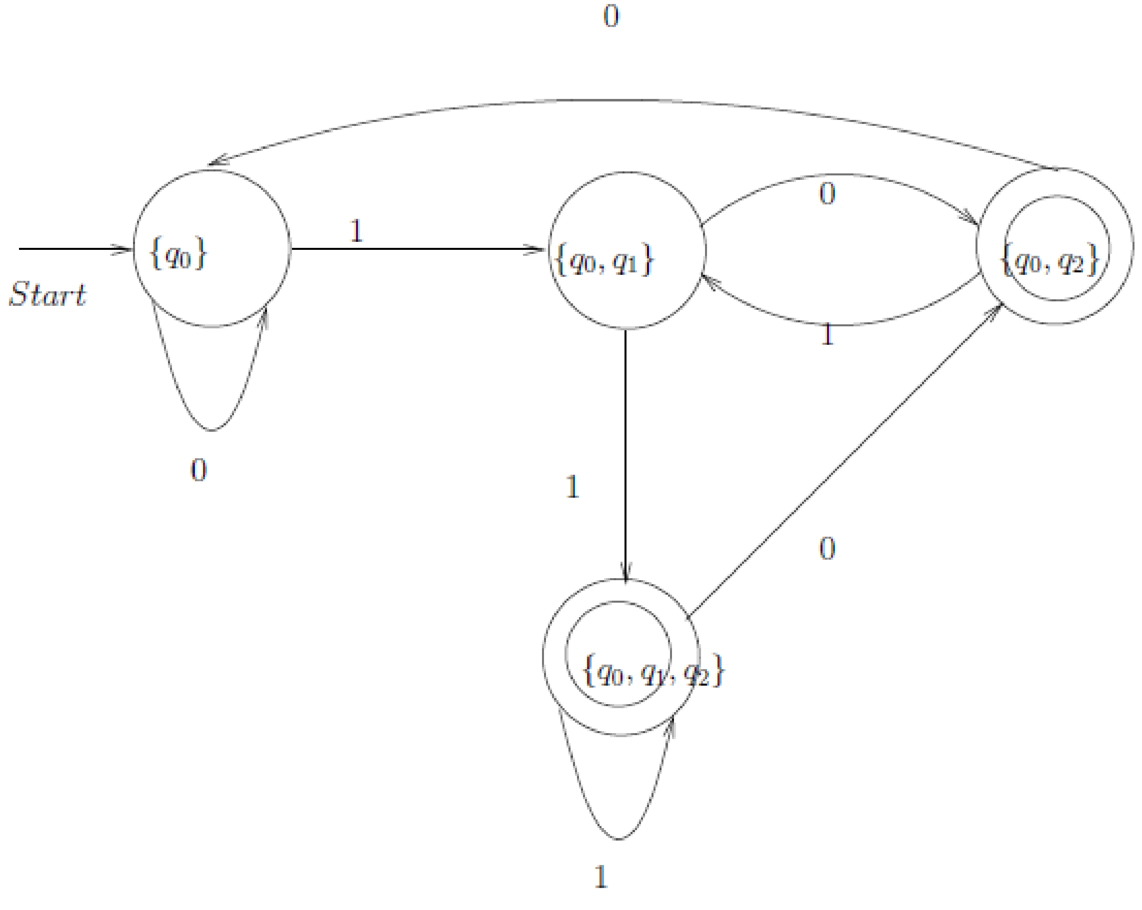


Figure 2.10: A DFA equivalent to the NFA of Fig. 2.9

Transition table

δ_D	0	1
$\{q_0\}$	$\{q_0\}$	$\{q_0, q_1\}$
$\{q_0, q_1\}$	$\{q_0, q_2\}$	$\{q_0, q_1, q_2\}$
$*\{q_0, q_2\}$	$\{q_0\}$	$\{q_0, q_1\}$
$*\{q_0, q_1, q_2\}$	$\{q_0, q_2\}$	$\{q_0, q_1, q_2\}$

able 2.3: Transition table for the DFA of Fig.2.10

Proof sketch (1)

- Let L_1 be the language accepted by the given NFA
- Let L_2 be the language accepted by the constructed DFA
- We show $L_1 = L_2$
- For this we show that $\hat{\delta}_D(\{q_0\},w)$ is an accepting state iff $\hat{\delta}_N(q_0,w)$ contains an accepting state of the given NFA.

Proof sketch (2)

- We establish the stronger fact

$$\hat{\delta}_D(\{q_0\}, w) = \hat{\delta}_N(q_0, w) \text{ for an arbitrary string } w \text{ in } \Sigma^*$$

Proof sketch (3)

- Induction on |w|
- Basis step: w = ε

$$\hat{\delta}_D(\{q_0\},\varepsilon) = \hat{\delta}_N(q_0,\varepsilon) = \{q_0\}$$

- Inductive hypothesis

$$\hat{\delta}_D(\{q_0\},w) = \hat{\delta}_N(q_0,w) = S$$

for some |w| ≥ 0

Formal proof (4)

For a string *wa* :

$$\begin{aligned}\hat{\delta}_D(\{q_0\}, wa) &= \delta_D(\hat{\delta}_D(\{q_0\}, w), a) \text{ (definition of } \hat{\delta}_D) \\ &= \delta_D(\hat{\delta}_N(q_0, w), a) \text{ (inductive hypothesis)} \\ &= \cup \delta_N(q, a), q \in S \text{ (definition of } \delta_D) \\ &= \hat{\delta}_N(q_0, wa) \text{ (by definition of } \hat{\delta}_N)\end{aligned}$$

