NFA to DFA Conversion (Subset Construction Method)

Course Code: CSC3220 Course Title: Compiler Design

Dept. of Computer Science
Faculty of Science and Technology

Lecturer No:	8	**Week No:**	8	**Semester:**	
Lecturer:					

Lecture Outline

1. NFA TO DFA (Subset Construction Method)
2. Subset Construction Algorithm
3. DFA Designing
4. Example
5. Exercise
6. References

Objective and Outcome

Objective:

- To explain the subset construction algorithm/method for converting a Non deterministic machine to deterministic machine.
- Provide necessary example and explanation of NFA to DFA conversion method using subset construction method.
- To explain and practice Deterministic Finite Automata (DFA) Machine Design for a given Grammar.

Outcome:

- After this lecture the students will be capable of demonstrating the subset construction algorithm
- After this lecture the student will be able to convert an NFA to relevant DFA by following subset construction method.
- After this class student will be able to design and demonstrate DFA construction from a given Grammar.

NFA to DFA Conversion

Subset Construction Algorithm

Input: An NFA N

Output: A DFA D accepting the same language

Method: Constructs a transition table Dtran for D. Each DFA state is a set of NFA states and construct Dtran so that D will simulate “in parallel” all possible moves N can make on a given input string

OPERATION	DESCRIPTION
ϵ-*closure*(s)	Set of NFA states reachable from NFA state s on ϵ-transitions alone.
ϵ-*closure*(T)	Set of NFA states reachable from some NFA state s in T on ϵ-transitions alone.
move(T, a)	Set of NFA states to which there is a transition on input symbol a from some NFA state s in T.

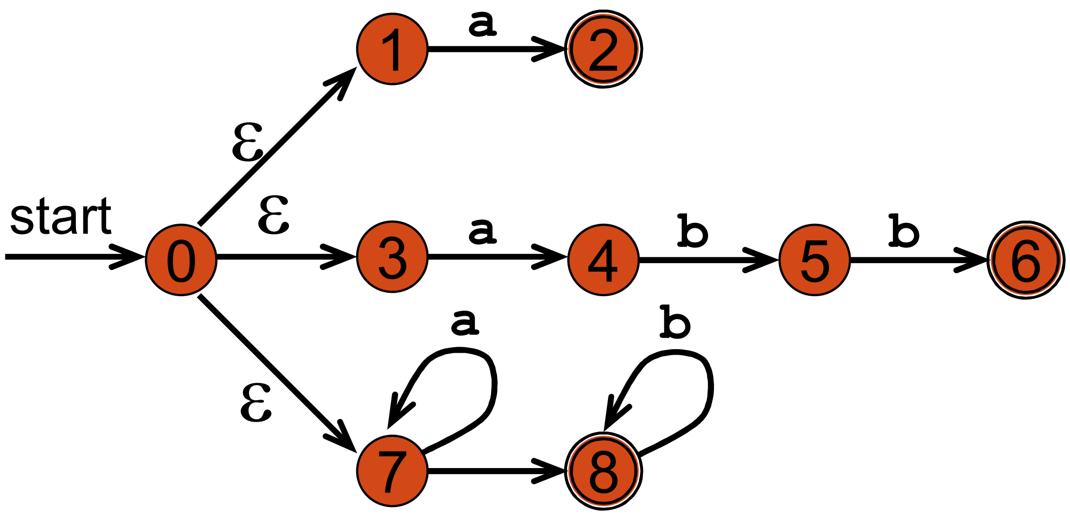
NFA to DFA Conversion

Subset Construction Algorithm

```
initially, ε-closure(s0) is the only state in Dstates and it is unmarked;
while there is an unmarked state T in Dstates do begin
        mark T;
        for each input symbol a do begin
                U := ε-closure(move(T, a));
                if U is not in Dstates then
                        add U as an unmarked state to Dstates;
                Dtran[T, a] := U
        end
end
```

NFA to DFA Conversion

ε-closure and *move* Examples



ε-closure({0}) = {0,1,3,7}
move({0,1,3,7},**a**) = {2,4,7}
ε-closure({2,4,7}) = {2,4,7}
move({2,4,7},**a**) = {7}
ε-closure({7}) = {7}
move({7},**b**) = {8}
ε-closure({8}) = {8}
move({8},**a**) = $\varnothing$

Alphabet / Symbol = {a, b}

Subset Construction Algorithm

Subset Construction Algorithm

The *subset construction algorithm* converts an NFA into a DFA using:

$$\varepsilon\text{-}closure(s) = \{s\} \cup \{t \mid s \to_\varepsilon \ldots \to_\varepsilon t\}$$
$$\varepsilon\text{-}closure(T) = \cup_{s \in T}\, \varepsilon\text{-}closure(s)$$
$$move(T,a) = \{t \mid s \to_a t \text{ and } s \in T\}$$

