Theory of Automata and Languages

Finite Automata and Regular Languages - 1

Fall 2024

Sharif University of Technology

Mehran Moeini Jam

Overview

- Strings and Languages
- On Finiteness
- Deterministic Finite[-State] Automata
- Regular Languages
- Operations on Regular Languages
- Extended Transition Function
- Construction and Proof of Correctness

Recall: Strings

- In the study of language theory, we begin with a finite set of basic symbols (letters), usually denoted as Σ, known as the alphabet.

- We define a string as a sequence of symbols from an alphabet.

- For example, “automata”, “Language”, “inFiniTe”, “asdjsafasfh”, “oafsuasFnasf”, … are all strings based on the finite alphabet $\Sigma = \{a, A, b, B, \ldots, z, Z\}$. *

Concatenation

- Given sets A and B of strings, their Concatenation, denoted $A . B$ is defined by:

$$A . B = \{ xy : x \in A \text{ and } y \in B\}$$

- It is also a common practice in the literature to write AB instead of $A . B$ when referring to concatenation.

- We also use A^n to represent the concatenation of set A with itself n - 1 times.

Kleene Star

- The Kleene star of an alphabet Σ represents the set of all possible finite strings

  that can be formed using symbols from Σ. We denote it as Σ^*.

- Formally, using the concatenation operator, we can define it as follows:

$$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \ldots$$

  where $\Sigma^0 = \{\varepsilon\}$, $\Sigma^1 = \Sigma$, $\Sigma^2 = \Sigma \,.\, \Sigma$ and so forth.

- We can also define: $\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \ldots$

Languages

- Now we can define a language over an alphabet Σ to be any subset of Σ^* .

- So essentially, the class of all languages which can be defined over an alphabet Σ is equal to the power set of Σ^*, or $\wp(\Sigma^*)$.

- Intuitively, a language is any set of strings formed from an alphabet, like the Persian language, which forms a meaningful subset of all strings over the Persian alphabet.

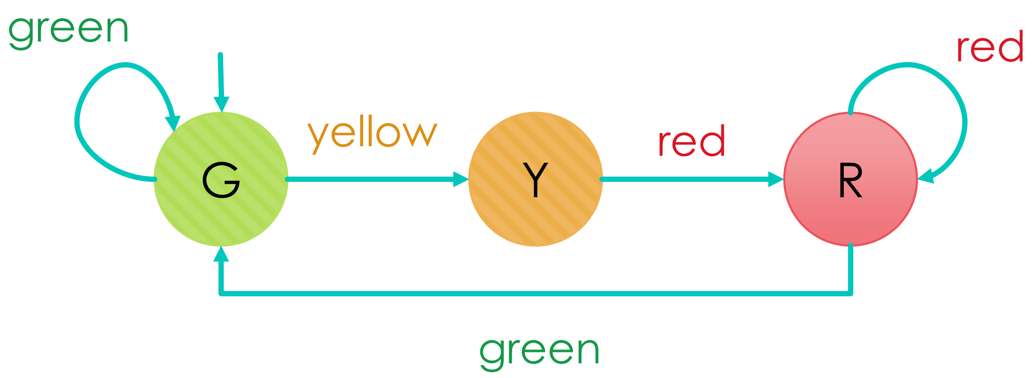
On Finiteness

- In computer science we find many examples of finite state systems, and the theory of finite automata is a useful tool for these systems.

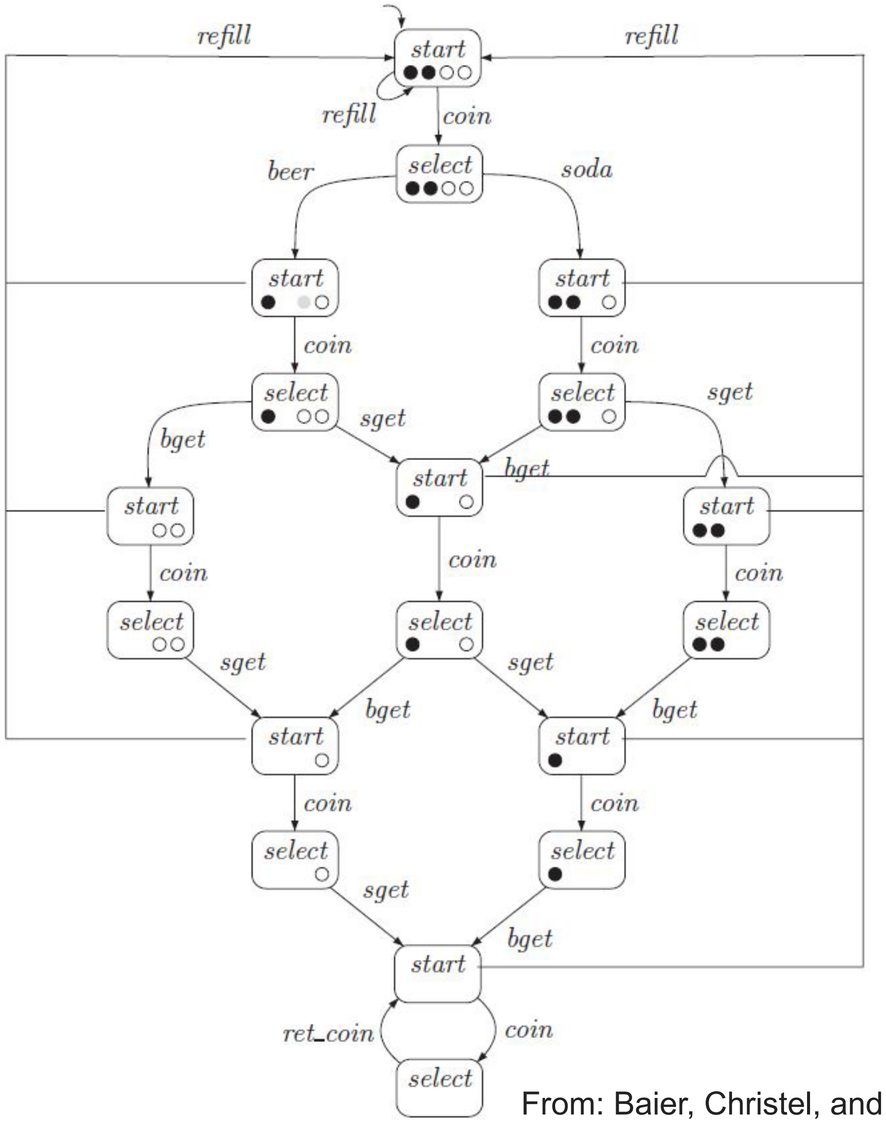
- The behavior of many software and hardware systems, can be specified using finite state systems. Some examples are text editors and the lexical analyzers found in most compilers.

Traffic Light

- Σ = {green, yellow, red}
- yellow red green yellow red
- green green yellow red green
- green yellow red red
- green green green yellow red red red
- green green green green green green yellow



Vending Machine



From: Baier, Christel, and Joost-Pieter Katoen. Principles of model checking. MIT press, 2008.

On Finiteness: Computer Programs

- Some programs written in common programming languages can also be modeled as finite state systems.

- To do that, we use the position of the program counter and the values assigned to bounded variables at each time as a guideline to define states.

Not suitable for all systems

- The computer itself can be viewed as a finite state system. Theoretically the state of the central processor, main memory, and auxiliary storage at any time is one of a very large, but finite number of states.

- Viewing a computer as a finite state system, however, is not satisfying mathematically or realistically.

So we'll need more powerful models, as we'll see in the future in the course.

The Membership Problem

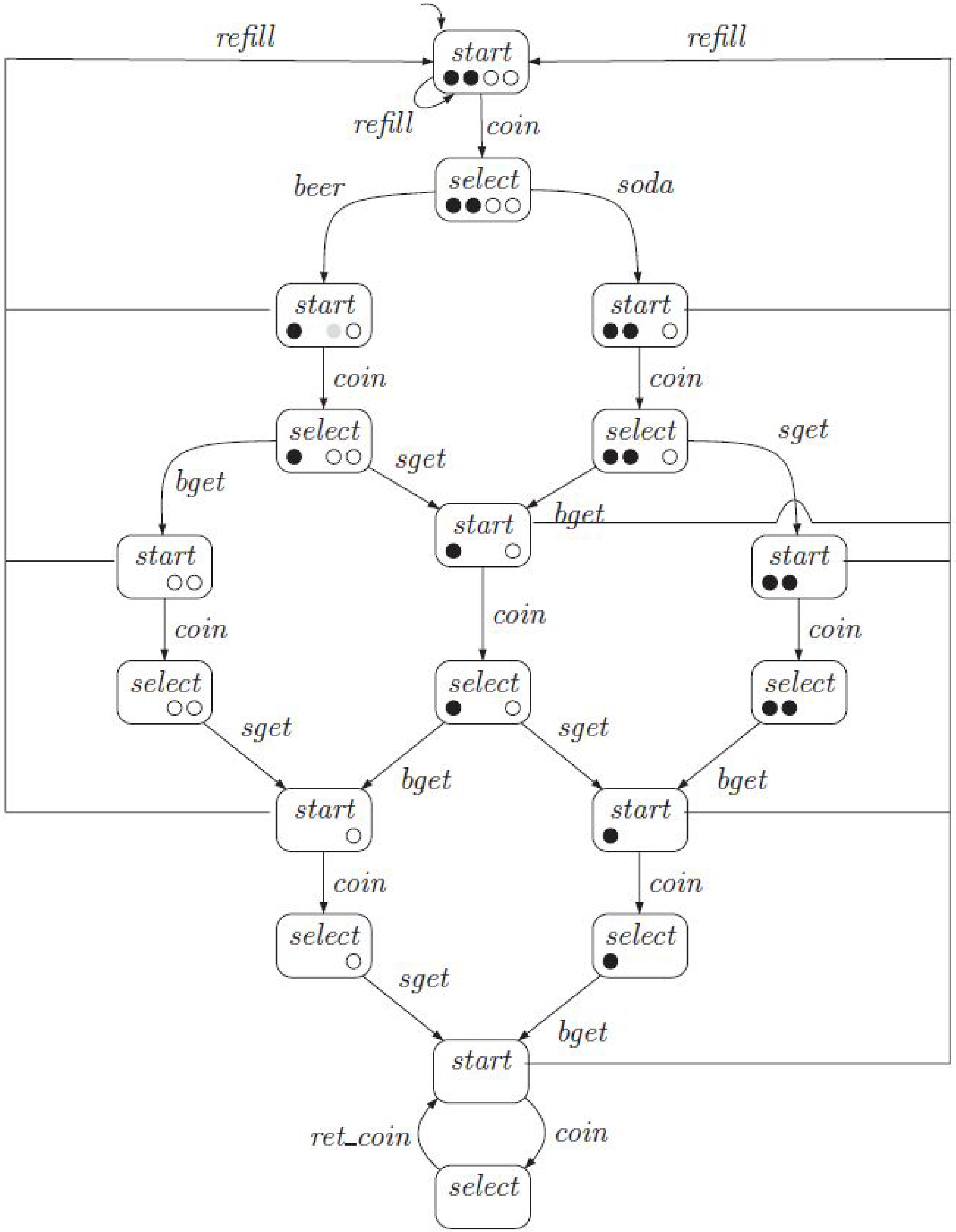
- Each of the systems presented in the previous slides corresponds to an infinite language, representing the behavior of the system in question.

