AMERICAN INTERNATIONAL UNIVERSITY-BANGLADESH

Faculty of Science and Technology

AIUB

DEPARTMENT OF COMPUTER SCIENCE

CSC3113: THEORY OF COMPUTATION

Lecture: # 3 Week: # 2 Semester: Spring 2022-2023

DETERMINISTIC FINITE AUTOMATON (DFA)
REGULAR LANGUAGE

Instructor: Shakila Rahman, Lecturer,

Department of Computer Science, Faculty of Science & Technology.

Shakila.Rahman@aiub.edu

Lecture Outline

- Practice Problem with DFA.
  - Exercise solving from the book.
  - DFA design Issues.
- Regular Language
  - Closure
  - Operations
  - Example: Regular Language closed under Union Operation

Learning Objective

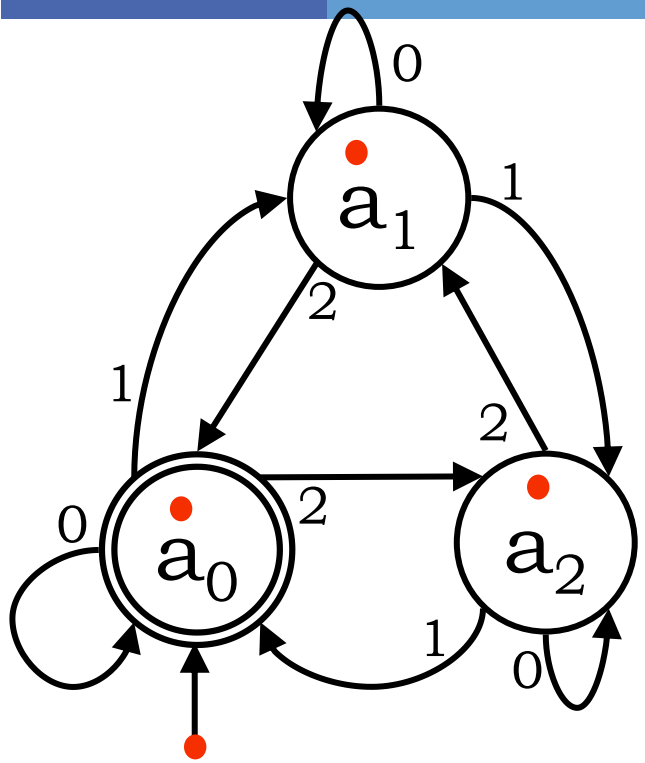
- Understand, learn & practice with example
  - Practice designing DFA.
- Learn Language, Regular Language & Regular Operations
- Analyze Closure under regular Operation
- Build one machine from 2 machine using closure under union

Learning Outcome

All outcome are represented with examples

- Understand, learn & practice design of DFA.
- Analyze and Design new machine model from one or more machine model(s) using closure rule of regular operation (example: Union).
- In doing so, understand that, there are certain cases where DFA might not give a desired machine model.

Designing DFA – Example 1



- Input example: 01120101
- Present State:

a_2

- Input symbol:

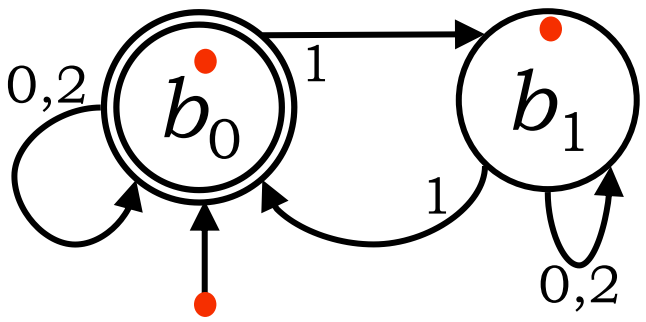
Accepted

ε

- **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.
    - We can model a_i as state representing *S* modulo 3 = *i*.
- The finite state machine $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$, 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

Designing DFA – Example 2



- Alphabet Σ={0,1,2}.
- Language A_2 = {*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