The algorithm produces:

- D_{states} is the set of states of the new DFA consisting of sets of states of the NFA
- D_{tran} is the transition table of the new DFA

Subset Construction Algorithm

Algorithm Explained

1. Create the start state of the DFA by taking the ε-closure of the start state of the NFA
2. Perform the following for the DFA state:
   - Apply move to the newly-created state and the input symbol; this will return a set of states.
   - Apply the ε-closure to this set of states, possibly resulting in a new set.

     This set of NFA states will be a single state in the DFA.
3. Each time we generate a new DFA state, we must apply step 2 to it. The process is complete when applying step 2 does not yield any new states.
4. The finish states of the DFA are those which contain any of the finish states of the NFA

Subset Construction Algorithm

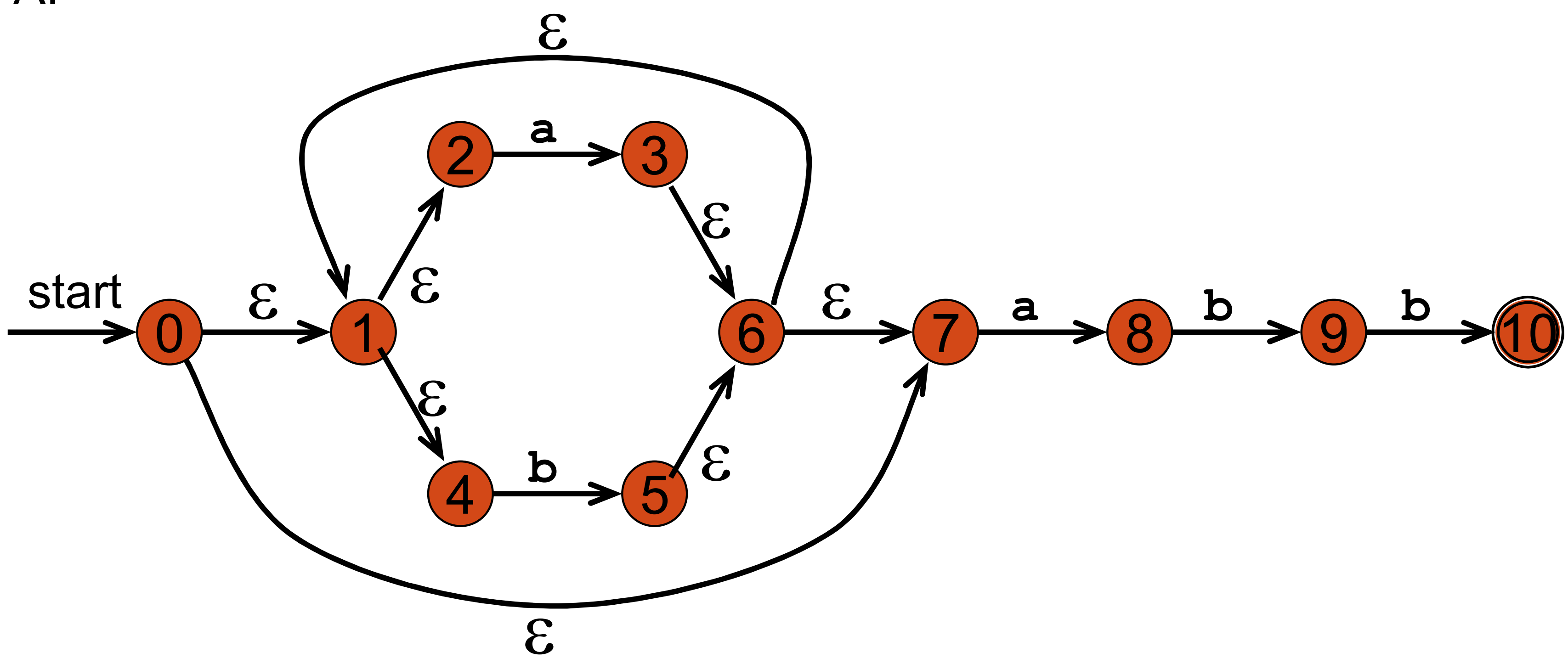
Algorithm with while Loop

```
fun nfa2dfa start edges =
 let val chars = nodup(sigma edges)
     val s0 = eclosure edges [start]
     val worklist = ref [s0]
     val work = ref []
     val old = ref []
     val newEdges = ref []
 in while (not (null (!worklist))) do
     ( work := hd(!worklist)
     ; old := (!work) :: (!old)
     ; worklist := tl(!worklist)
     ; let fun nextOn c = (Char.toString c
                      ,eclosure edges (nodesOnFromMany (Char c) (!work) edges))
           val possible = map nextOn chars
           fun add ((c,[])::xs) es = add xs es
             | add ((c,ss)::xs) es = add xs ((!work,c,ss)::es)
             | add [] es = es
           fun ok [] = false
             | ok xs = not(exists (fn ys => xs=ys) (!old)) andalso
                    not(exists (fn ys => xs=ys) (!worklist))
           val new = filter ok (map snd possible)
       in worklist := new @ (!worklist);
          newEdges := add possible (!newEdges)
       end
     );
    (s0,!old,!newEdges)
 end;
```