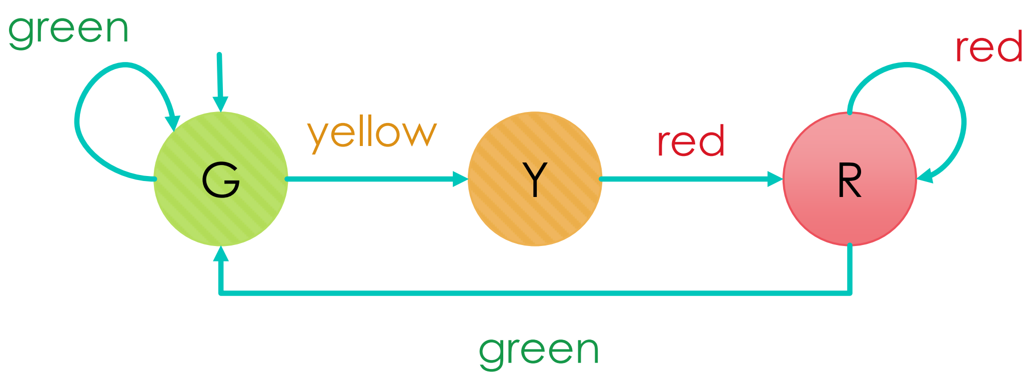
- One important question arises: How to determine whether a string is the member of the corresponding language?

The Membership Problem

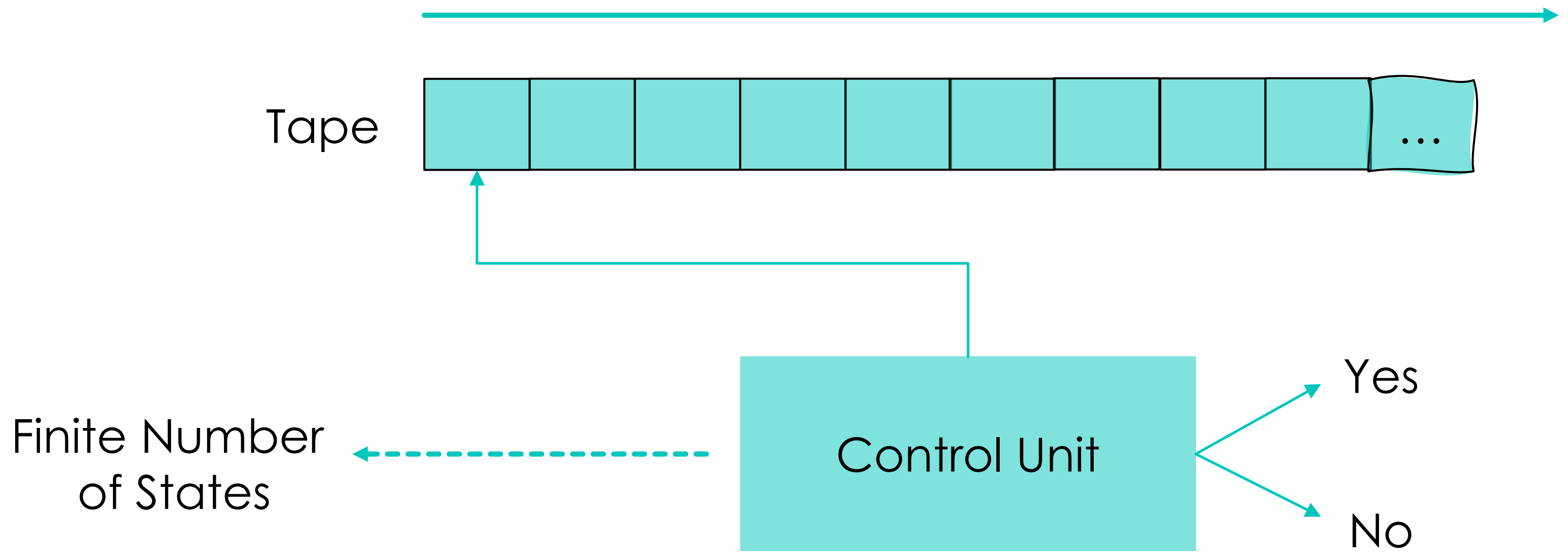
- In the formal language theory, the Membership Problem is the problem of determining whether a given string belongs to a specific language.

- The solution to this problem has much broader implications. It contributes to the development of a theory of computability and helps us address a wide range of problems.


refill
start
refill
coin
select
beer
soda
start
start
coin
coin
select
select
sget
bget
sget
start
bget
start
start
coin
coin
coin
select
select
select
sget
sget
bget
bget
start
start
coin
coin
select
select
sget
bget
start
ret_coin
coin
select


green
yellow
red
red
G
Y
R
green

Finite[-State] Automata (as an acceptor)



Deterministic Finite[-State] Automata (DFA)

A deterministic finite automaton is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where

- Q is a finite set called **states**,
- Σ is a finite set called the **alphabet**,
- $\delta : Q \times \Sigma \longrightarrow Q$ is **the transition function**,
- $q_0 \in Q$ is the **start state**, and
- $F \subseteq Q$ is the **set of accept (final) states**.

Example

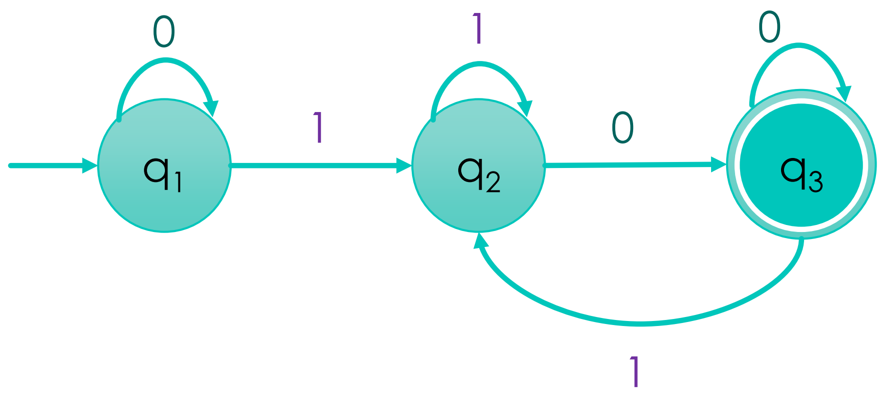
- $M_1 = (Q, \Sigma, \delta, q_0, F)$, where
- $Q = \{q_1, q_2, q_3\}$,
- $\Sigma = \{0,1\}$,
- δ is described as

$$\delta(q_1,0) = q_1 \qquad \delta(q_1,1) = q_2$$

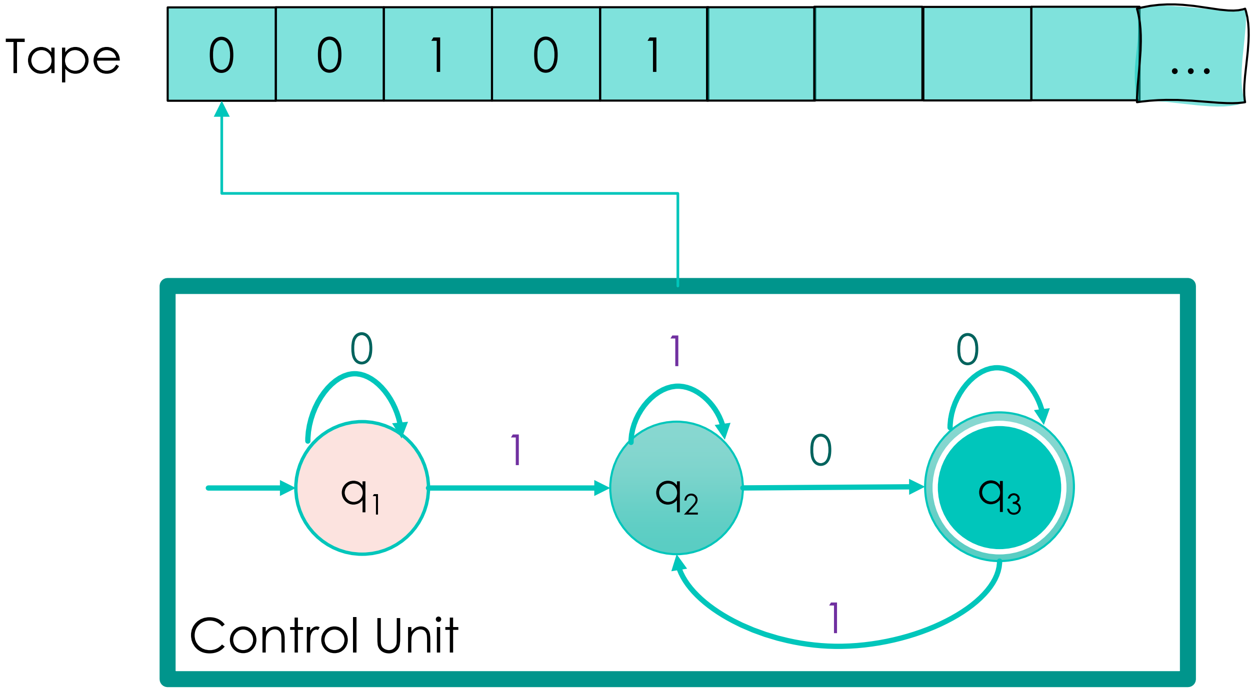
$$\delta(q_2,0) = q_3 \qquad \delta(q_2,1) = q_2$$