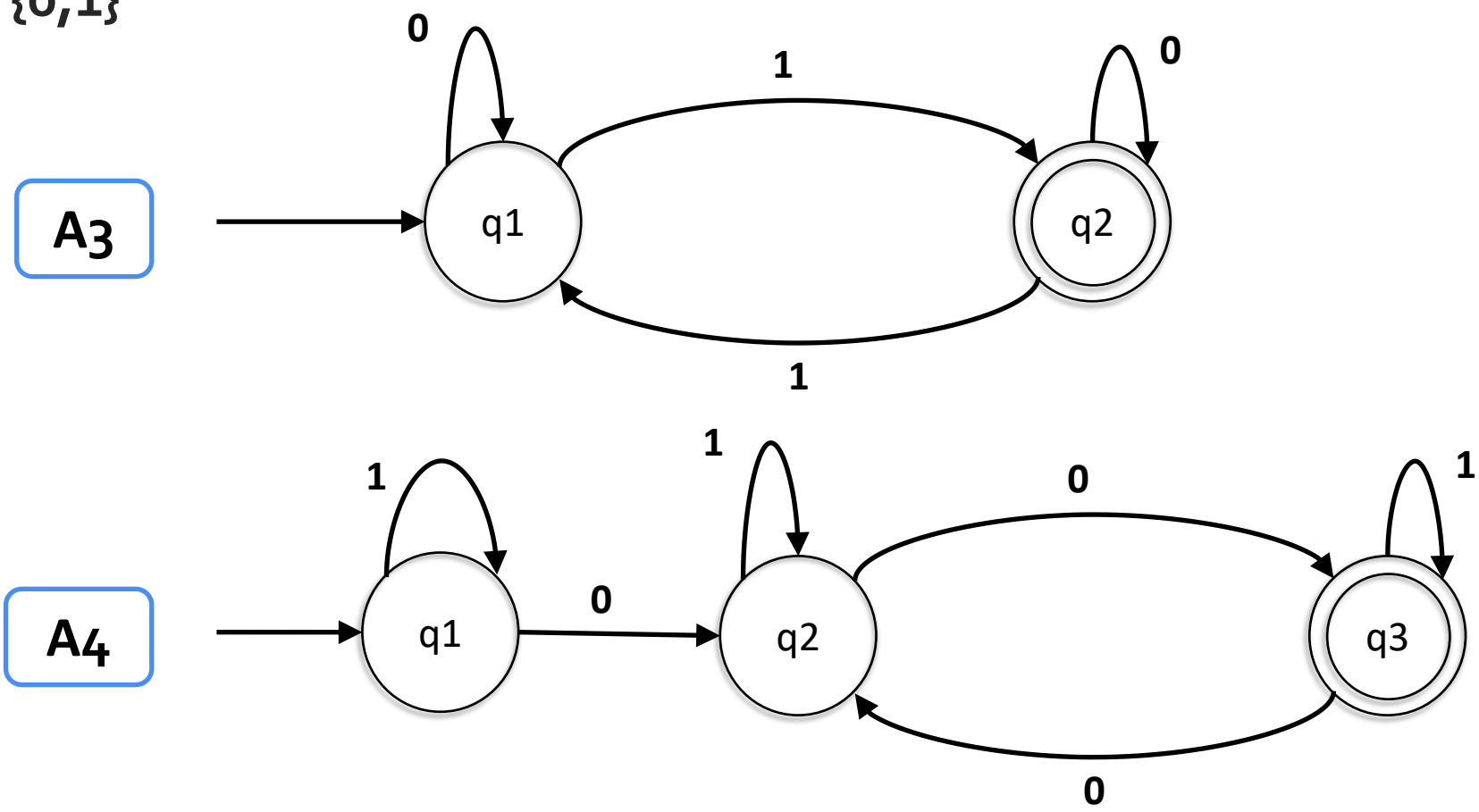
DFA DESIGN ISSUES

- The key concept here is to understand the given language.
  - Type/pattern of input strings that the language gives.
  - Match the pattern from left to right with the states & transitions of the state diagram, one by one.
  - First match the necessary conditions with the transition then complete the diagram with rest of the transition maintaining the rules.
  - Maintain the rules –
    - From each state there must be exactly one transition for each input symbol.
    - Self-loop represents zero or more number of occurrences of the input symbol.
    - There can be one or more number of final states.
  - Make some of your own input strings based on the given language. Few that should be ACCEPTED and few that should be REJECTED. Test them on your state diagram after completing or while drawing the state diagram and update accordingly.
- Let us practice some DFA in the following slides.

A_3 ={w : w is a binary string containing an odd number of *1*s}.

A_4 ={w : w is a binary string containing an even number of *0*s}.