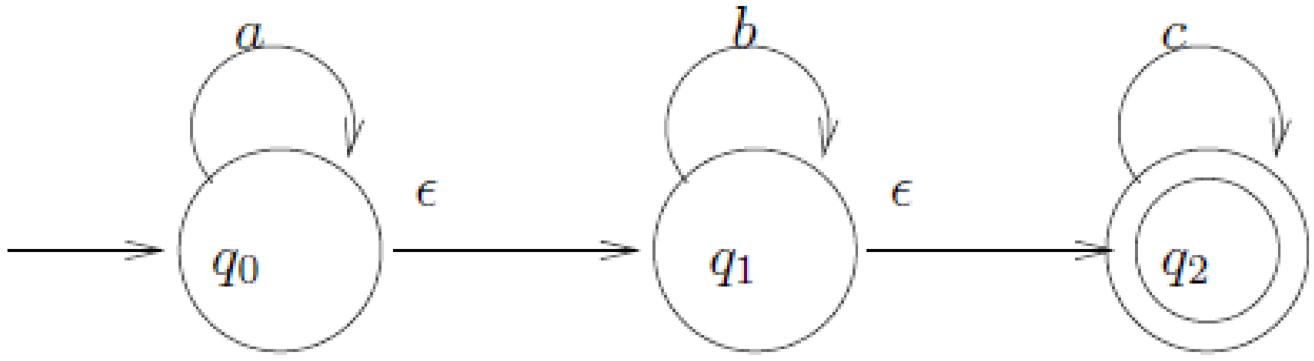
Formal proof (5)

- Class of languages accepted by NFAs is contained in the class of languages accepted by DFAs
- Conversely, as every DFA is trivially an NFA the class of languages accepted by DFAs is contained in the class of languages accepted by NFAs

ε-NFA

- Allow transitions on ε
- Include ε in the alphabet ∑

Example



A new definition

- The ε-closure of a state *q*, *ECLOSE(q)*, is the set of all states reachable from *q* by a finite number of transitions
- Thus, $ECLOSE(q_0) = \{q_0, q_1, q_2\}$
- For a set of states *S*:

$$ECLOSE(S) = \bigcup_q ECLOSE(q),\ q \varepsilon S$$

Extended transition function

$$\hat{\delta}_{NE}(q,\varepsilon) = ECLOSE(\{q\})$$

$$\hat{\delta}_{NE}(q,wa) = ECLOSE(\bigcup_r \delta_{NE}(r,a)), \text{ where } r\,\varepsilon\,\hat{\delta}_{NE}(q,w)$$

ε-NFA to DFA

- Same as the reduction: NFA to DFA
- With one additional step:

$$\delta_D(S,a) = ECLOSE\left(\bigcup_r \delta_{NE}(r,a)\right), \text{ where } r \in S$$

Example ε-NFA

$$\delta_D(\{q_0, q_1, q_2\}, a) = \{q_0, q_1, q_2\}$$

$$\delta_D(\{q_0, q_1, q_2\}, b) = \{q_1, q_2\}$$

$$\delta_D(\{q_0, q_1, q_2\}, c) = \{q_2\}$$

$$\delta_D(\{q_1, q_2\}, a) = \Phi \text{ (dead state)}$$

$$\delta_D(\{q_1, q_2\}, b) = \{q_1, q_2\}$$

$$\delta_D(\{q_1, q_2\}, c) = \{q_2\}$$

$$\delta_D(\{q_2\}, a) = \Phi$$

$$\delta_D(\{q_2\}, b) = \Phi$$

$$\delta_D(\{q_2\}, c) = \{q_2\}$$

Equivalence proof for any ε-NFA

- Set q_D = ECLOSE(q_0)
- Prove by induction for an arbitrary string *w* that:

$$\hat{\delta}_D(q_D, w) = \hat{\delta}_{NE}(q_0, w) = S$$

Equivalence proof…..

- Base case: $w = \varepsilon$

$$\hat{\delta}_D(q_D, \varepsilon) = \hat{\delta}_{NE}(q_0, \varepsilon) = ECLOSE(q_0)$$

Equivalence proof.....

- Assume inductively

$$\hat{\delta}_D(q_D, w) = \hat{\delta}_{NE}(q_0, w) = S \text{ for some } |w| \geq 0$$

Equivalence proof.....

- Then

$$\begin{aligned}\hat{\delta}_D(q_D, wa) &= \delta_D(\hat{\delta}_D(q_D, w), a) \text{ (definition of } \hat{\delta}_D()) \\ &= \delta_D(\hat{\delta}_{NE}(q_0, w), a) \text{ (inductive hypothesis)} \\ &= \text{ECLOSE}(\cup_r \delta_{NE}(r, a)),\ r \in \hat{\delta}_{NE}(q_0, w) \text{ (definition of } \delta_D()) \\ &= \hat{\delta}_{NE}(q_0, wa) \text{ (definition of } \hat{\delta}_{NE}())\end{aligned}$$

Thus a string is accepted by the constructed DFA iff it is accepted by the ε-NFA

What we have shown…

- The class of languages accepted by DFAs is in the class of languages accepted by ε-NFAs
- Now to show the converse ….