$$\delta(q_3,0) = q_2 \qquad \delta(q_3,1) = q_2$$

- $q_0 = \{q_1\}$
- $F = \{q_3\}$

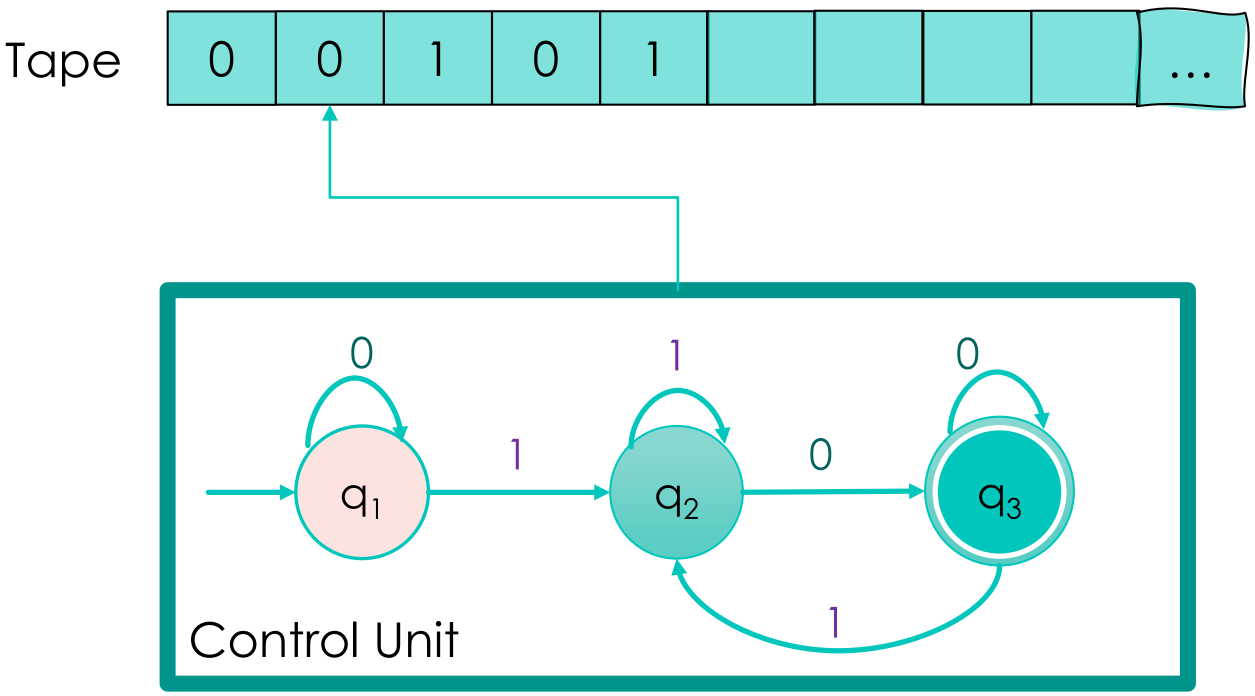


Example

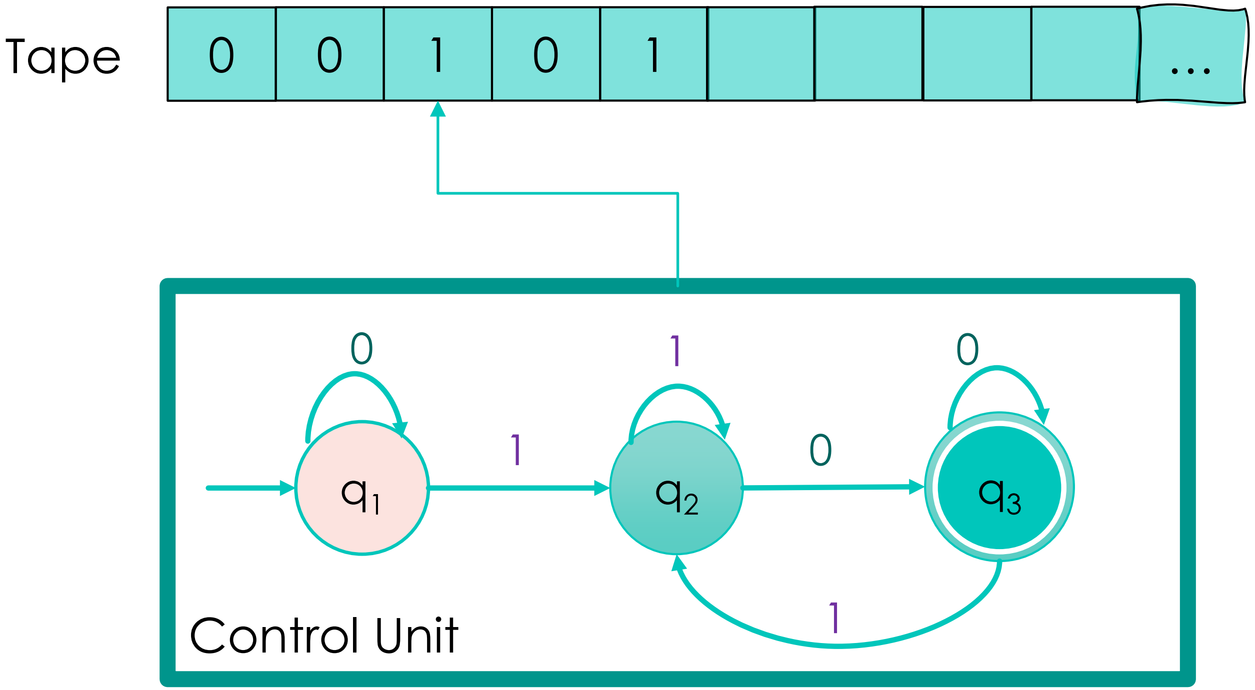


Initial Configuration

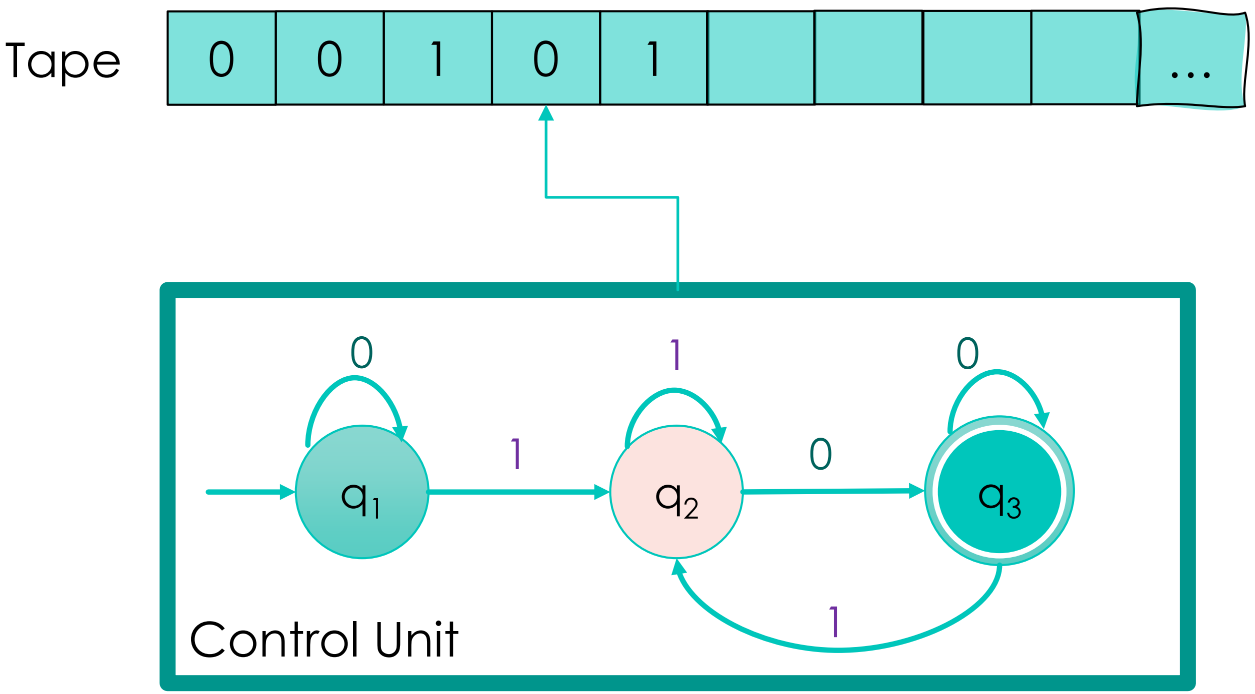
Example



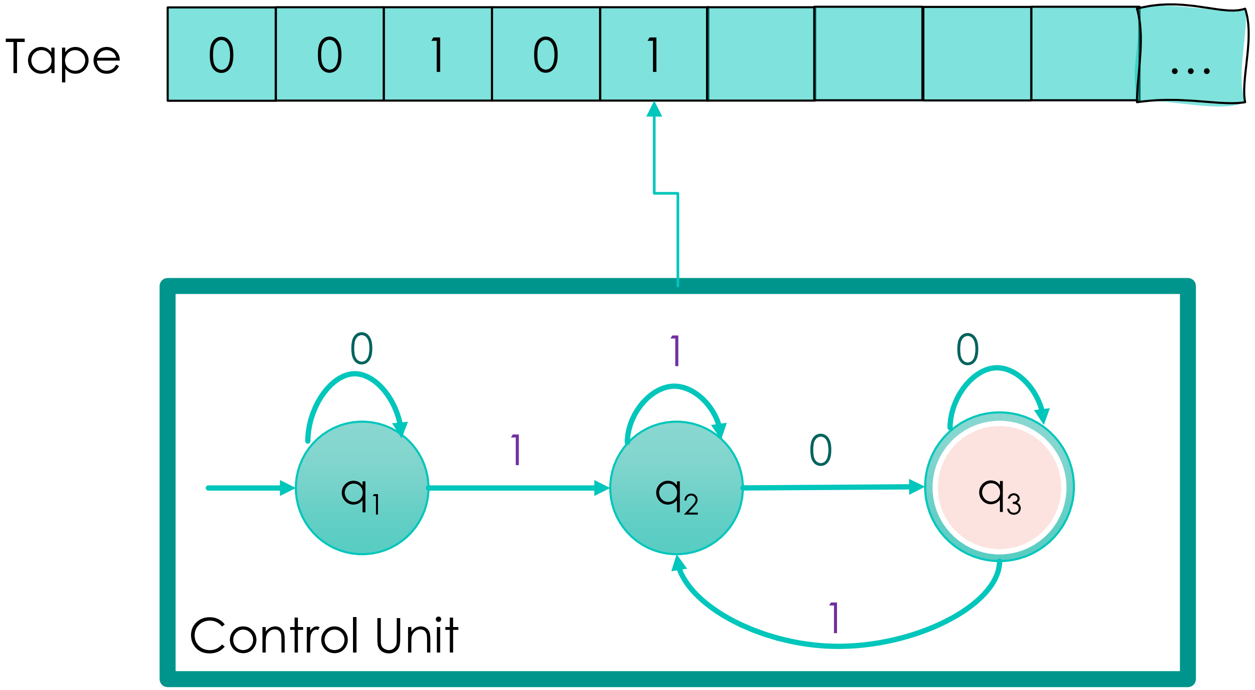
Example



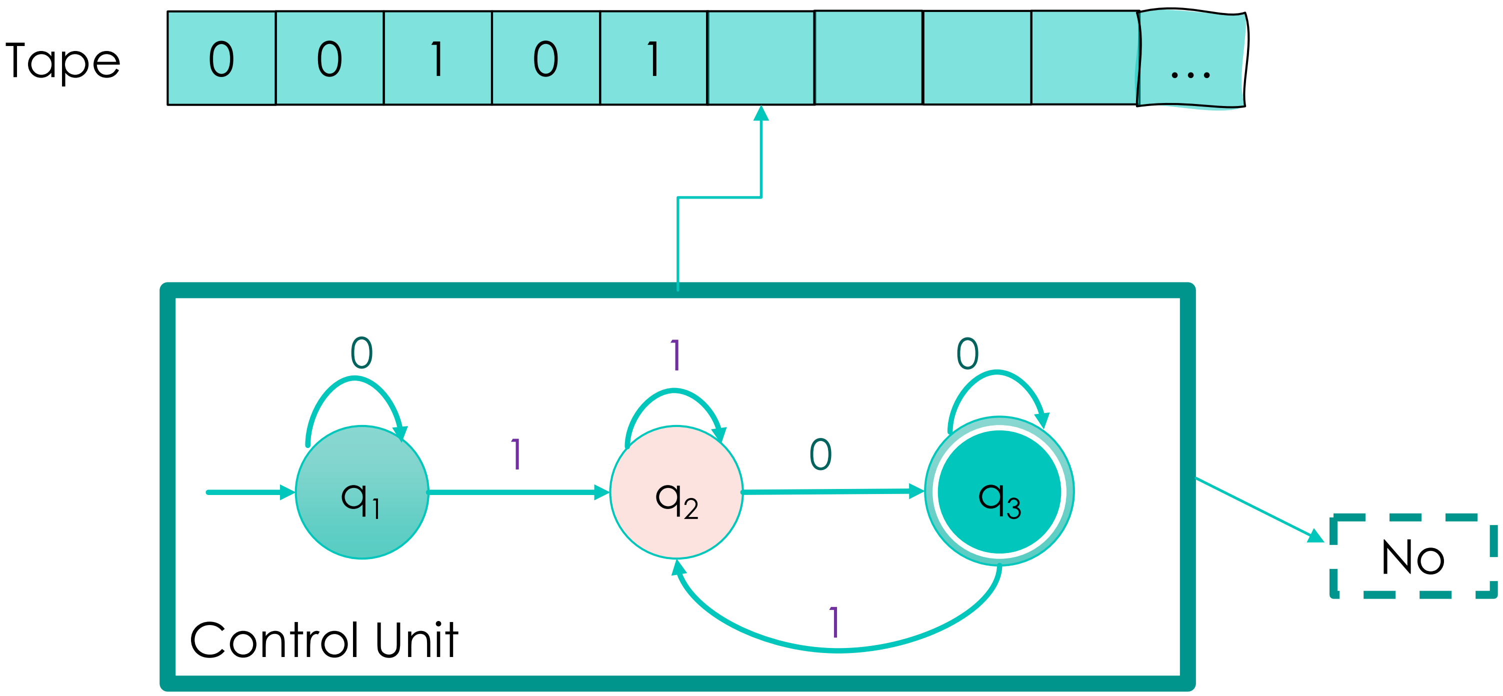
Example



Example



Example



Example: Traffic Light

- Σ = {green, yellow, red}

✓ yellow red green yellow red

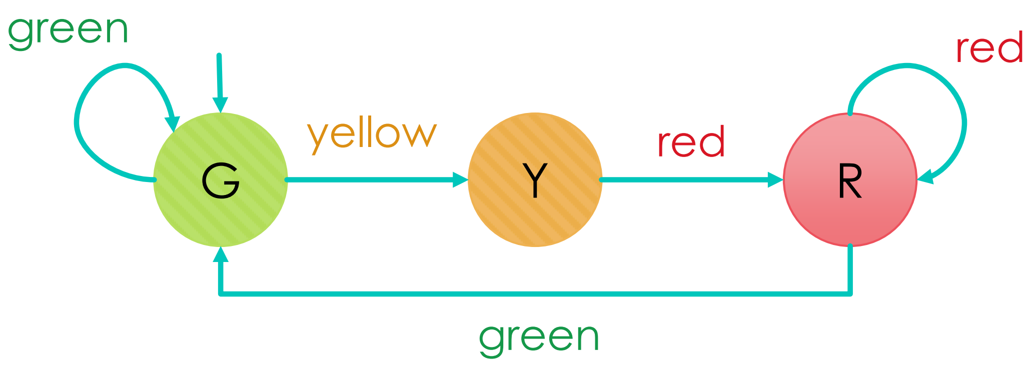
✓ green green yellow red green

✓ green yellow red red

✓ green green green yellow red red red

✓ green green green green green green yellow

X green green green red green green yellow



Example: Traffic Light

- Σ = {green, yellow, red}

✓ yellow red green yellow red

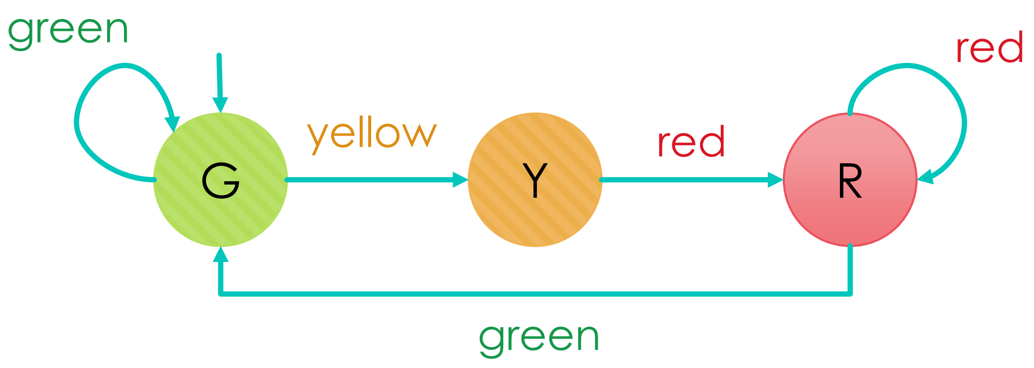
✓ green green yellow red green

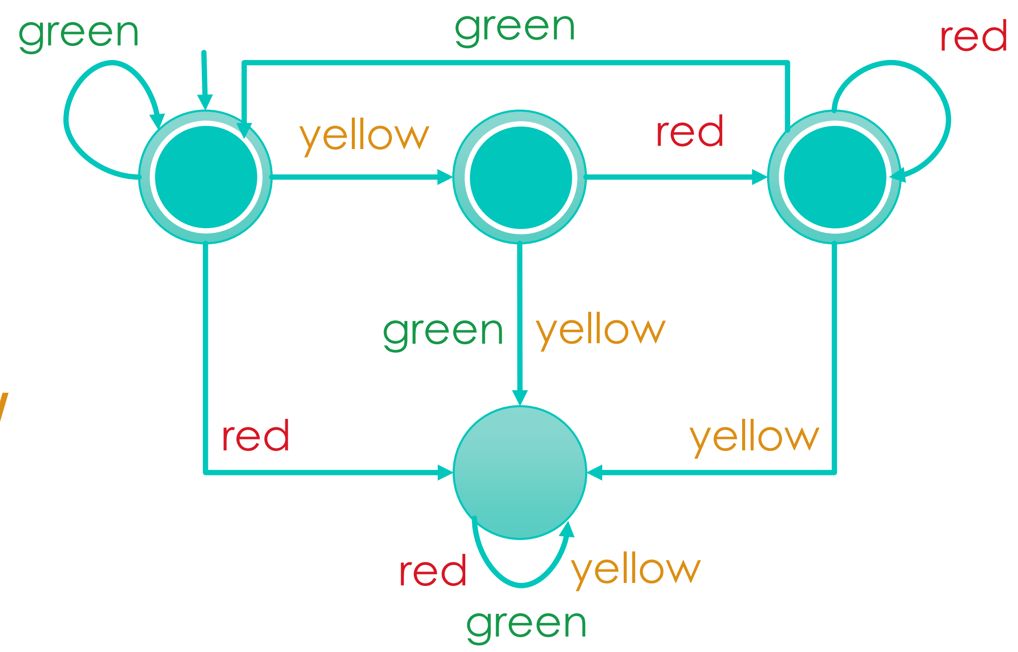
✓ green yellow red red

✓ green green green yellow red red red

✓ green green green green green green yellow

X green green green red green green yellow





Example: Traffic Light

- Σ = {green, yellow, red}

✓ yellow red green yellow red

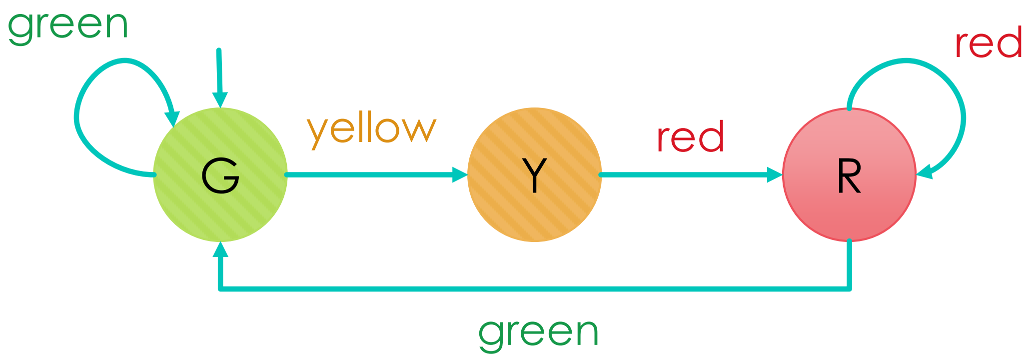
✓ green green yellow red green

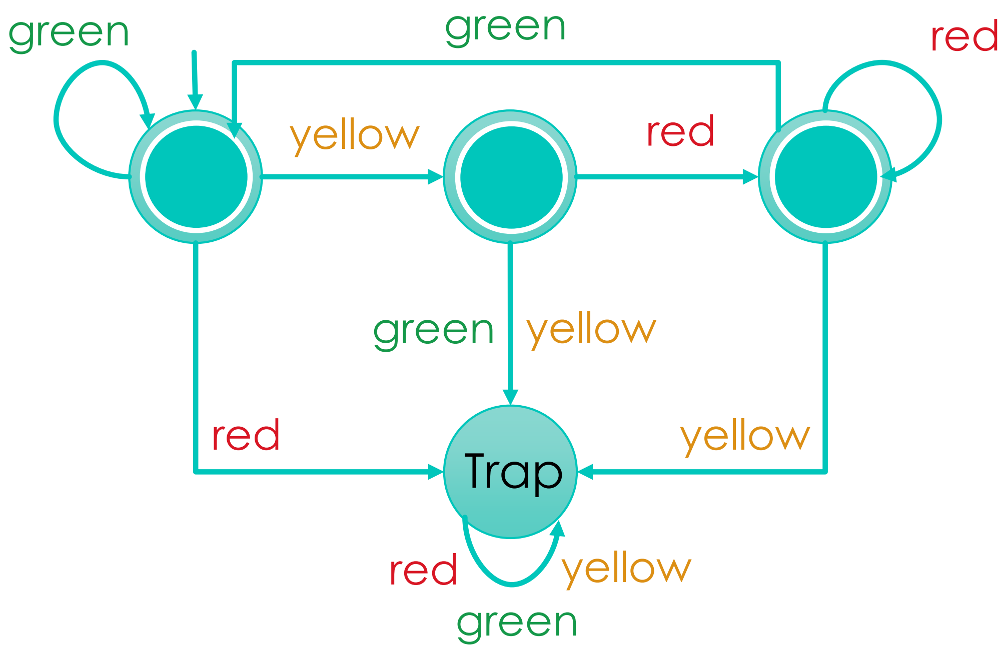
✓ green yellow red red

✓ green green green yellow red red red

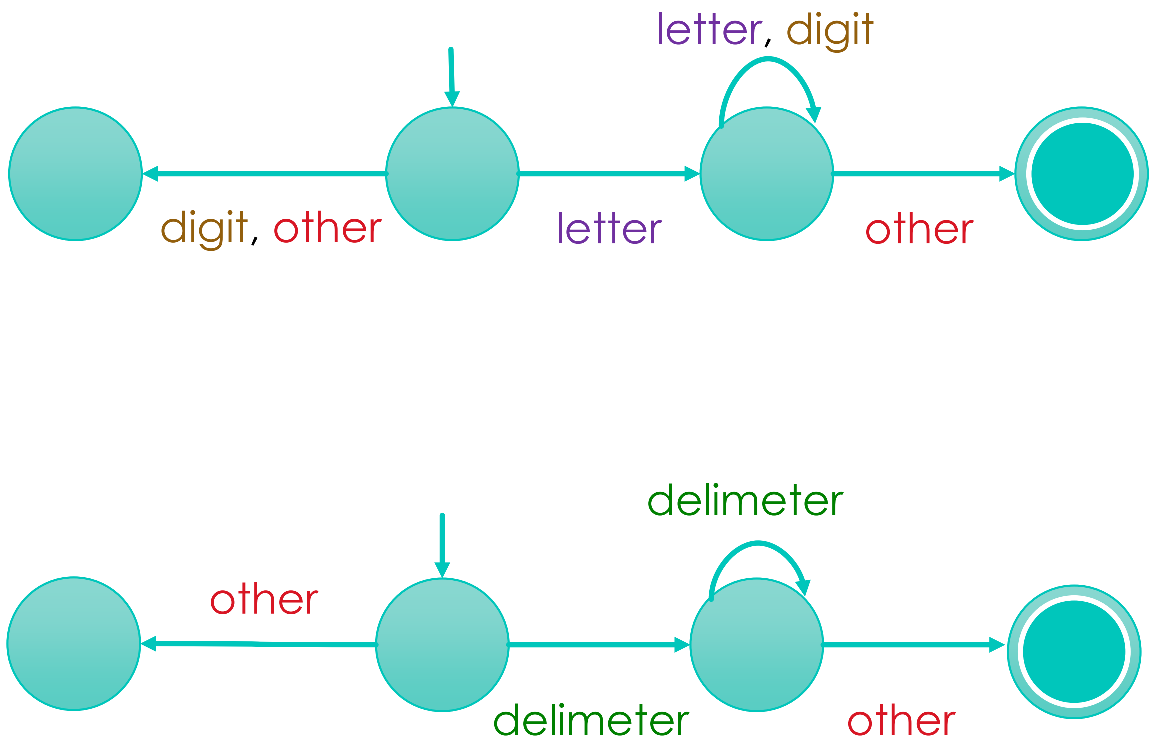
✓ green green green green green green yellow