$\sum = \{0,1\}$

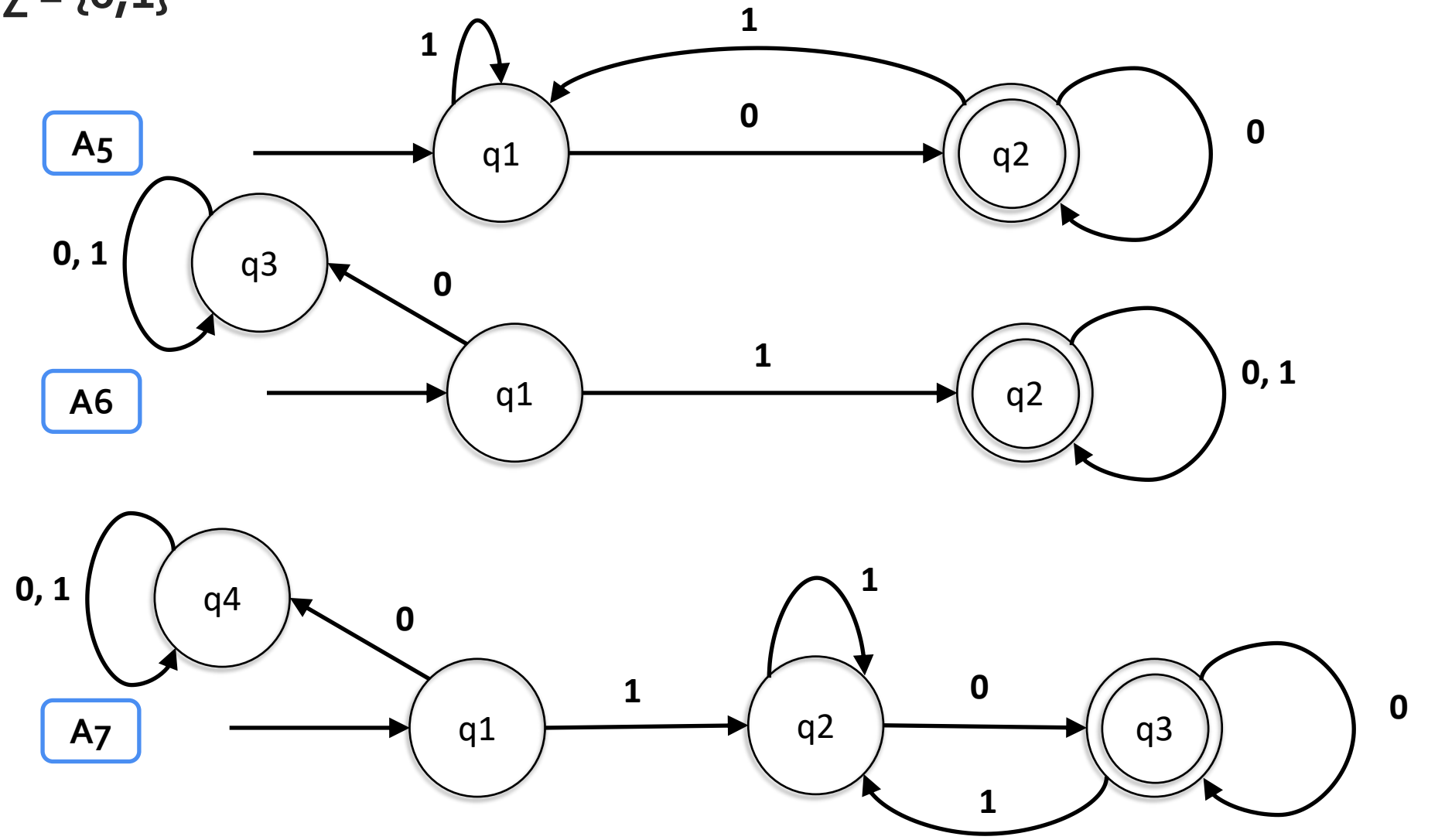


A5 ={w| w ends with a *0*}.

A6 ={w| w begins with a *1*}.

A7 ={w| w begins with a *1* and ends with a *0*}.