Regular expressions

- Definition
  - **a**, **ε**, **ϕ** are regular expressions (a in ∑)
  - If **r** and **s** are regular expressions then so are **r+s**, **r*s** (usually written **rs**) and **r***
  - $\mathbf{r}^* = \boldsymbol{\varepsilon} + \mathbf{r} + \mathbf{r}^2 + \mathbf{r}^3 + \ldots$

Languages and regular expressions

- There is a language corresponding to every regular expression
- {a}, {ε}, ϕ for **a**, **ε** and **ϕ** respectively
- If L(**r**) and L(**s**) are the languages corresponding to **r** and **s** then L(**r**) UL(**s**), L(**r**)L(**s**) and (L(**r**))* are languages corresponding to the regular expressions **r+s**, **rs** and **r***

Constructing re's …

- Construct a regular expression for the set of all strings over ∑ = {0,1}
- **ε** is the re for the empty string ε
- (**0**+**1**) for the strings 0 and 1 of length 1
- (**0**+**1**)(**0**+**1**) for all the strings of length 2
- ….
- Thus: $(\mathbf{0}+\mathbf{1})^* = \boldsymbol{\varepsilon} + (\mathbf{0}+\mathbf{1}) + (\mathbf{0}+\mathbf{1})^2 + (\mathbf{0}+\mathbf{1})^3 + \ldots.$

More examples (1)

- Construct a regular expression from the following description: the language consisting of all strings of 0’s and 1’s whose tenth symbol from the right end is 1.

Solution

- $(0+1)^*1(0+1)^9$

More examples (2)

- Construct a regular expression from the following description: the language consisting of all strings of 0’s and 1’s whose number of 0’s is divisible by 5.

Solution

- Think backwards!
- Remove all 1’s and block the 0’s in groups of 5
- Reinsert the 1’s
- The internal structure of a block of string with exactly five 0’s is: 01*01*01*01*0
- These blocks are separated by zero or more 1’s
- Thus the r.e.: (1 + 01*01*01*01*0)*

The other way round

- Give an English language description of the language corresponding to the following regular expression: $(0 + 10)^*1^*$

Solution

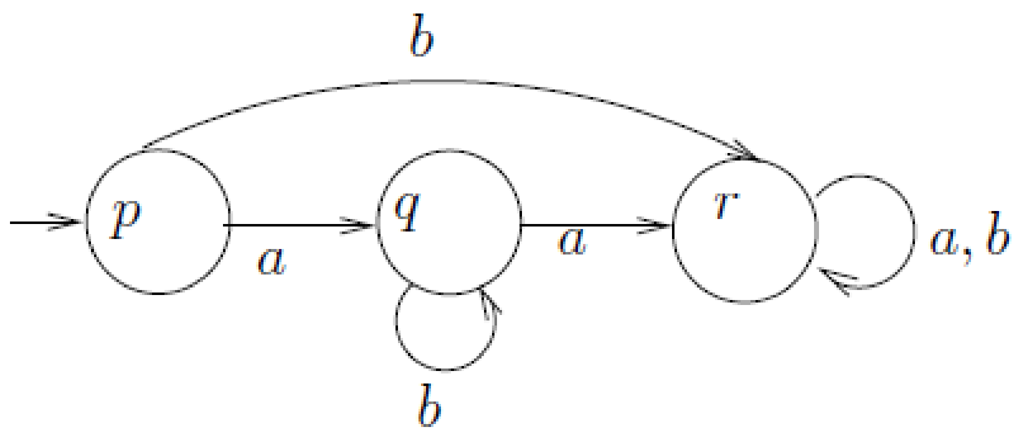
- Any string s in L((0 + 10)*1*) can be written as αβ, where
  - α = ε or α ε L((0 + 10)*) and
  - β = ε or β ε L(1*).
- When α = ε or β = ε, s cannot have 110 as a substring;
- Otherwise, consider
  - a substring $a_1a_2a_3$ of length 3 that spans both α and β.
  - If a_1a_2 is a suffix of α, a_1a_2 = 10|00; and a_3 = 1.
  - Hence $a_1a_2a_3 \neq 110$.
  - If a_2a_3 is a prefix of , then a_2a_3 = 11 and a_1 = 0.
  - Hence $a_1a_2a_3 \neq 110$.
  - This covers all the cases and hence the claim.

Regular expressions from a DFA

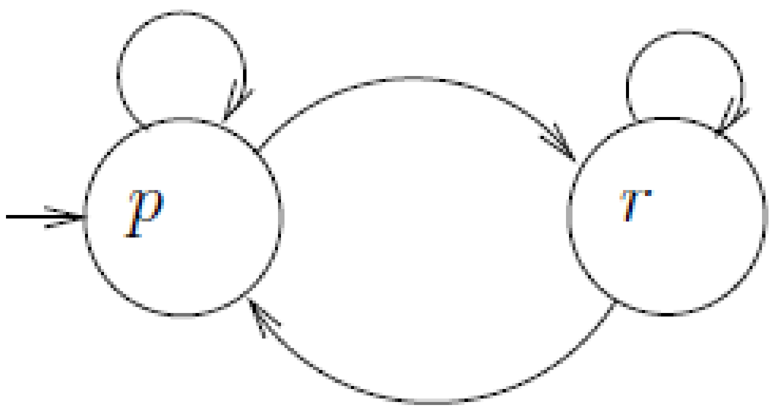
- Idea:
  - find a regular expression label for all paths from the start state to an accepting state, by eliminating all other states except these two
  - If the start state is also accepting, we determine this regular expression by eliminating all states except the start state
  - The regular expression corresponding to the language accepted by the DFA is the “sum” of all the regular expressions so obtained.