X green green green red green green yellow





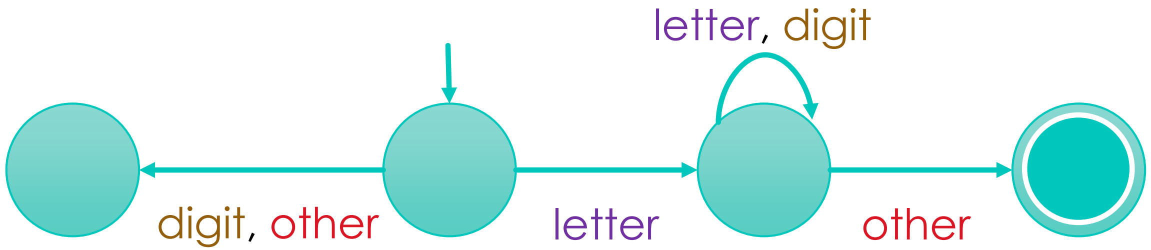
Example: Lexical Analyzer

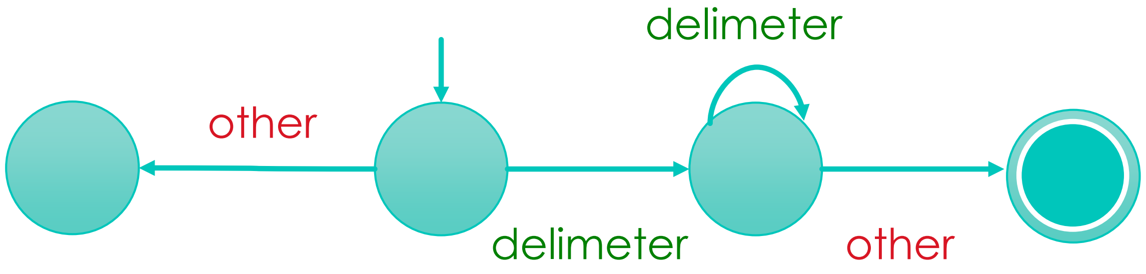


Pitfall ! Find the Error

$(Q, \Sigma, \delta, q_0, F)$ where

- Q is a finite set called **states**,
- Σ is a finite set called the **alphabet**,
- $\delta : Q \times \Sigma \longrightarrow Q$ is the **transition function**,
- $q_0 \in Q$ is the **start state**, and
- $F \subseteq Q$ is the **set of accept (final) states**.





Regular Languages

- Let M = $(Q, \Sigma, \delta, q_0, F)$ be a deterministic finite automata

  and $\omega = a_1 a_2 \cdots a_n$ be a string defined over Σ.

- We say M accepts ω iff a sequence of states $r_0, r_1, \ldots, r_n$ in Q exists such that:

  1. $r_0 = q_0$
  2. $\delta(r_i, a_{i+1}) = r_{i+1}$ for $i = 0, \ldots, n-1$,
  3. $r_n \in F$

Regular Languages

- We say that **M** accepts (**recognizes**) language **A** iff A = { ω ∈ Σ* : M accepts ω }

- A language is called a regular language iff some finite automaton recognizes it.

Question

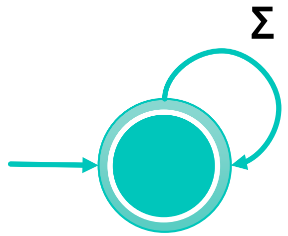
Given a Language L and a DFA M,

if M accepts every string in L, can we conclude that that M accepts L?

Pitfall

- One important point to consider is that we exclusively say that a DFA M accepts a language L when it accepts every string in L and rejects all other strings.