∑ = {0,1}

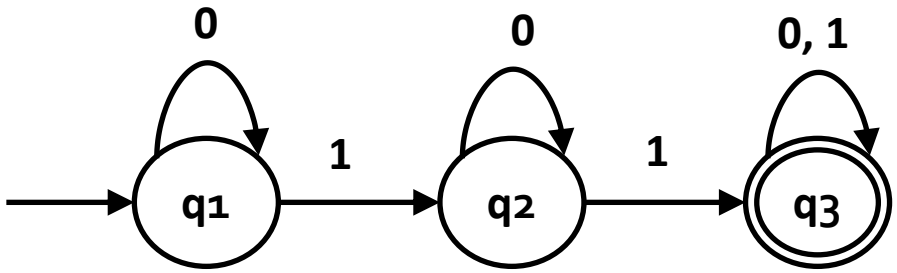


A8 ={w| w has at least two *1*s}.

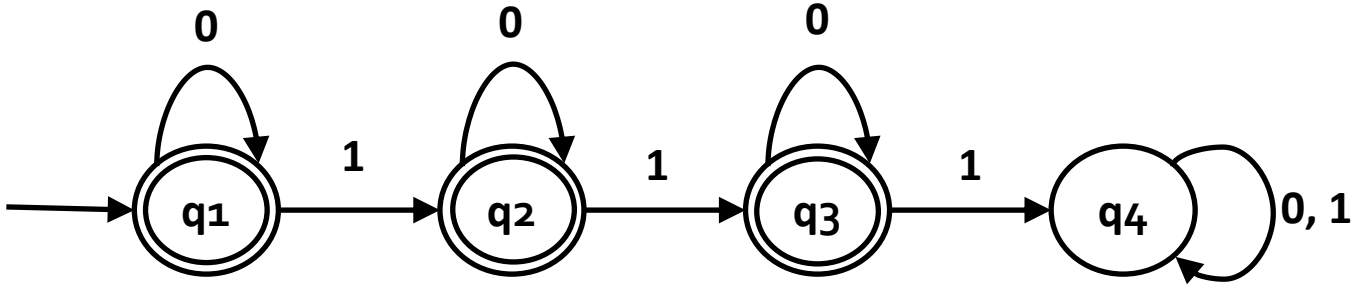
A9 ={w| w has at most two *1*s}.