State Elimination

- If we remove the state q, the transition from the state p to the state r has to be labelled by the regular expression **b + ab*a**

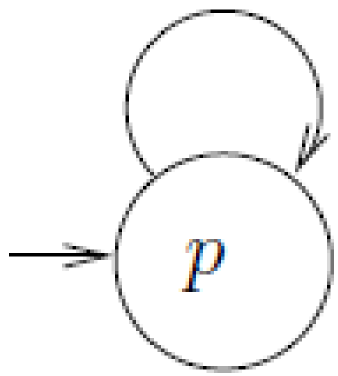


Case 1



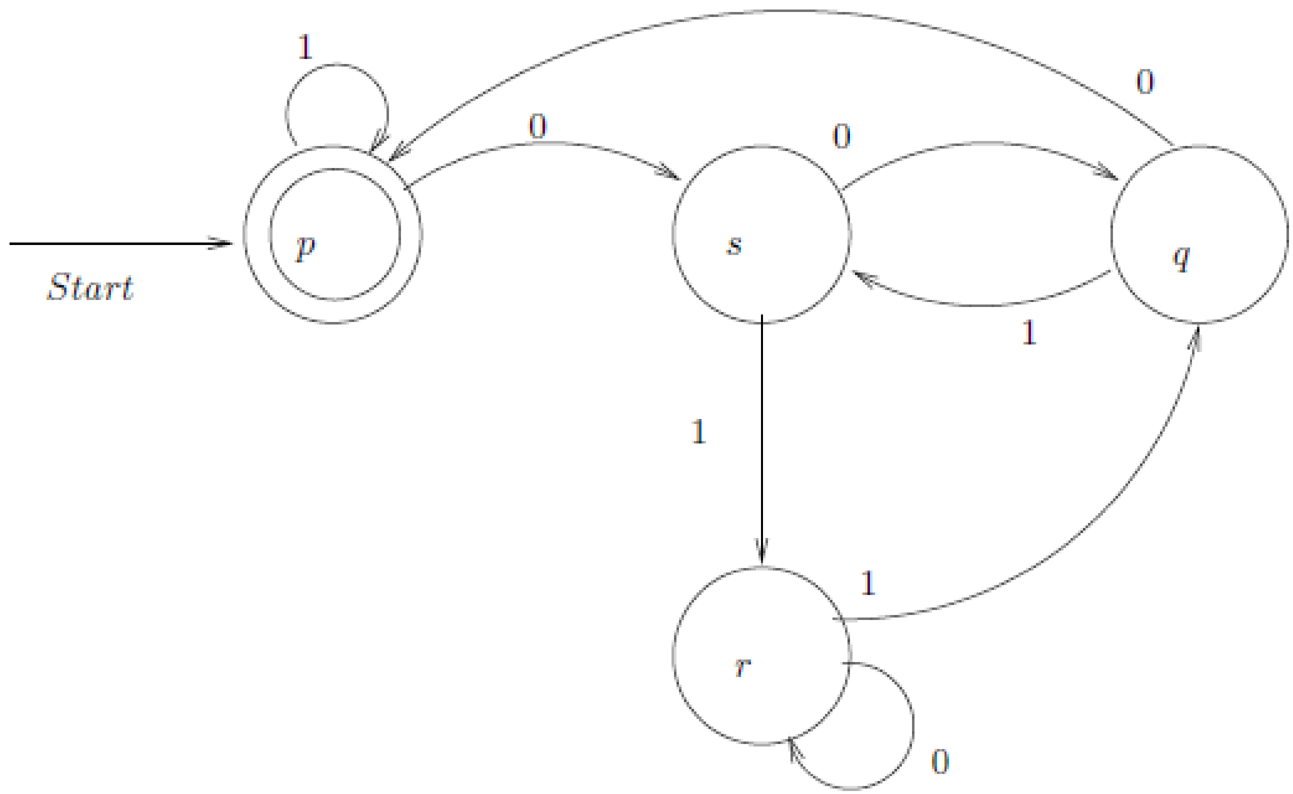
: Reduced DFA to start state p and final state r