- Note that it's easy to describe a one-state DFA that accepts every string; which points us to the fact that the language Σ^* is a regular language.



- But, many subsets of Σ^* are not regular.

Pitfall

- So a finite automaton does its job by distinguishing between strings in **A** and strings not in **A**.

- Note that a DFA only accepts a single language, but each language essentially may have infinitely many finite automata that accept it.

Constructive Proofs

- Many theorems state that a particular type of object exists.
- One way to prove such a theorem is by demonstrating how to construct the object. This technique is called a proof by construction.
- We will use it to prove some closure properties on regular languages.

Operations on Regular Languages : Union

- Suppose $M_1 = (Q_1, \Sigma, \delta_1, q_0^1, F_1)$ and $M_2 = (Q_2, \Sigma, \delta_2, q_0^2, F_2)$ are two deterministic finite automata and $a \in \Sigma$.

- Let M be the deterministic finite automata $(Q, \Sigma, \delta, q_0, F)$ where

  $Q = Q_1 \times Q_2$ $\qquad$ $q_0 = (q_0^1, q_0^2)$ $\qquad$ $F = (F1 \times Q2) \cup (Q1 \times F2)$

  $\delta((q_1, q_2), a) = (\delta_1(q_1, a), \delta_2(q_2, a))$

  Then M recognizes $L(M_1) \cup L(M_2)$

Operations on Regular Languages : Intersection

- Suppose $M_1 = (Q_1, \Sigma, \delta_1, q_0^1, F_1)$ and $M_2 = (Q_2, \Sigma, \delta_2, q_0^2, F_2)$ are two deterministic finite automata and $a \in \Sigma$.