∑ = {0,1}

A8



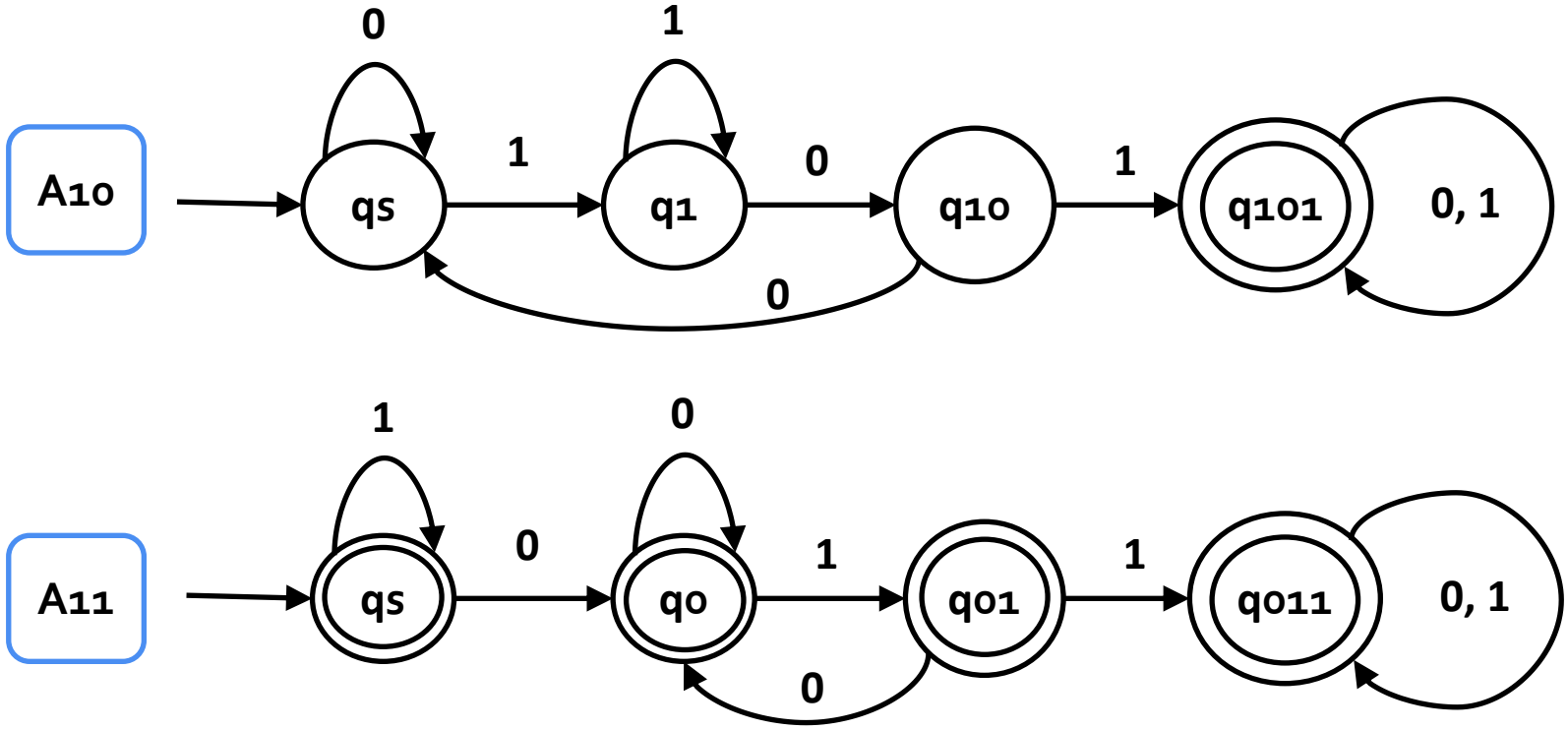
A9



A10 ={w| w has substring 101}.

A11 ={w| w has substring 011}.

∑ = {0,1}



What happens for the language, **A11 ={w| w does not have substring 011}**?

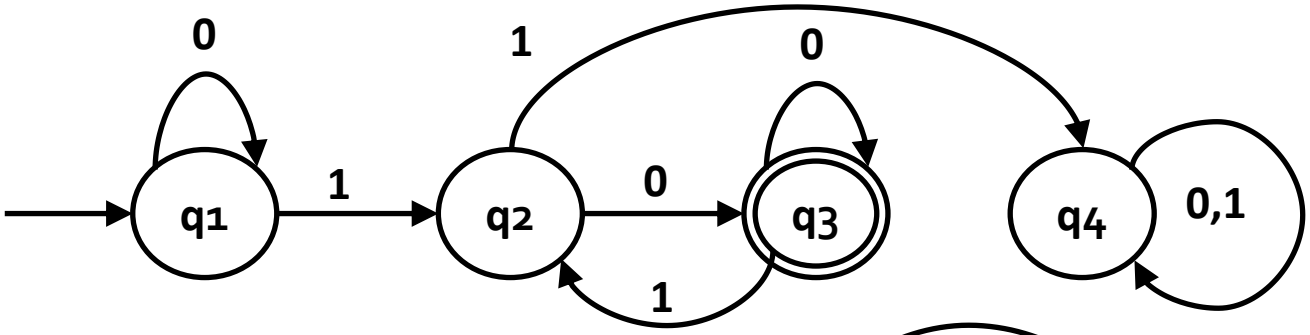
A12={w| each *1* in w is followed by at least one *0*}.

A13 ={w | The length of w is at least three and have *0* in 2nd last position}

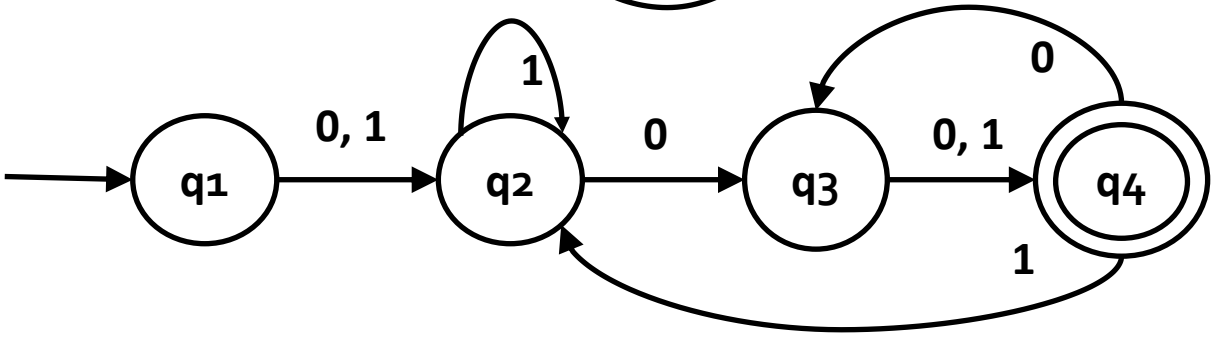
A14 ={w : The length of w is even and contains *1* in every odd position}