Case 2

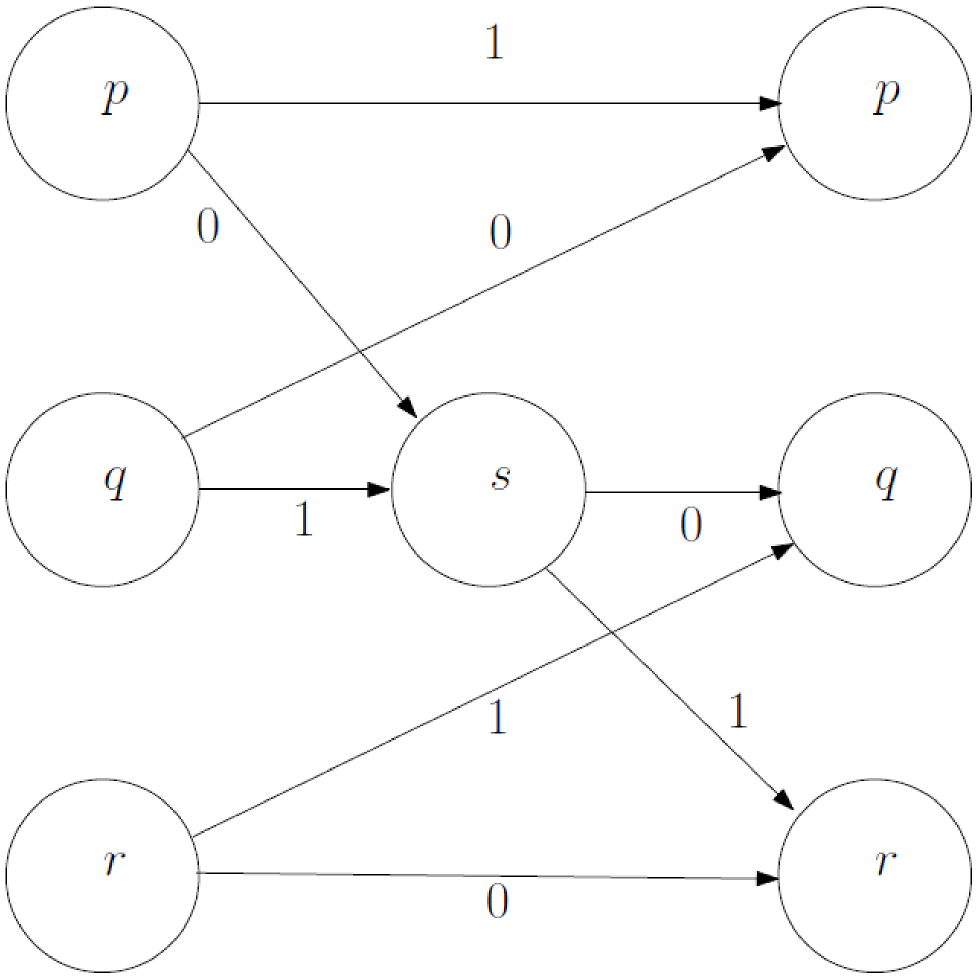


: Reduced DFA to final state p

Example DFA for state elimination



Removing state s



Reduced DFA

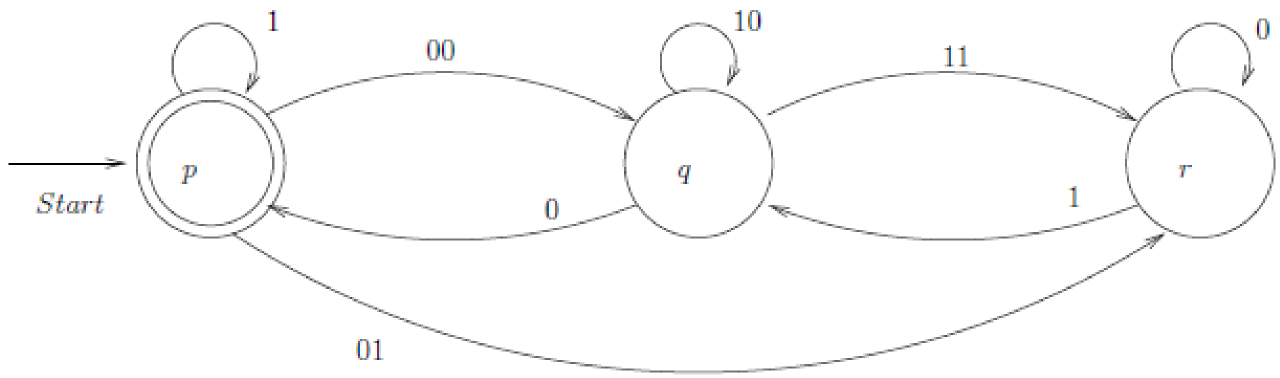
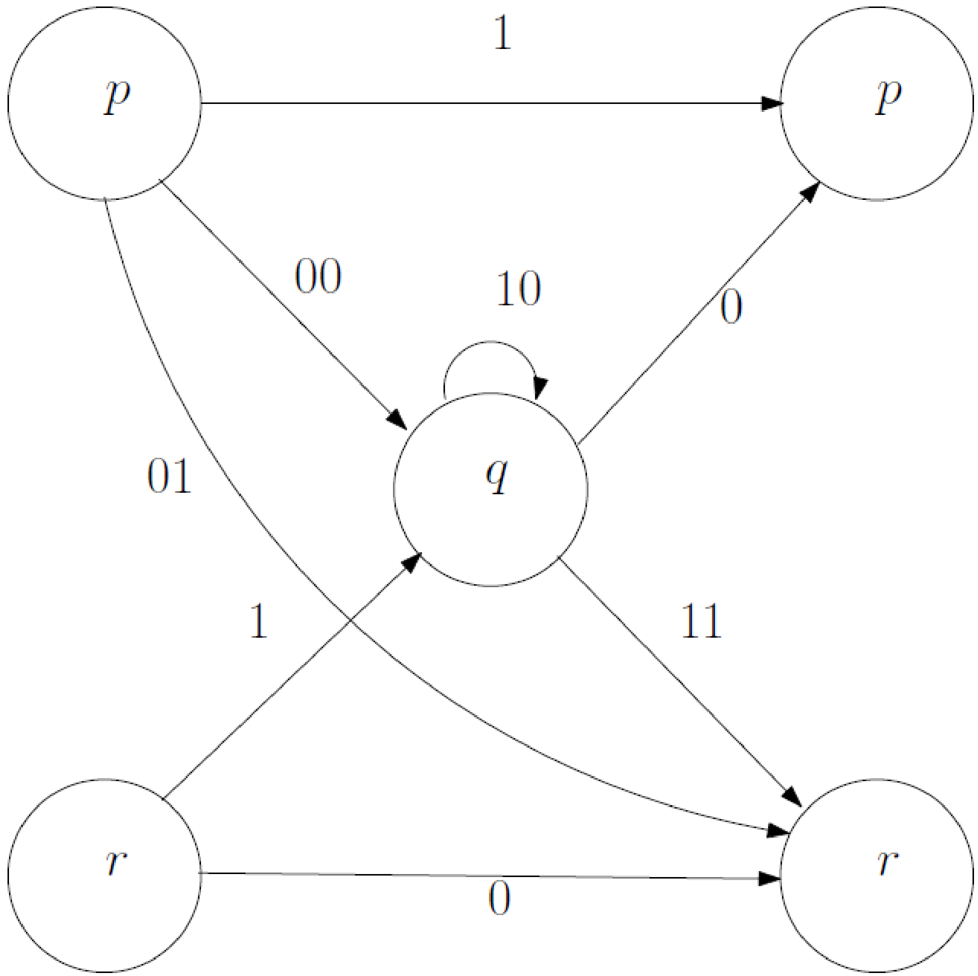
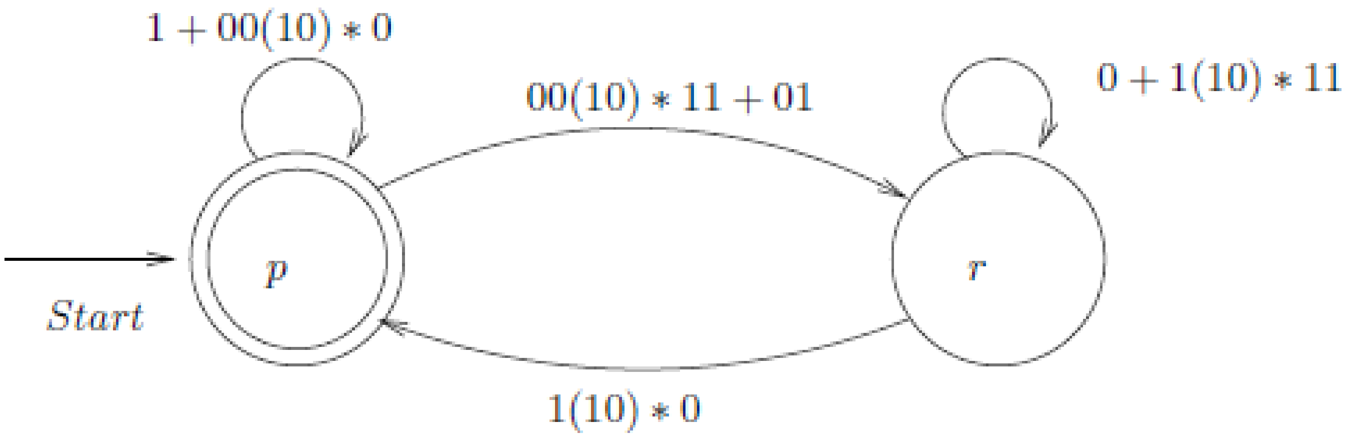