- Let M be the deterministic finite automata $(Q, \Sigma, \delta, q_0, F)$ where

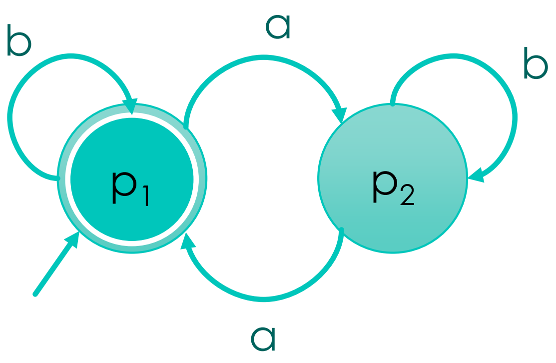
  $Q = Q_1 \times Q_2$ $\qquad$ $q_0 = (q_0^1, q_0^2)$ $\qquad$ $F = (F1 \times Q2) \cap (Q1 \times F2)$

  $\delta((q_1, q_2), a) = (\delta_1(q_1, a), \delta_2(q_2, a))$

  Then M recognizes $L(M_1) \cap L(M_2)$

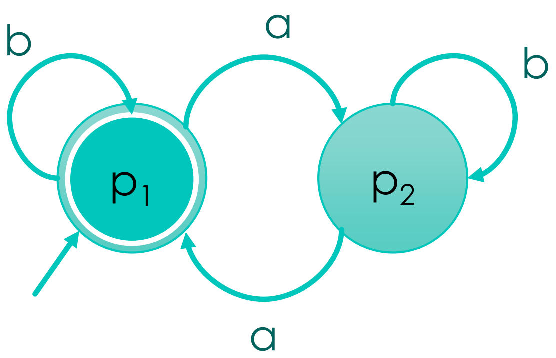
Example

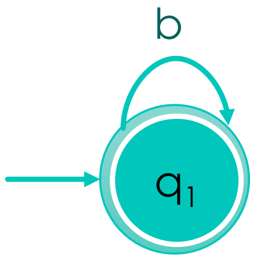
- The set of all strings over the alphabet $\Sigma = \{a, b\}$ such that in each string, the number of **a**'s is even and doesn't include the substring aa.



Example

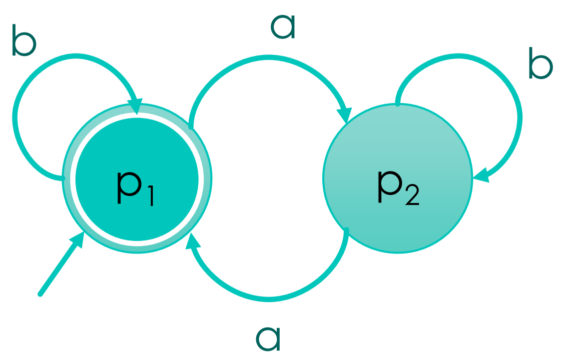
- The set of all strings over the alphabet $\Sigma = \{a, b\}$ such that in each string, the number of **a**'s is even and doesn't include the substring aa.

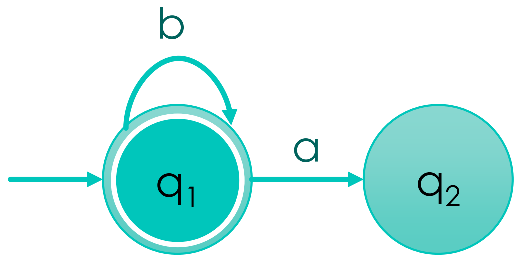




Example

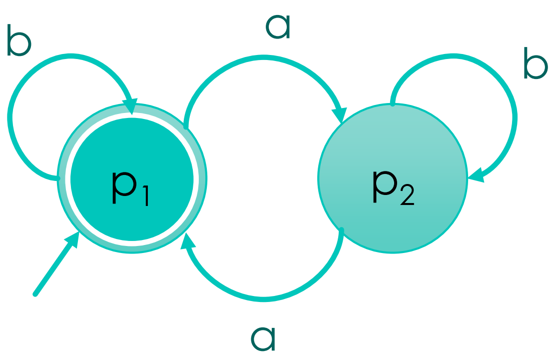
- The set of all strings over the alphabet $\Sigma = \{a, b\}$ such that in each string, the number of **a**'s is even and doesn't include the substring aa.

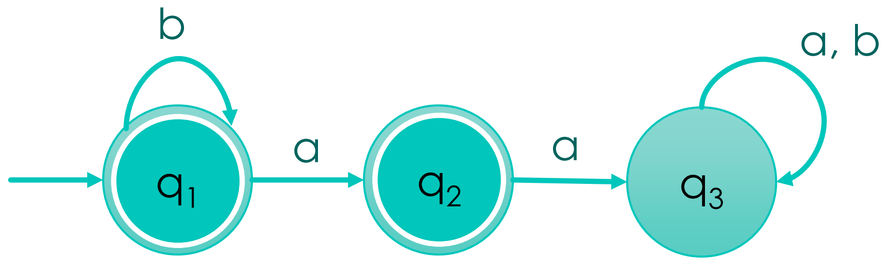




Example

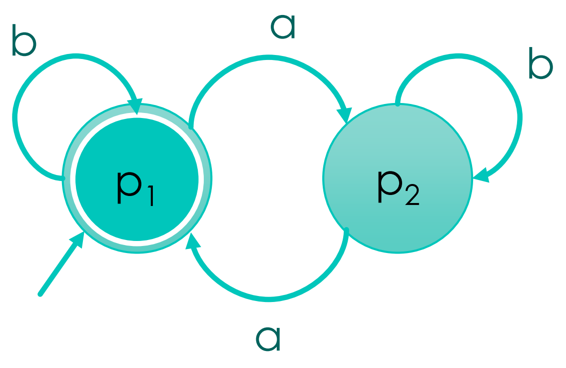
- The set of all strings over the alphabet $\Sigma = \{a, b\}$ such that in each string, the number of a's is even and doesn't include the substring aa.

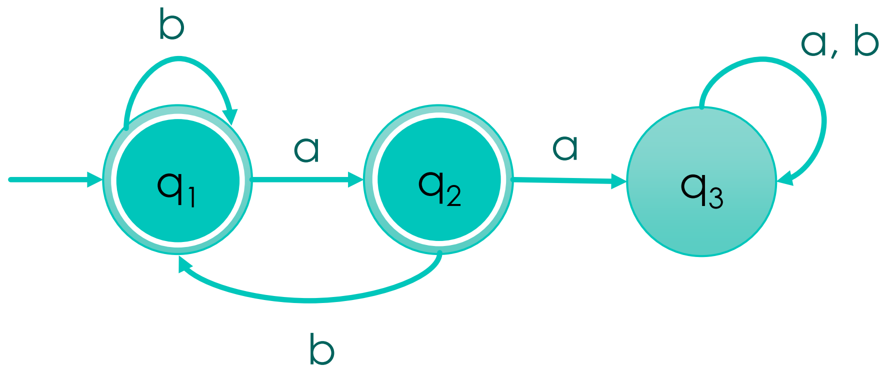




Example

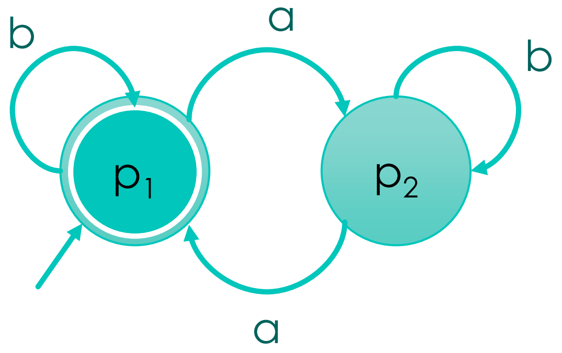
- The set of all strings over the alphabet $\Sigma = \{a, b\}$ such that in each string, the number of a's is even and doesn't include the substring aa.

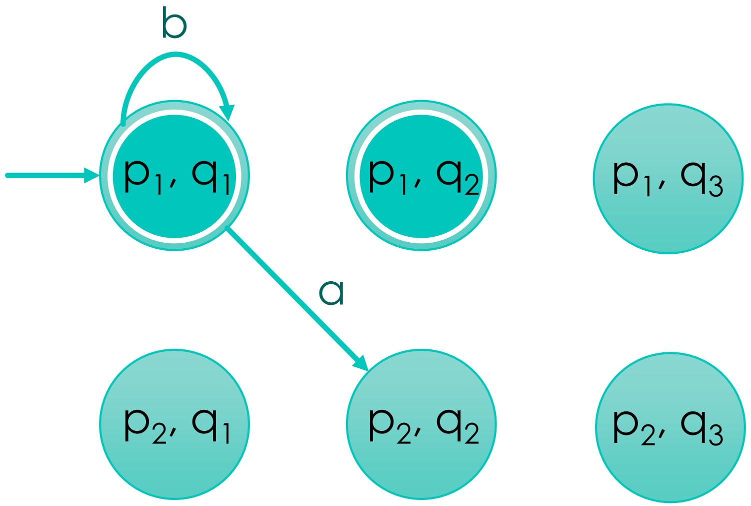


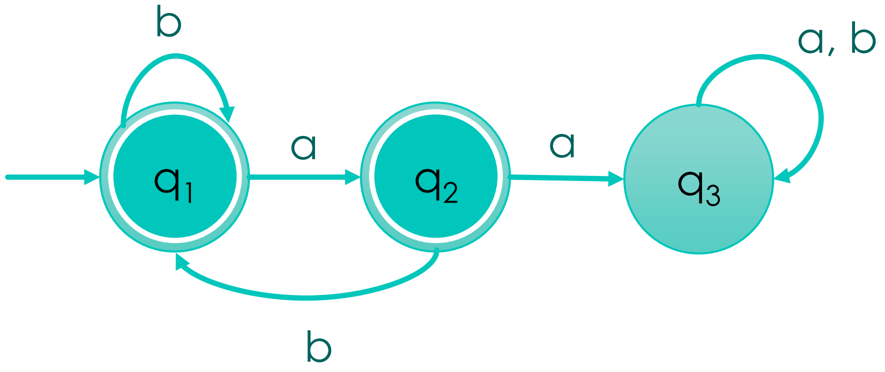


Example

- The set of all strings over the alphabet $\Sigma = \{a, b\}$ such that in each string, the number of a's is even and doesn't include the substring aa.