∑ = {0,1}

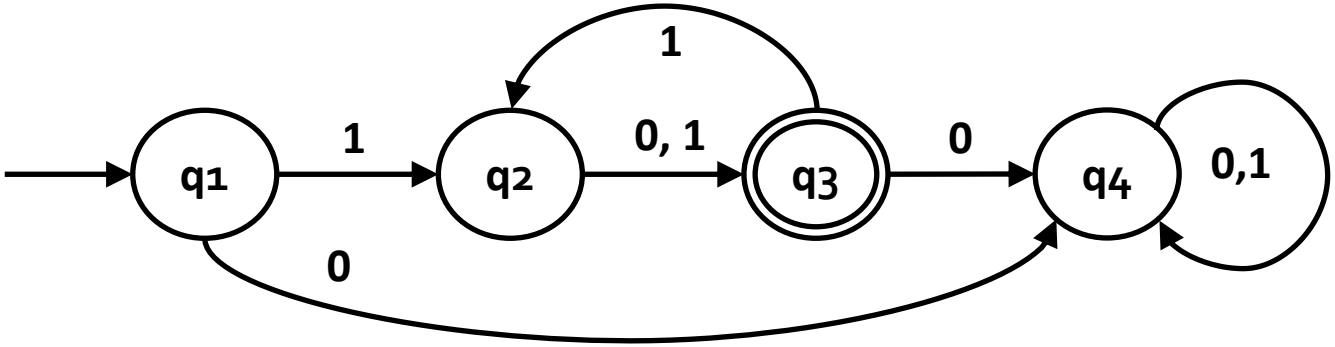
A12



A13



A14



Regular Operations

An Analogy

- In algebra, we try to identify operations which are common to many different mathematical structures.
- Example:
  - The integers $\mathbb{Z} = \{\dots, -2, -1, 0, 1, 2, \dots\}$ are closed under –
    - Addition: $X + Y$
    - Multiplication: $X \times Y$
    - Negation: $-X$
    - …but NOT Division: X/Y
- We'd like to investigate similar closure properties of the class of regular languages

REGULAR OPERATIONS ON LANGUAGES

- Let $A, B \subseteq \sum^*$ be languages.
- Union: $A \cup B = \{w \in \sum^* | w \in A \; or \; w \in B \}$.
- Concatenation: $A \circ B = \{wv | w \in A \; and \; v \in B \}$
- Star: $A^* = \{w_1 w_2 w_3 \ldots w_n | n \geq 0 \; and w_i \in A \; for \; i = 1 \ldots n\}$
- Complement: $\bar{A} = \{w \in \sum^* | w \notin A\}$
- Intersection: $A \cap B = \{w \in \sum^* | w \in A \; and \; w \in B \}$
- Reverse: $A^R = \{w_1 w_2 w_3 \ldots w_n | \; w_n w_{n-1} \ldots w_2 w_1 \in A\}$

Regular Language

- A language is called a ***regular language*** if some finite automaton recognizes it.
- Regular Operations: Let A={good, bad}, B = {boy, girl}.
- Basic 3 operations used to study the properties of the regular languages –
  - ***Union***: $A \cup B = \{x : x \in A \text{ or } x \in B\}$ = {good, bad, boy, girl}.
    - Takes all the strings in both A and B and lumps them together into one language.
  - ***Concatenation***: $A \bullet B = \{xy : x \in A \text{ and } y \in B\}$ = {goodboy, goodgirl, badboy, badgirl}.
    - Attaches a string from A in front of a string B in all possible ways to get the strings in the new language.
  - ***Star***: $A^* = \{x_1x_2...x_k : k \geq 0 \text{ and each } x_i \in A\}$ = {ε, good, bad, goodgood, goodbad, badgood, badbad, goodgoodgood, goodgoodbad, …}.
    - Attaching any number of strings in A together to get a string in the new language. It is a unary operation, where ε is always a member of A^* (as 'any number' also includes 0).