Figure 2.16: Reduced DFA on elimination of state s

Removing state q

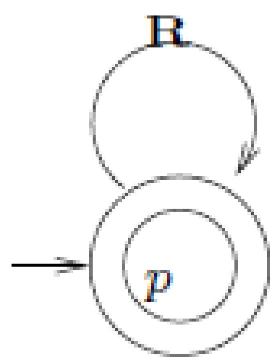


Reduced DFA



Reduced DFA on elimination of state q

Removing state r



: Reduced DFA on elimination of state r

$$\mathbf{R} = (1 + 00(10)^*0 + (00(10)^*11 + 01)(0 + 1(10)^*11)^*1(10)^*0)^*$$

Regular expressions to ε-NFAs

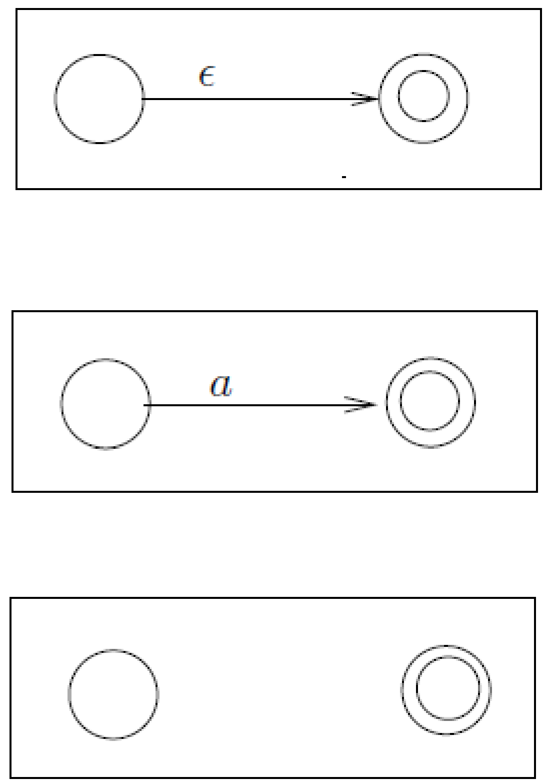
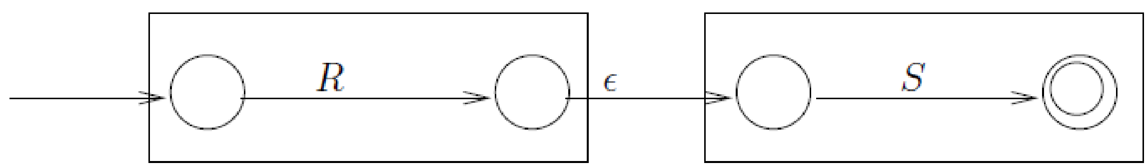