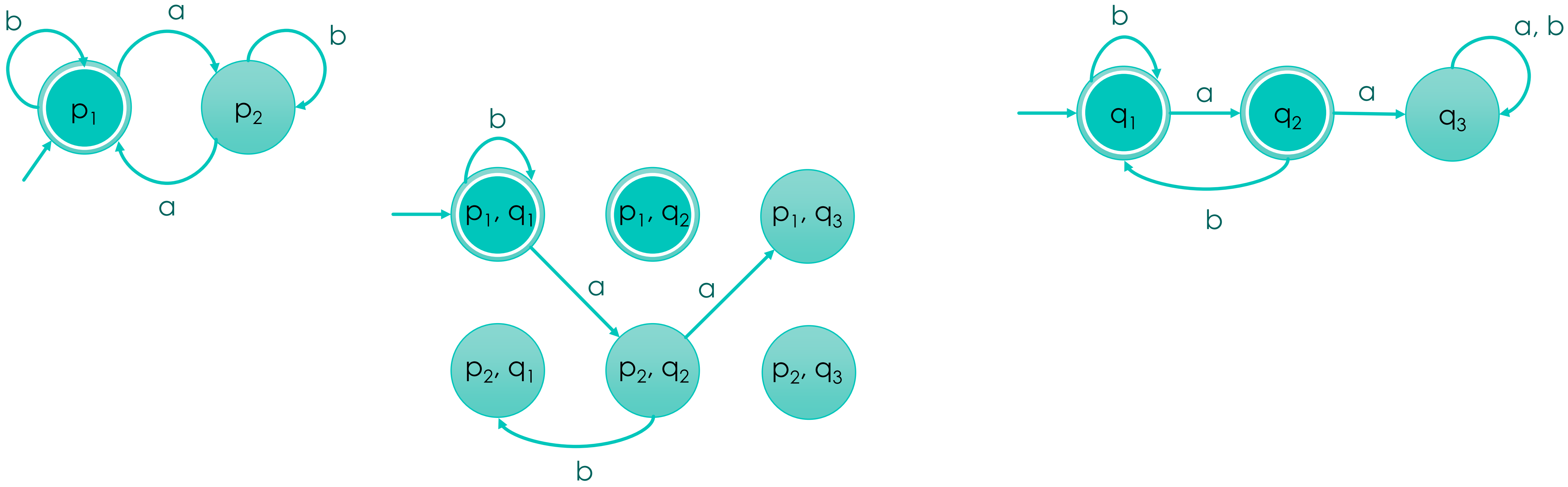






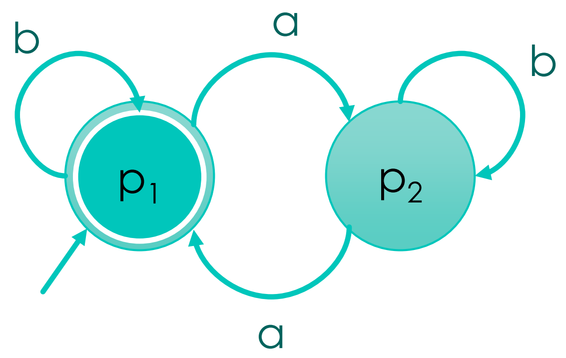
Example

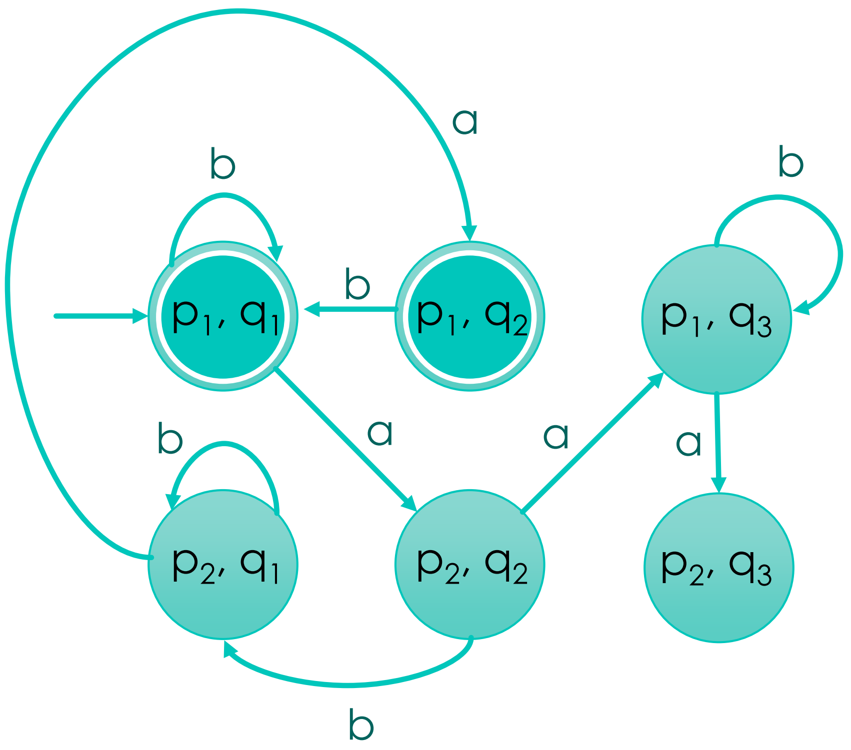
- The set of all strings over the alphabet $\Sigma = \{a, b\}$ such that in each string, the number of a's is even and doesn't include the substring aa.

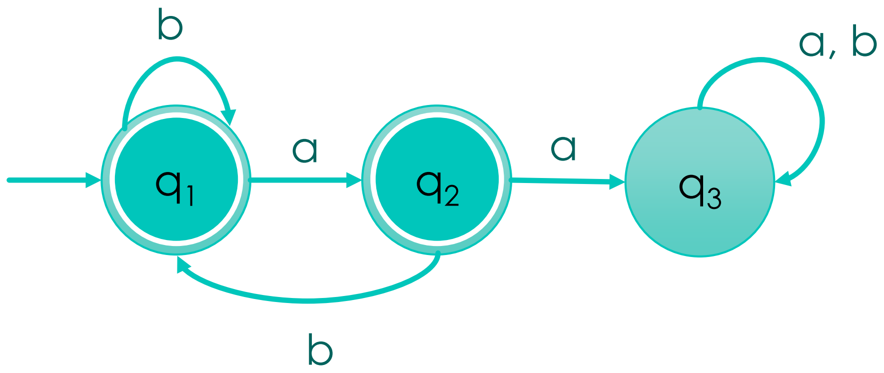


Example

- The set of all strings over the alphabet $\Sigma = \{a, b\}$ such that in each string, the number of a's is even and doesn't include the substring aa.

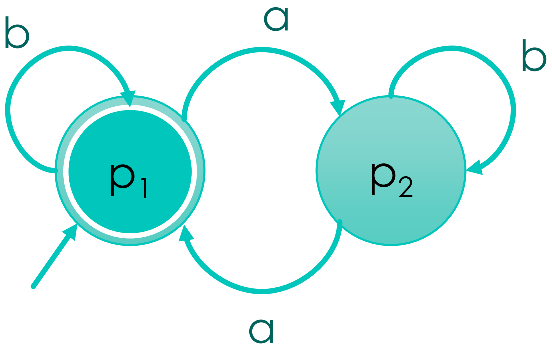


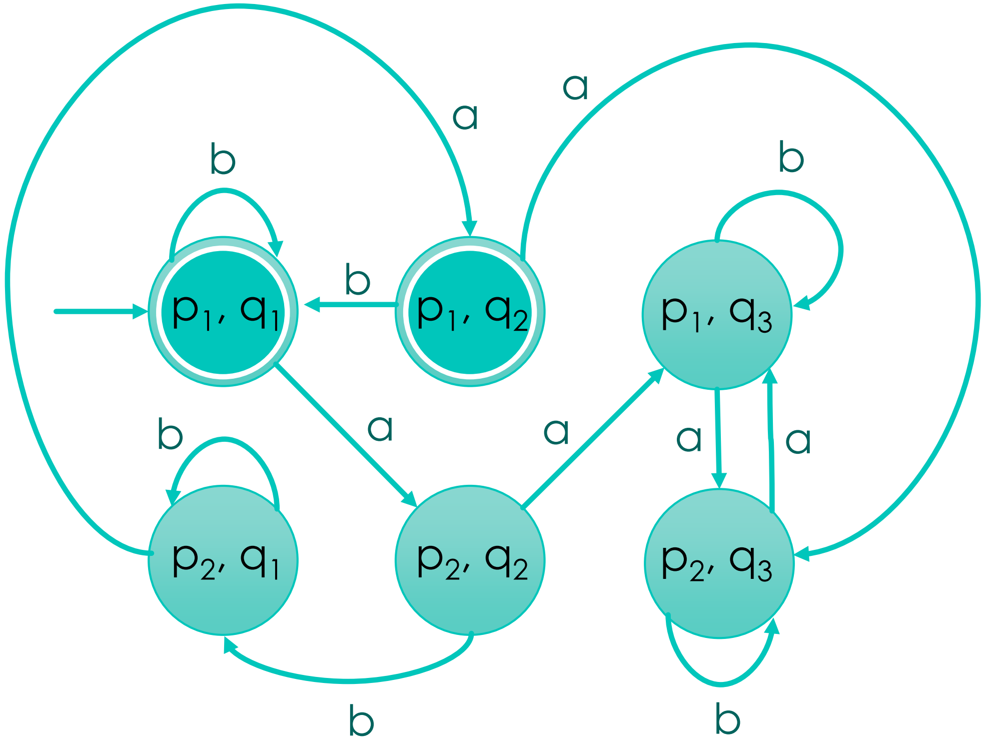


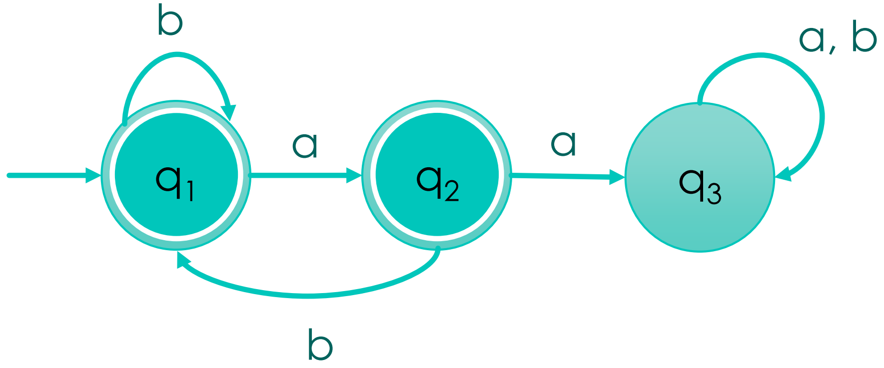


Example

- The set of all strings over the alphabet Σ = {a, b} such that in each string, the number of a’s is even and doesn’t include the substring aa.







Example

- The set of all strings over the alphabet $\Sigma = \{a, b\}$ such that each string starts with an a and contains at most one b.

On Blackboard!

Operations on Regular Languages : Complement

- Suppose $M = (Q, \Sigma, \delta, q_0, F)$ is a deterministic finite automaton.