Closure

- A collection of objects is closed under some operation, if applying that operation to the members of the collection returns an object still in the collection.
- Theorem: The class of regular languages is closed under all three regular operations (union, concatenation, star), as well as under complement, intersection, and reverse.
  - i.e., if set *A* and *B* are regular, applying any of these operations on these sets yields a regular language.
- Next, we will prove it for *Union* operation.

Regular Language closed under Union

- We will prove it by construction.
- Let M_1 recognize A_1, where $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$, and
  M_2 recognize A_2, where $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$.
- Construct M to recognize $A_1 \cup A_2$, where $M = (Q, \Sigma, \delta, q_0, F)$.
  - $Q = \{(r_1, r_2) : r_1 \in Q_1 \text{ and } r_2 \in Q_2\}$.
    - $Q = Q_1 \times Q_2$. (All combination of states of machine M_1 and M_2).
  - $\Sigma = \Sigma_1 \cup \Sigma_2$.
    - But here, for simplicity, we have considered $\Sigma_1 = \Sigma_2$ to be same.
  - For each $(r_1,r_2) \in Q$ and each $a \in \Sigma$, let $\delta((r_1,r_2), a) = (\delta(r_1, a), \delta(r_2, a))$.
    - Hence δ gets a state of M (which actually is a pair of states from M_1 and M_2), together with an input symbol, and returns M's next state.
  - q_0 is the pair (q_1, q_2).
  - $F = \{ (r_1, r_2) : r_1 \in F_1 \text{ or } r_2 \in F_2 \}$
    - $F = (F_1 \times Q_2) \cup (Q_1 \times F_2)$

DFA Union Example

- Let,
- $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$, where –
  - $Q_1 = \{a_0, a_1, a_2\}$,
  - $\Sigma = \{0, 1, 2\}$
  - $q_1 = a_0$,
  - $F_1 = \{a_0\}$,
  - δ_1 **Machine 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

- $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$, where –
  - $Q_2 = \{b_0, b_1\}$,
  - $\Sigma = \{0, 1, 2\}$
  - $q_2 = b_0$,
  - $F_2 = \{b_0\}$,
  - δ_2 **Machine 2**