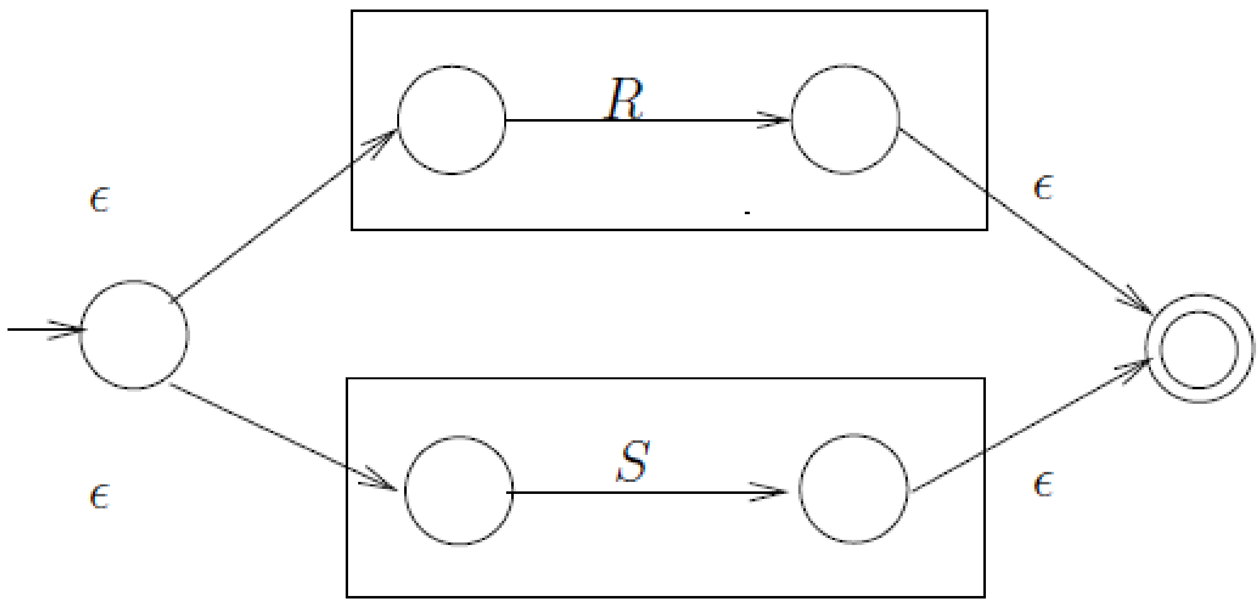
Figure 2.19: ε-NFAs for the base expressions