NFA to DFA Conversion

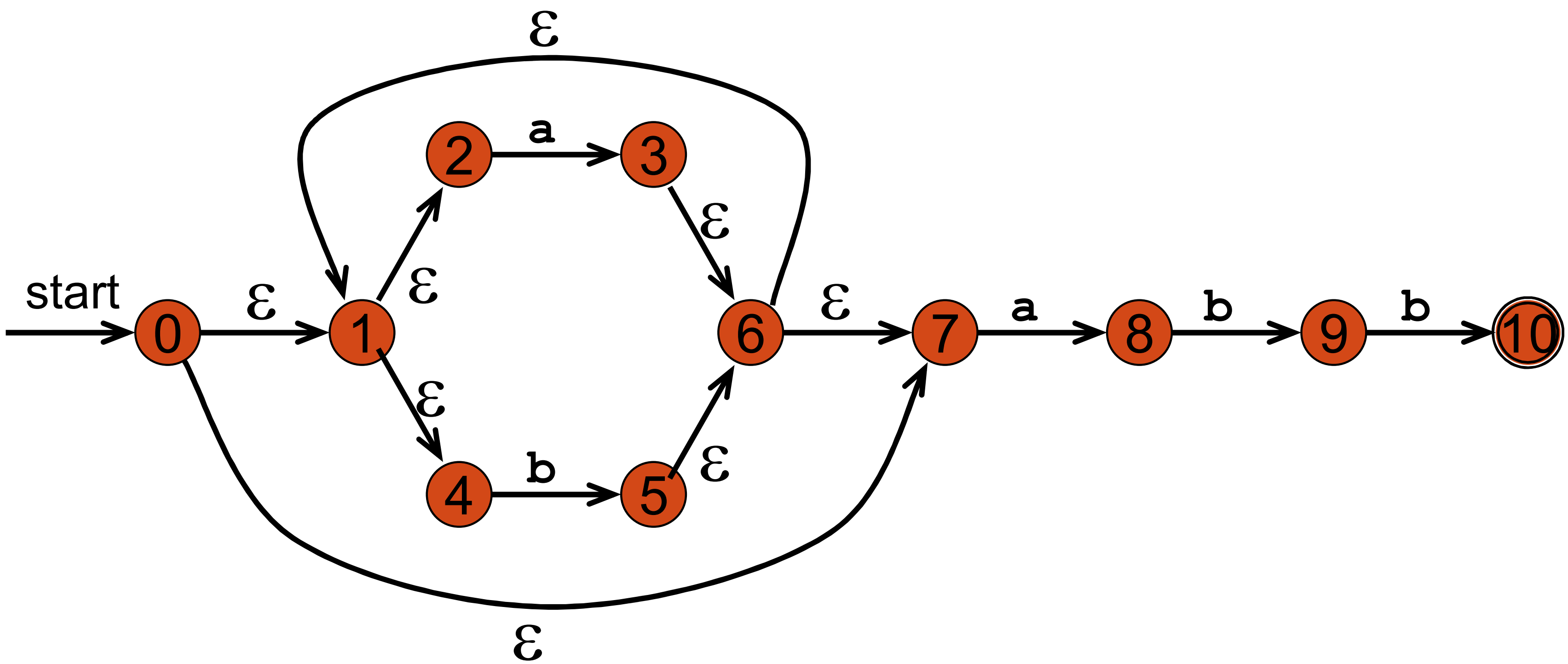
Subset Construction Method (Example-1)

NFA:



Regular Expression: (a | b)* abb

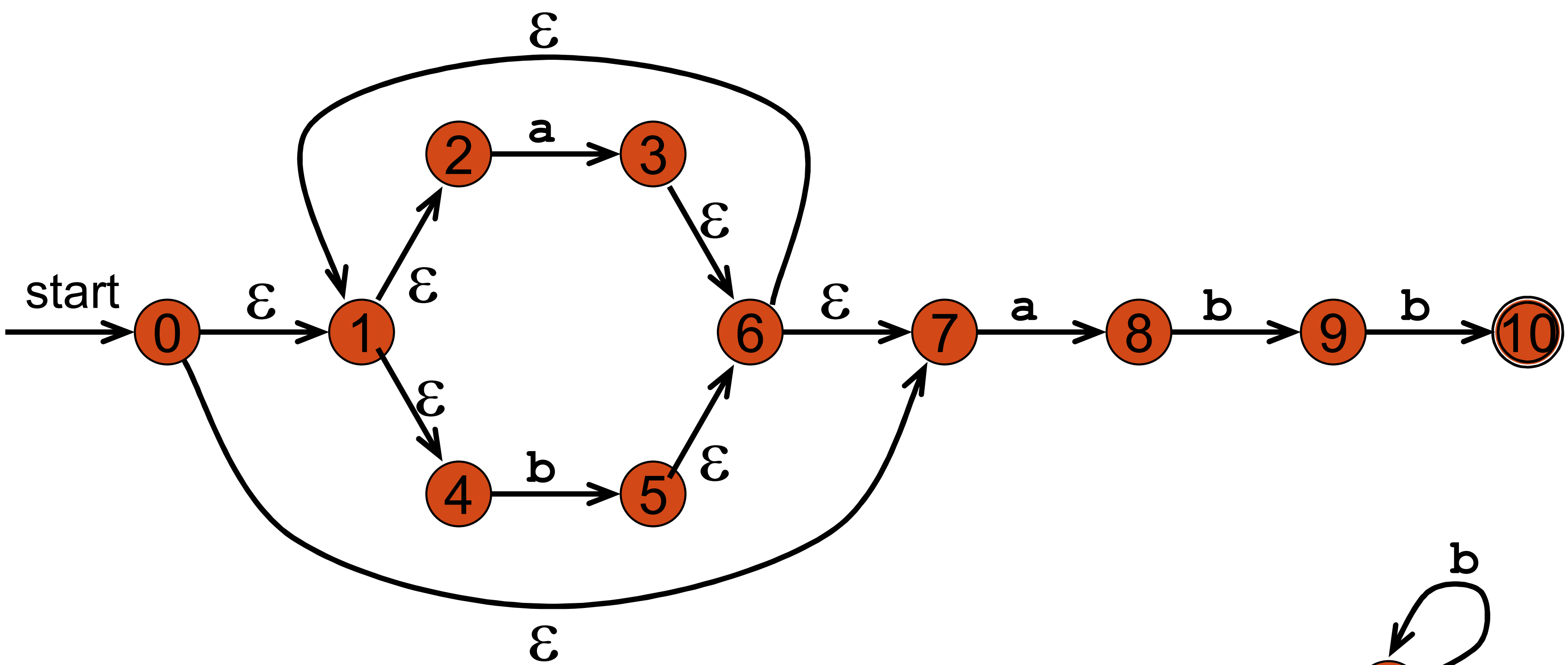
Subset Construction Method (Example-1)

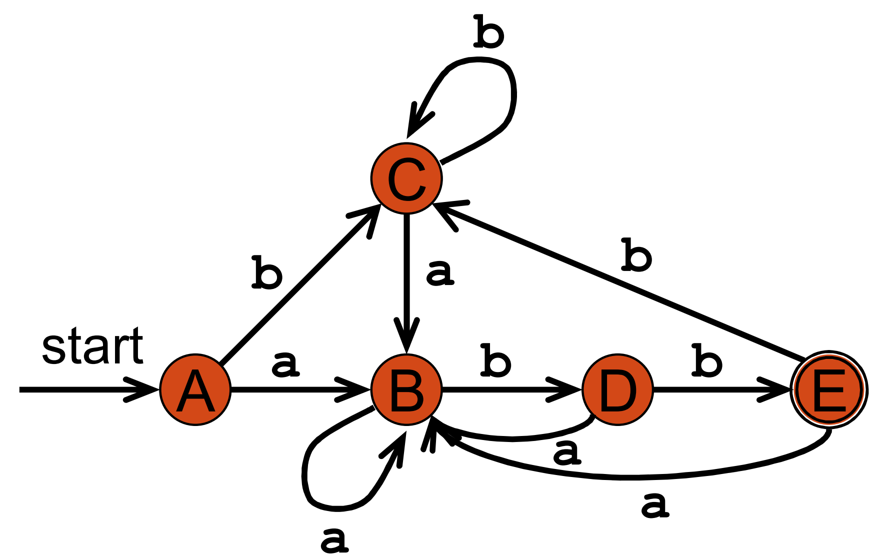


DFA State	E-closure of	E-closure outcome states
A	E-closure ({0})	0,1,2,4,7
B	E-closure ({3,8})	1,2,3,4,6,7,8
C	E-closure ({5})	1,2,4,5,6,7
D	E-closure({5,9})	1,2,4,5,6,7,9
E	E-closure({5,10})	1,2,4,5,6,7,10