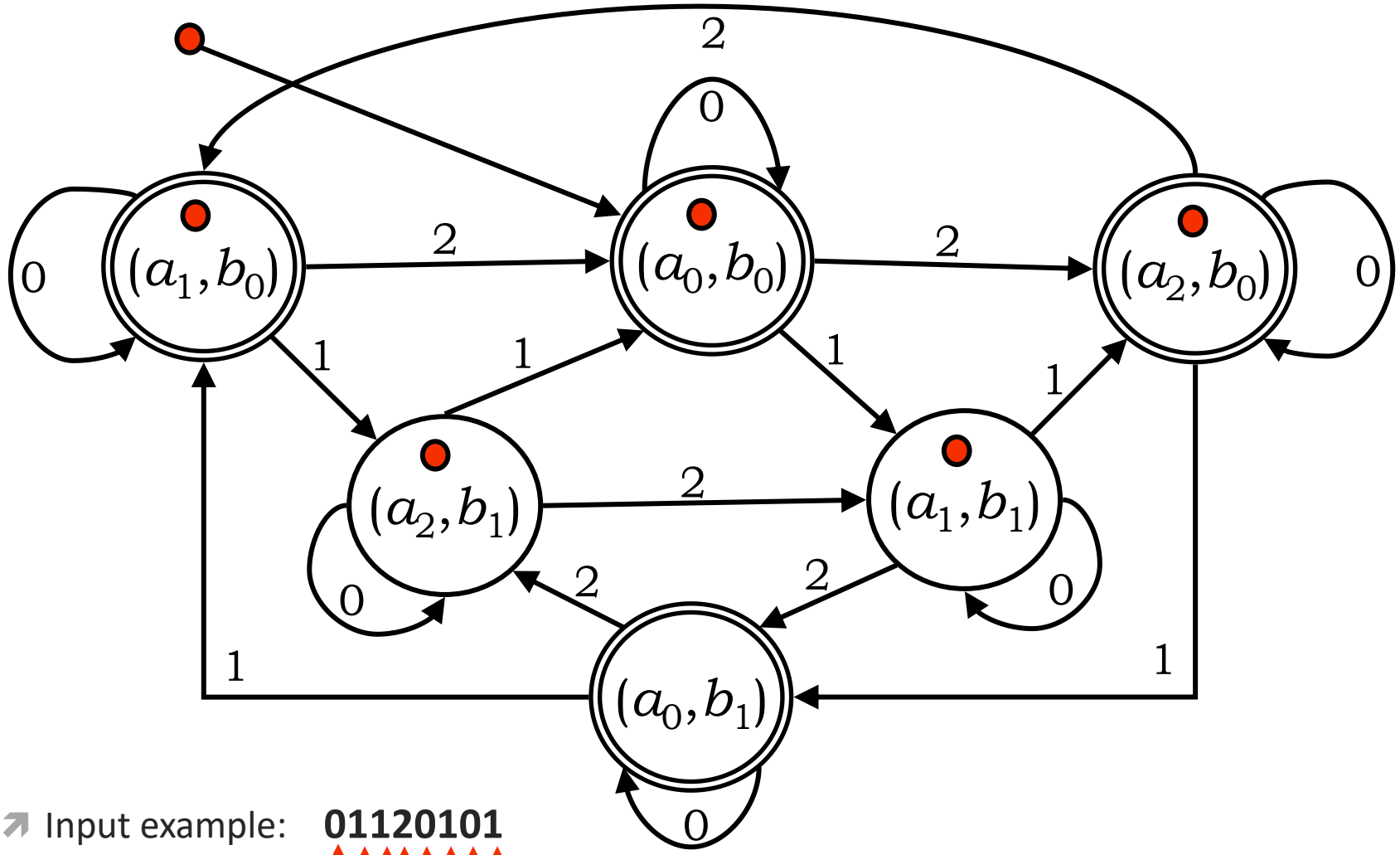
	0	1	2
b_0	b_0	b_1	b_0
b_1	b_1	b_0	b_1

- $M = (Q, \Sigma, \delta, q_0, F)$, where –
  - $Q = \{(r_1, r_2) : r_1 \in Q_1 \text{ and } r_2 \in Q_2\} = Q_1 \times Q_2$
    $Q = \{(a_0,b_0), (a_0, b_1),(a_1, b_0), (a_1, b_1),(a_2, b_0), (a_2, b_1)\}$,
  - $\Sigma = \Sigma_1 \cup \Sigma_2 = \{0, 1, 2\}$
  - $q_0 = (q_1,q_2) = (a_0, b_0)$
  - $F = \{ (r_1, r_2) : r_1 \in F_1 \text{ or } r_2 \in F_2 \} = (F_1 \times Q_2) \cup (Q_1 \times F_2)$
    $F = \{(a_0, b_0), (a_0, b_1), (a_1, b_0), (a_2, b_0)\}$
  -

δ	0	1	2
(a_0, b_0)	(a_0, b_0)	(a_1, b_1)	(a_2, b_0)
(a_0, b_1)	(a_0, b_1)	(a_1, b_0)	(a_2, b_1)
(a_1, b_0)	(a_1, b_0)	(a_2, b_1)	(a_0, b_0)
(a_1, b_1)	(a_1, b_1)	(a_2, b_0)	(a_0, b_1)
(a_2, b_0)	(a_2, b_0)	(a_0, b_1)	(a_1, b_0)
(a_2, b_1)	(a_2, b_1)	(a_0, b_0)	(a_1, b_1)

- The above finite automata machine should give the same output for any given input to machine M_1 or M_2.
- Here M_1 recognizes A_1 and M_2 recognizes A_2. So M should recognize $A = A_1 \cup A_2$.
- A_1= {w: sum of all the symbols in w is multiple of 3}.
- A_2 = {w: sum of all the symbols in w is even}.

DFA Union Simulation



- Input example: **01120101**
- Input symbol:

Accepted

Closure under Concatenation

- Let M_1 recognize A_1, where $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$, and M_2 recognize A_2, where $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$.
- Construct M to recognize $A_1 \circ A_2$, where $M = (Q, \Sigma, \delta, q_0, F)$.
- M must accept if its input can be broken into two pieces where M_1 accepts the first piece and M_2 accepts the second piece.

- ***Problem***: M doesn't know where to break its input. i.e., where the first part ends and the second begins.
- To solve the problem we will learn a new technique called *nondeterministic automaton*.

References

- Elements of the Theory of Computation, Papadimitriou (2nd ed), DFA + Exercise.
- Introduction to Automata Theory, Languages, and Computation (3rd ed), DFA + Exercise.