Regular expressions to ε-NFAs



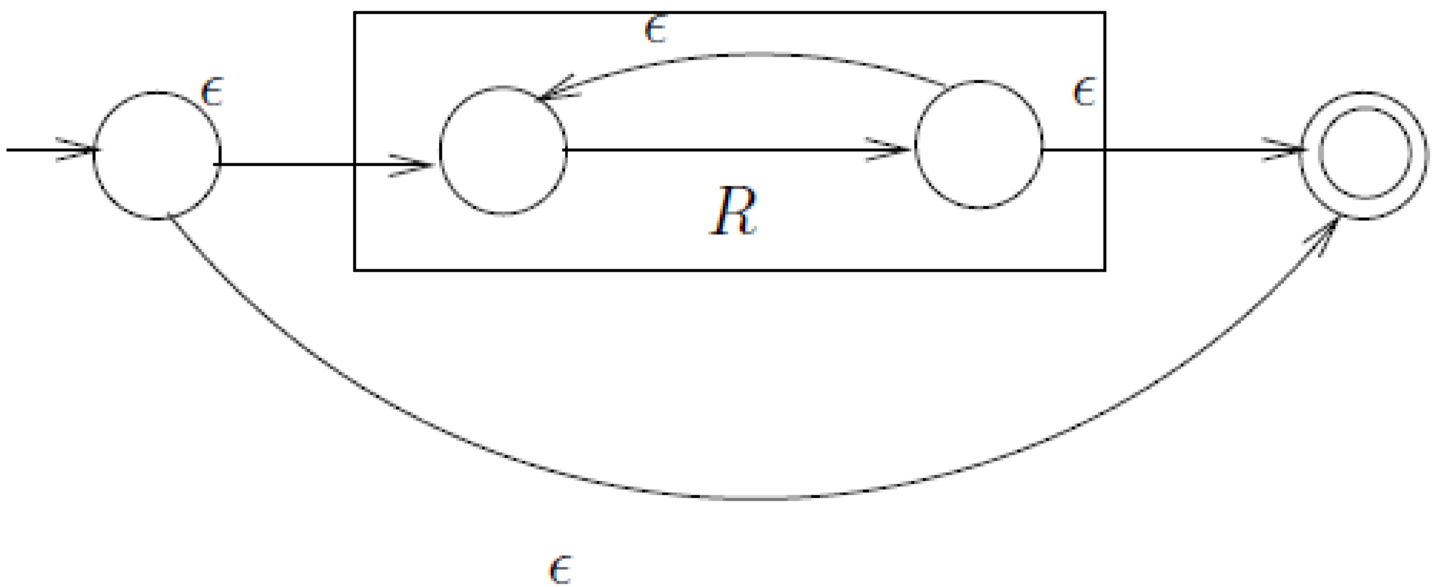
ε-NFA for RS

Regular expressions to ε-NFAs



ε-NFA for R+S

Regular expressions to ε-NFAs

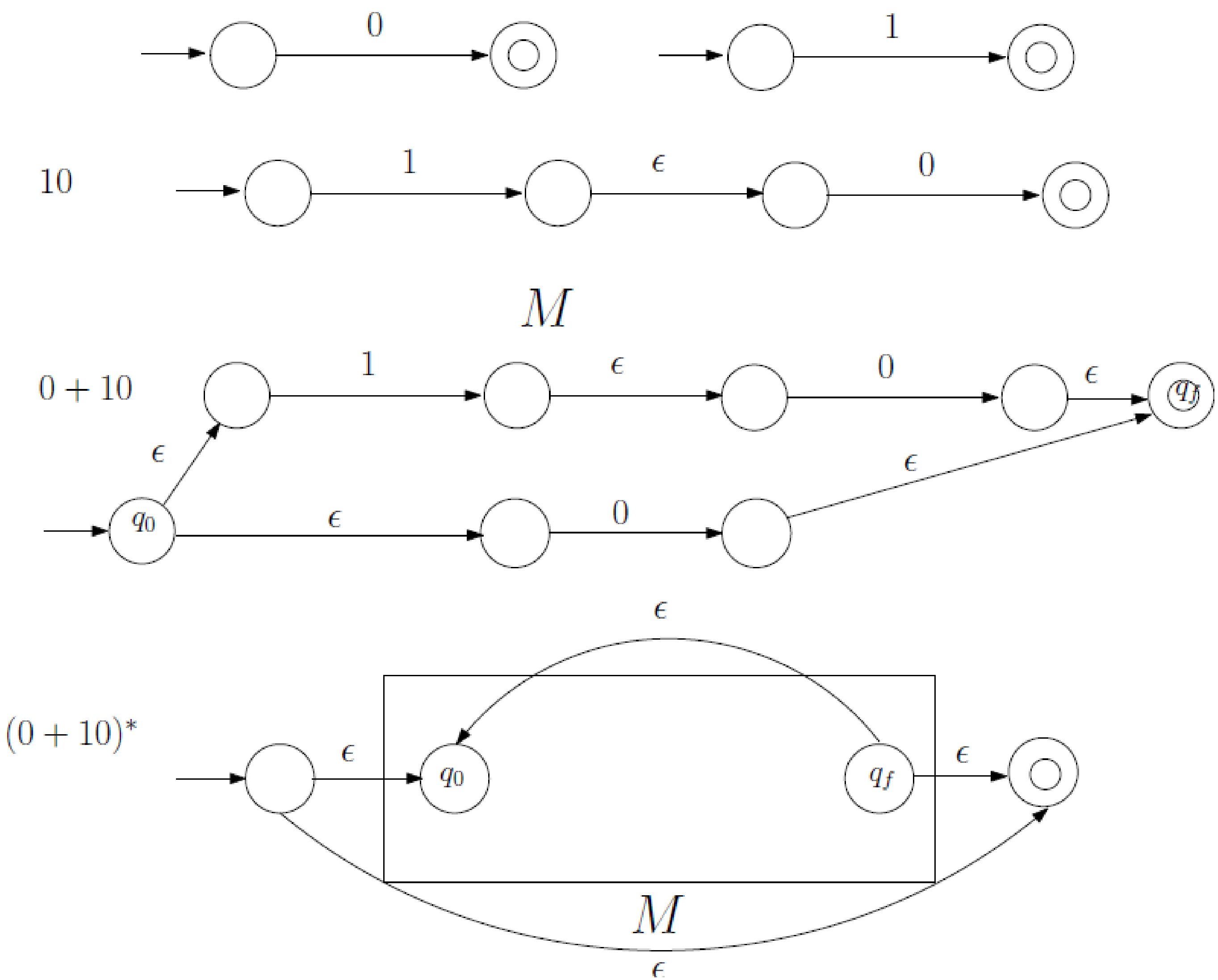


ε-NFA for R*

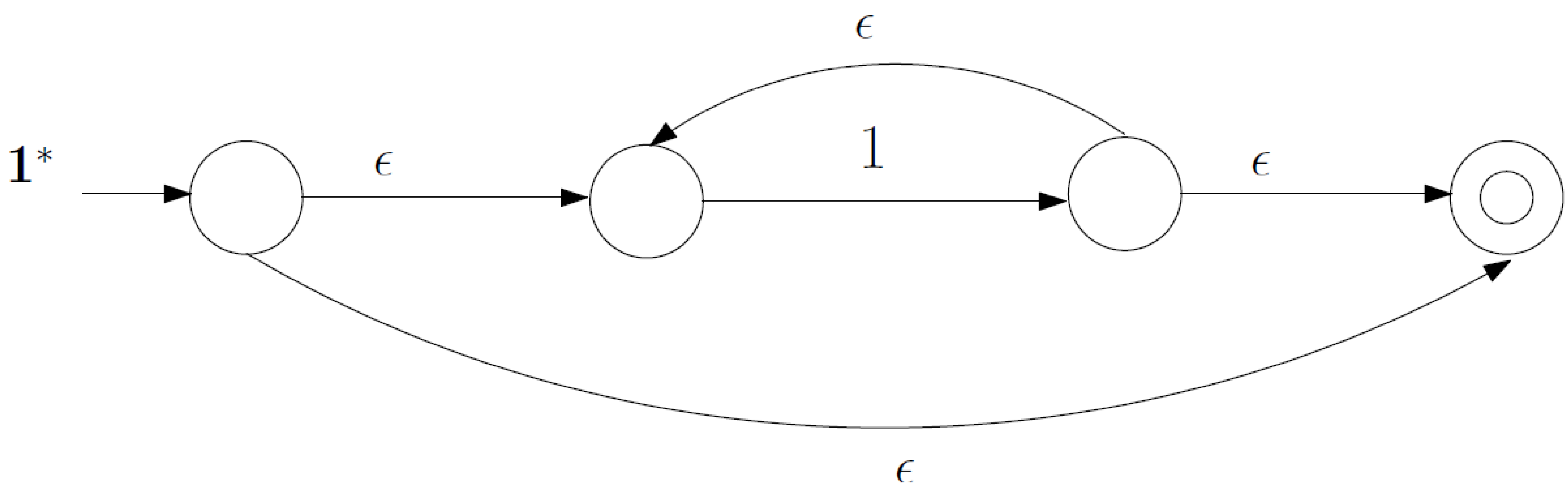
Example

- Construct ε-NFA that accepts L((0 +10)*1*))

ε-NFA for L((0 +10)*1*))



ε-NFA for L((0 +10)*1*))



Summing up (1)

- With this we have completed the sequence of reductions

$$NFA \rightarrow DFA \rightarrow RE \rightarrow \epsilon - NFA \rightarrow DFA \rightarrow NFA,$$

Summing up (2)

- This establishes the equivalence of all the computational models in that they all recognize the class of regular languages