NFA States	DFA State	a	b
0,1,2,4,7	A	B	C
1,2,3,4,6,7,8	B	B	D
1,2,4,5,6,7	C	B	C
1,2,4,5,6,7,9	D	B	E
1,2,4,5,6,7,10	E	B	C

Subset Construction Method (Example-1 Cont.)



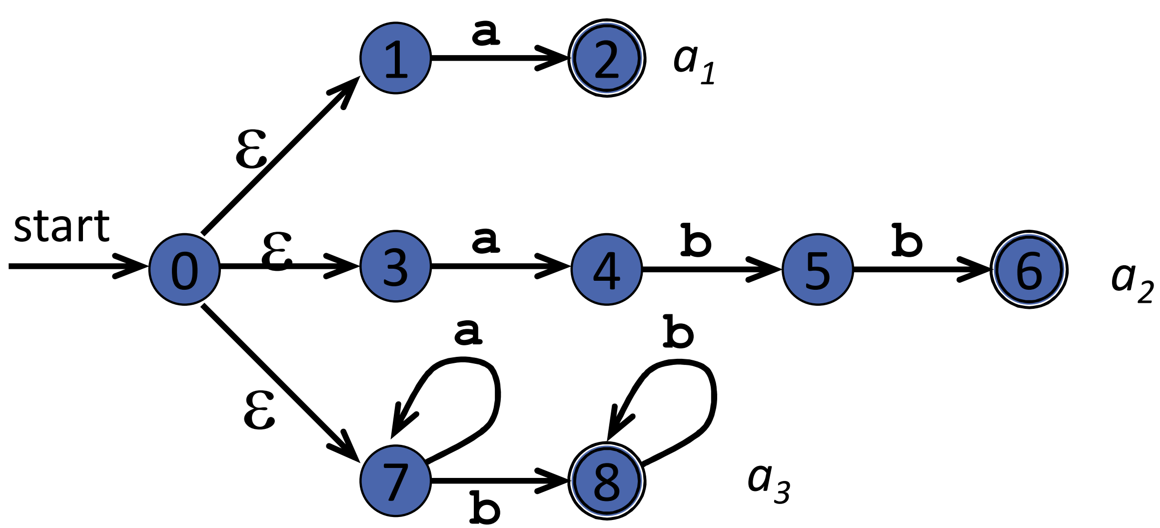


NFA State	DFA State	a	b
0,1,2,4,7	A	B	C
1,2,3,4,6,7,8	B	B	D
1,2,4,5,6,7	C	B	C
1,2,4,5,6,7,9	D	B	E
1,2,4,5,6,7,10	E	B	C

NFA to DFA Conversion

Subset Construction Method (Exercise 1)

NFA

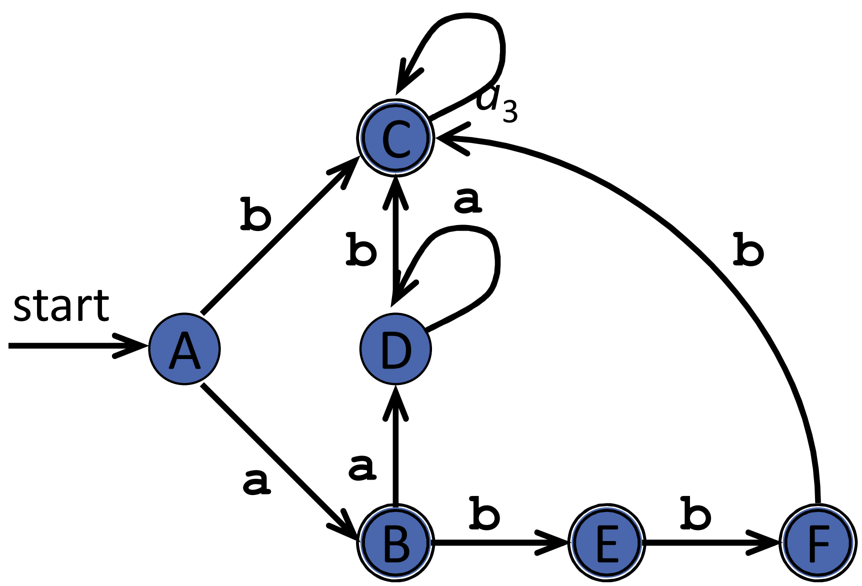


Converted DFA in the next Slide

NFA to DFA Conversion

Subset Construction Method (Exercise 1)