- Let M’ be the deterministic finite automata $(Q', \Sigma, \delta', q_0', F')$ where

$Q' = Q$ $\quad$ $q_0' = q_0$ $\quad$ $F' = Q \setminus F$

$\delta' = \delta$

Then M recognizes $L(M)^c$

Example

- The set of all strings over the alphabet $\Sigma = \{a, b\}$ such that each string contains neither **ab** nor **ba**.

On Blackboard!

Operations on Regular Languages : Reversal

- Suppose $M = (Q, \Sigma, \delta, q_0, F)$ is a deterministic finite automaton and $a \in \Sigma$.

- Let M' be the finite automata $(Q', \Sigma, \delta', q_0', F')$ where

$Q' = Q \qquad q_0' = F \qquad F' = q_0$

$\delta'(q, a) = \{ p \in Q \mid \delta(p, a) = q \}$

Then M recognizes $L(M)^R$

The "transition function" may no longer be a function, and the start state may no longer be singular

So the FA definition needs to be changed, as we will see later

Operations on Regular Languages : Reversal

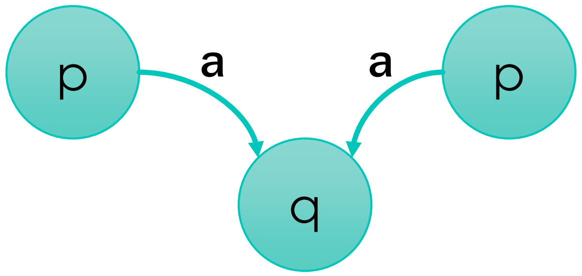
- Suppose $M = (Q, \Sigma, \delta, q_0, F)$ is a deterministic finite automata and $a \in \Sigma$.

- Let M' be the finite automata $(Q', \Sigma, \delta', q_0', F')$ where

  $Q' = Q$ $\quad\quad q_0' = F$ $\quad\quad F' = q_0$

  $\delta'(q, a) = \{ p \in Q \mid \delta(p, a) = q \}$

  Then M recognizes $L(M)^R$



Operations on Regular Languages : Reversal

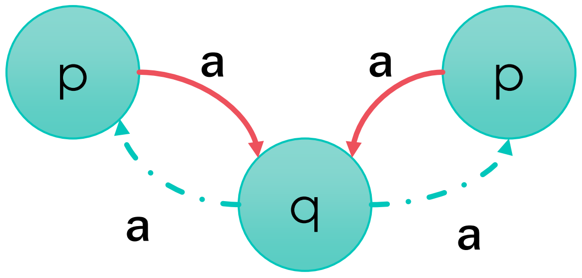
- Suppose $M = (Q, \Sigma, \delta, q_0, F)$ is a deterministic finite automata and $a \in \Sigma$.

- Let M' be the finite automata $(Q', \Sigma, \delta', q_0', F')$ where

  $Q' = Q$ $q_0' = F$ $F' = q_0$

  $\delta'(q, a) = \{ p \in Q \mid \delta(p, a) = q \}$

  Then M recognizes $L(M)^R$



Operations on Regular Languages : Difference

- Suppose $M_1 = (Q_1, \Sigma, \delta_1, q_0^1, F_1)$ and $M_2 = (Q_2, \Sigma, \delta_2, q_0^2, F_2)$ are two deterministic finite automata.

- $L(M_1) \setminus L(M_2) \quad = \quad L(M_1) \cap L(M_2)^c$

- We thereby conclude that the class of regular languages is also closed under the difference operator.

Example

- The set of all strings over the alphabet $\Sigma = \{a, b\}$ such that each block of five consecutive characters contains at least two **a**’s.

Try Solving it Yourself!

(We will go through the solution in the next session)

Inductive Definitions

- In logic, we very often define kinds of objects inductively, i.e., by specifying rules for what counts as an object of the kind to be defined which explain how to get new objects of that kind from old objects of that kind.

- For a simple example, consider strings of letters a, b, c, d, the symbol ◦, and brackets [and], such as "[[c ◦ d][", "[a[]◦]", "a" or "[[a ◦ b] ◦ d]".

Inductive Definitions

- You probably feel that there's something "wrong" with the first two strings: the brackets don't "balance" at all in the first, and you might feel that the "$\circ$" should "connect" expressions that themselves make sense.

- We would like to precisely specify what counts as a "nice term".

Inductive Definitions: Nice Terms

- The set of "nice terms" is inductively defined as follows:

1. Any letter a, b, c, d is a nice term.
2. If s_1 and s_2 are nice terms, then so is $[s_1 \circ s_2]$.
3. Nothing else is a nice term.

Extended Transition Function

- Suppose $M = (Q, \Sigma, \delta, q_0, F)$ be a deterministic finite automaton.

  We define the extended transition function δ^* as follows :

  (considering $q \in Q$, $\omega \in \Sigma^*$, $a \in \Sigma$)

1. Basis: $\delta^*(q, \varepsilon) = q$
2. Recursive (Inductive) Step: $\delta^*(q, \omega a) = \delta(\delta^*(q, \omega), a)$

Extended Transition Function

- **Basis:** $\delta^*(q, \varepsilon) = q$

- **Recursive (Inductive) Step:** $\delta^*(q, \omega a) = \delta(\delta^*(q, \omega), a)$

- **For example:**

$$\delta^*(q, aba) = \delta(\delta^*(q, ab), a) = \delta(\delta(\delta^*(q, a), b), a) = \delta(\delta(\delta(\delta^*(q, \varepsilon), a), b), a)$$

$$= \delta(\delta(\delta(q, a), b), a)$$

Operations on Regular Languages : Union

- Suppose $M_1 = (Q_1, \Sigma, \delta_1, q_0^1, F_1)$ and $M_2 = (Q_2, \Sigma, \delta_2, q_0^2, F_2)$ are two deterministic finite automata and $a \in \Sigma$.

- Let M be the finite automata $(Q, \Sigma, \delta, q_0, F)$ where

  $Q = Q_1 \times Q_2$ $\quad$ $q_0 = (q_0^1, q_0^2)$ $\quad$ $F = (F1 \times Q2) \cup (Q1 \times F2)$

  $\delta((q_1, q_2), a) = (\delta_1(q_1, a), \delta_2(q_2, a))$

  Then M recognizes $L(M_1) \cup L(M_2)$

Structural Induction

- The previous construction needs to be verified using structural induction.

- Mathematical Induction: Prove that a statement $P(n)$ holds for all $n \in \mathbb{N}$

- Structural Induction: Prove that a statement $P(x)$ holds for every x of some sort of recursively defined structure.

Structural Induction

- Structural induction works as follows:

- Base case: Prove P about the basis part in the recursive definition

- Inductive step: Assuming that P holds for sub-structures used in the recursive step of the definition, show that P holds for the next recursively constructed structure.

Structural Induction on the Size of the String

- We usually use structural induction on the size of the string

  based on the following recursive definition:

- Basis: $\varepsilon \in \Sigma^*$
- Recursive (Inductive) Step: for every $\omega \in \Sigma^*$ and $a \in \Sigma$, $\omega a \in \Sigma^*$

Correctness Proof

- M is the finite automata $(Q, \Sigma, \delta, q_0, F)$ where

  $Q = Q_1 \times Q_2$ $\quad$ $q_0 = (q_0^1, q_0^2)$ $\quad$ $F = (F1 \times Q2) \cup (Q1 \times F2)$

  $\delta((q_1, q_2), a) = (\delta_1(q_1, a), \delta_2(q_2, a))$

  We want to show that M recognizes $L(M_1) \cup L(M_2)$

We claim that $\quad \forall \omega \in \Sigma^*, \quad \delta^*((q_0^1, q_0^2), \omega) = (\delta_1^*(q_0^1, \omega), \delta_2^*(q_0^2, \omega))$

So every string $\omega \in \Sigma^*$ is accepted by M exactly when one of the following is true:

$$\left.\begin{array}{lll} \bullet\ \delta_1^*(q_0^1, \omega) \in F1 & \Longrightarrow & \delta^*(q_0, \omega) \in (F1 \times Q2) \\ \bullet\ \delta_2^*(q_0^2, \omega) \in F2 & \Longrightarrow & \delta^*(q_0, \omega) \in (Q1 \times F2) \end{array}\right\} \text{M recognizes } L(M_1) \cup L(M_2)$$

Correctness Proof

We claimed that $\forall \omega \in \Sigma^*, \quad \delta^*(q_0, \omega) = (\ \delta_1^*(q_0^1, \omega), \delta_2^*(q_0^2, \omega)\ )$

- Base case: $\delta^*(q_0, \varepsilon) = (\ \delta_1^*(q_0^1, \varepsilon), \delta_2^*(q_0^2, \varepsilon)\ )$

  Obvious based on $q_0 = (q_0^1, q_0^2)$

Correctness Proof

- **Inductive Step:**

assuming $\delta^*(q_0, \omega) = (\delta_1^*(q_0^1, \omega), \delta_2^*(q_0^2, \omega))$ is true for every ω of size n

$$\delta^*(q_0, \omega a)$$

$$= \delta\,(\delta^*(q_0, \omega)\,,\,a)$$

$$= \delta\,(\,(\delta_1^*(q_0^1, \omega), \delta_2^*(q_0^2, \omega)), a)$$

Correctness Proof

- **Inductive Step:**

$$\delta^*(q_0, \omega a)$$

$$= \delta\,(\delta^*(q_0, \omega)\,,\, a)$$

$$= \delta\,(\,(\,\delta_1^*(q_0^1, \omega),\, \delta_2^*(q_0^2, \omega)\,),\, a)$$

$$= (\,\delta_1\,(\delta_1^*(q_0^1, \omega),\, a),\, \delta_2\,(\delta_2^*(q_0^2, \omega),\, a)\,)$$

$$\delta\,(\,(q_1, q_2)\,,\, a\,) = (\,\delta_1\,(q_1, a),\, \delta_2\,(q_2, a)\,)$$

Correctness Proof

- **Inductive Step:**

$$\delta^*(q_0, \omega a)$$

$$= \delta\,(\delta^*(q_0, \omega)\,,\, a)$$

$$= \delta\,(\,(\,\delta_1^*(q_0^1, \omega), \delta_2^*(q_0^2, \omega)\,), a)$$

$$= (\,\delta_1\,(\delta_1^*(q_0^1, \omega), a), \delta_2\,(\delta_2^*(q_0^2, \omega), a)\,)$$

$$= (\,\delta_1^*(q_0^1, \omega a), \delta_2^*(q_0^2, \omega a)\,)$$

Done!

$$\boxed{\delta^*(q, \omega a) = \delta(\delta^*(q, \omega), a)}$$

Non-determinism

Next Set of Slides!