DFA



Dstates

A = {0,1,3,7}

B = {2,4,7}

C = {8}

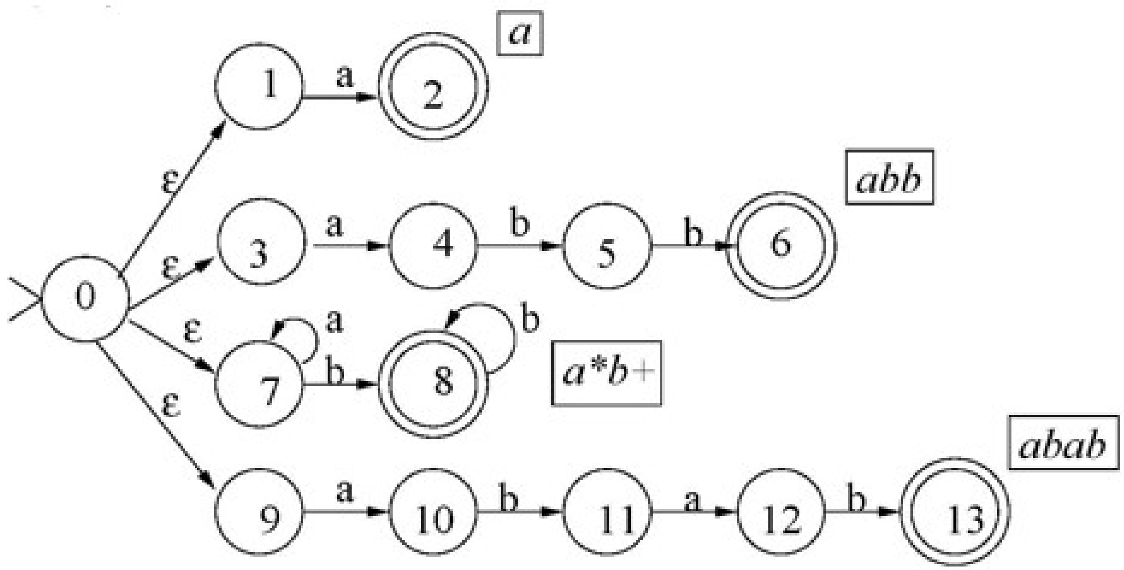
D = {7}

E = {5,8}

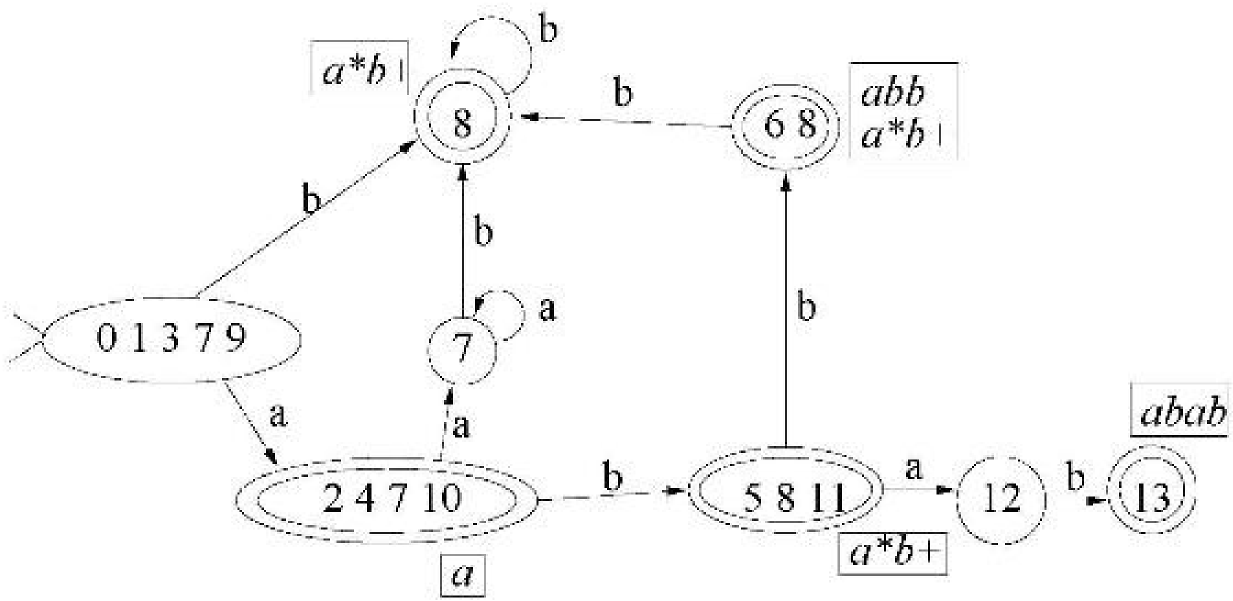
F = {6,8}

NFA to DFA / Subset Construction Method (Exercise 2)

NFA



DFA
Hints



Deterministic Finite Machine

DFA DESIGN

- A finite automaton is a 5-tuple (Q, Σ, δ, q_0, F), where
  - Q is a finite set called the ***states***,
  - Σ is a finite set called the ***alphabet***,
  - $\delta : Q \times \Sigma \rightarrow Q$ is the ***transition function***,
  - $q_0 \in Q$ is the ***start state***,
  - $F \subseteq Q$ is the set of ***accept*** (*final*) ***states***.
- If A is the set of all strings that a machine *M* accepts, we say that *A* is the ***language of machine M*** and write $L(M)=A$, ***M recognizes A*** or ***M accepts A***.

Deterministic Finite Machine

DFA Example 1

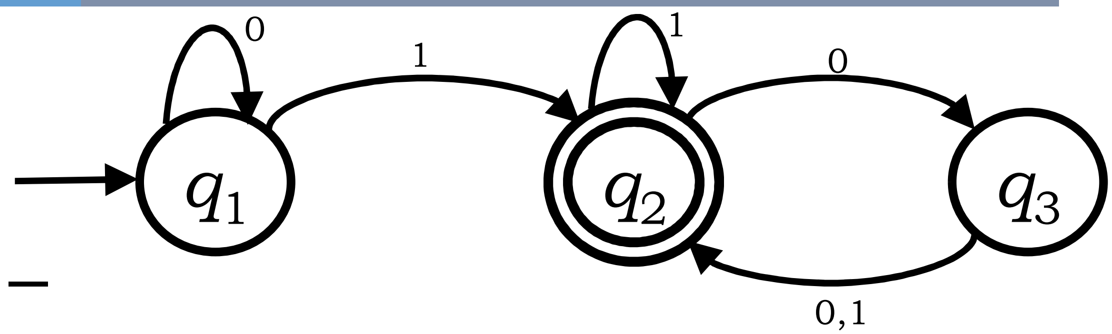


Figure: Finite Automaton M_1

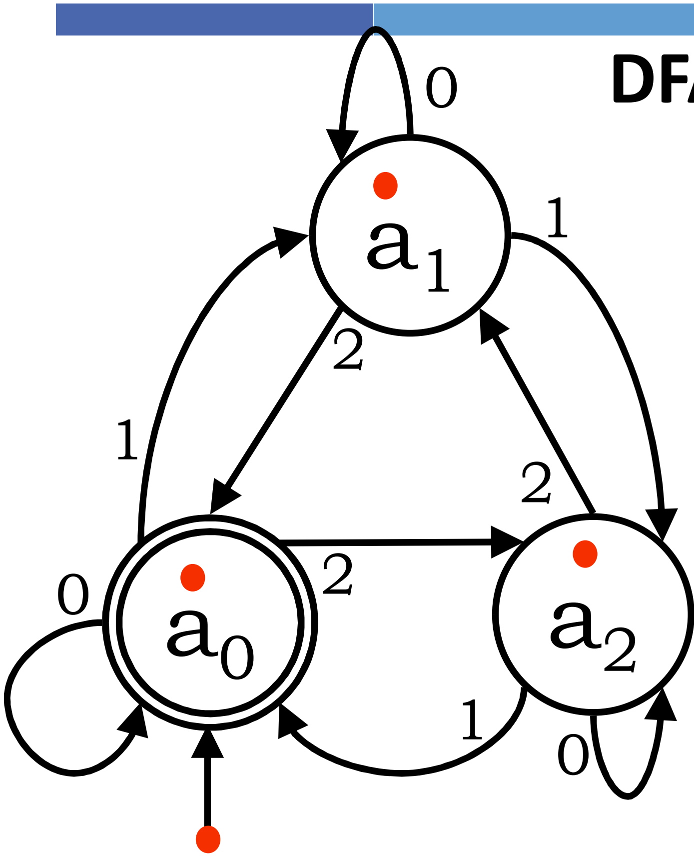
- $M_1 = (Q, \Sigma, \delta, q_0, F)$, where –
  - $Q = \{q_1, q_2, q_3\}$,
  - $\Sigma = \{0, 1\}$,
  - δ *is describe as* –
  - $q_0 = q_1$,
  - $F = \{q_2\}$.

δ	0	1
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_2	q_2

or

$$\delta(q_1,0) = q_1,\ \delta(q_1,1) = q_2,$$
$$\delta(q_2,0) = q_3,\ \delta(q_2,1) = q_2,$$
$$\delta(q_3,0) = q_2,\ \delta(q_3,1) = q_2.$$

DFA Design Example



- Input example: 01120101
- Present State:

a_2

- Input symbol:

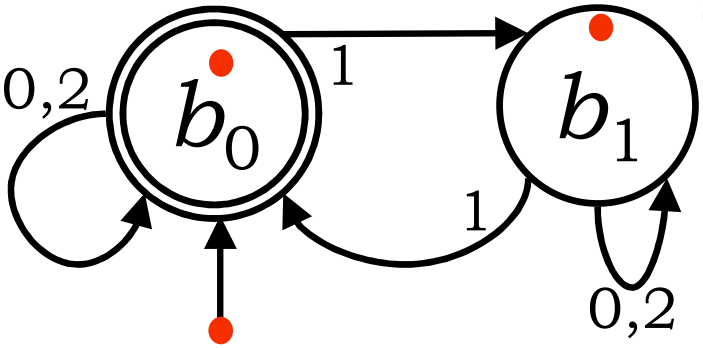
ε

Accepted

- Alphabet Σ={0,1,2}.
- Language A_1 = {w : the sum of all the symbols in w is multiple of 3 }.
  - Can be represented as follows –
    - S= the sum of all the symbols in w.
    - If S modulo 3 = 0 then the sum is multiple of 3.
    - So the sum of all the symbols in w is 0 modulo 3.
    - Here, a_i is modeled as S modulo 3 = i.
- The finite state machine M_1= (Q_1, Σ, δ_1, q_1, F_1), where –
  - $Q_1 = \{a_0, a_1, a_2\}$,
  - $q_1 = a_0$,
  - $F_1 = \{a_0\}$,
  - δ_1

	0	1	2
a_0	a_0	a_1	a_2
a_1	a_1	a_2	a_0
a_2	a_2	a_0	a_1

DFA Design Example



- Alphabet $\Sigma=\{0,1,2\}$.
- Language $A_1 = \{w$: the sum of all the symbols in w is an even number $\}$.
  - Can be represented as follows –
    - S= the sum of all the symbols in w.
    - If S modulo 2 = 0 then the sum is even.
    - Here, b_i is modeled as S modulo 2 = i.
- The finite state machine $M_2= (Q_2, \Sigma, \delta_2, q_2, F_2)$, where –
  - $Q_2 = \{b_0,b_1\}$,
  - $q_2 = b_0$,
  - $F_2 = \{b_0\}$,
  - δ_2

	0	1	2
b_0	b_0	b_1	b_0
b_1	b_1	b_0	b_1

- Input example: 01120101
- Present State:

b_1

- Input symbol:

ε

Accepted

DFA Design Example (Type 1)

The construction of DFA for languages consisting of strings ending with a particular substring.

- Determine the minimum number of states required in the DFA.
  - Calculate the length of substring.
  - All strings ending with ‘n’ length substring will always require minimum (n+1) states in the DFA.
- Draw those states.
- Decide the strings for which DFA will be constructed.
- Construct a DFA for the decided strings
  - While constructing a DFA, Always prefer to use the existing path. Create a new path only when there exists no path to go with.
- Send all the left possible combinations to the starting state.
- Do not send the left possible combinations over the dead state.

DFA Design Example and Exercise

- Draw a DFA for the language accepting strings ending with ‘abb’ over input alphabets ∑ = {a, b}
- Draw a DFA for the language accepting strings starting with ‘ab’ over input alphabets ∑ = {a, b}
- Draw a DFA for the language accepting strings ‘ab’ in the middle (sub string) over input alphabets ∑ = {a, b}

Lecture References

- Portland State University Lectures (Link)
- Power set Construction Wikipedia (Link)
- Maynooth University Lectures (Link)

References/Books

- 1. Compilers-Principles, techniques and tools (2nd Edition) V. Aho, Sethi and D. Ullman
- 2. Principles of Compiler Design (2nd Revised Edition 2009) A. A. Puntambekar
- 3. Basics of Compiler Design Torben Mogensen