Introduction to Finite Automata

Languages

Deterministic Finite Automata

Representations of Automata

The Central Concepts of Automata Theory

Alphabet

An alphabet is a finite, non-empty set of symbols

- We use the symbol ∑ (sigma) to denote an alphabet
- Examples:
  - Binary: ∑ = {0,1}
  - All lower case letters: ∑ = {a,b,c,..z}
  - Alphanumeric: ∑ = {a-z, A-Z, 0-9}
  - DNA molecule letters: ∑ = {a,c,g,t}
  - …

Strings

A string or word is a finite sequence of symbols chosen from Σ

- ϵ stands for the *empty string* (string of length 0).
- Length of a string *w*, denoted by "|*w*|", is equal to the *number of (non- ε) characters in the string*
  - *E.g., x = 010100* *|x| = 6*
  - *x = 01 ε 0 ε 1 ε 00 ε* *|x| = ?*

- xy = concatenation of two strings x and y

Powers of an alphabet

Let Σ be an alphabet.

- Σ^{k} = the set of all strings of length k
- $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \ldots$
- $\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \ldots$

Example: Strings

- $\{0,1\}^* = \{\epsilon, 0, 1, 00, 01, 10, 11, 000, 001, \ldots\}$
- Subtlety: 0 as a string, 0 as a symbol look the same.
  - Context determines the type.

Languages

L is a said to be a language over alphabet Σ, only if $L \subseteq \Sigma^*$

➔ this is because Σ^* is the set of all strings (of all possible length including 0) over the given alphabet Σ

Examples:

1. Let L be *the* language of all strings consisting of n 0's followed by n 1's:

   L = {ε, 01, 0011, 000111,…}

2. Let L be *the* language of all strings of with equal number of 0's and 1's:

   L = {ε, 01, 10, 0011, 1100, 0101, 1010, 1001,…}

   $\longrightarrow$ Canonical ordering of strings in the language

Definition: Ø denotes the Empty language

- Let L = {ε}; Is L=Ø? $\boxed{\text{NO}}$

Languages

- Example: The set of strings of 0's and 1's with no two consecutive 1's.
- L = {ϵ, 0, 1, 00, 01, 10, 000, 001, 010, 100, 101, 0000, 0001, 0010, 0100, 0101, 1000, 1001, 1010, . . . }

Hmm… 1 of length 0, 2 of length 1, 3, of length 2, 5 of length 3, 8 of length 4. I wonder how many of length 5?

Set-Formers as a Way to Define Languages

It is common to describe a language using a "set-former":

$$\{w \mid \text{something about } w\}$$

This expression is read "the set of words w such that (whatever is said about w to the right of the vertical bar)." Examples are:

1. $\{w \mid w$ consists of an equal number of 0's and 1's $\}$.
2. $\{w \mid w$ is a binary integer that is prime $\}$.
3. $\{w \mid w$ is a syntactically correct C program $\}$.

It is also common to replace w by some expression with parameters and describe the strings in the language by stating conditions on the parameters. Here are some examples; the first with parameter n, the second with parameters i and j:

1. $\{0^n1^n \mid n \geq 1\}$. Read "the set of 0 to the n 1 to the n such that n is greater than or equal to 1," this language consists of the strings $\{01, 0011, 000111, \ldots\}$. Notice that, as with alphabets, we can raise a single symbol to a power n in order to represent n copies of that symbol.
2. $\{0^i1^j \mid 0 \leq i \leq j\}$. This language consists of strings with some 0's (possibly none) followed by at least as many 1's.

The Membership Problem

Given a string $w \in \Sigma^*$ and a language L over Σ, decide whether or not $w \in L$.

Example:

Let w = 100011

Q) Is $w \in$ the language of strings with equal number of 0s and 1s?

Deterministic Finite Automata (DFA)

- A formalism for defining languages, consisting of:
  1. A finite set of *states* (Q, typically).
  2. An *input alphabet* (Σ, typically).
  3. A *transition function* (δ, typically).

     $\delta: Q \times \Sigma \rightarrow Q$

     $\delta(q, a) \rightarrow p$
  4. A *start state* (q_0, in Q, typically).
  5. A set of *final states* ($F \subseteq Q$, typically).
     - "Final" and "accepting" are synonyms.
- DFA in five-tuple notation:

$$A=(Q, \Sigma, \delta, q_0, F)$$

The Transition Function

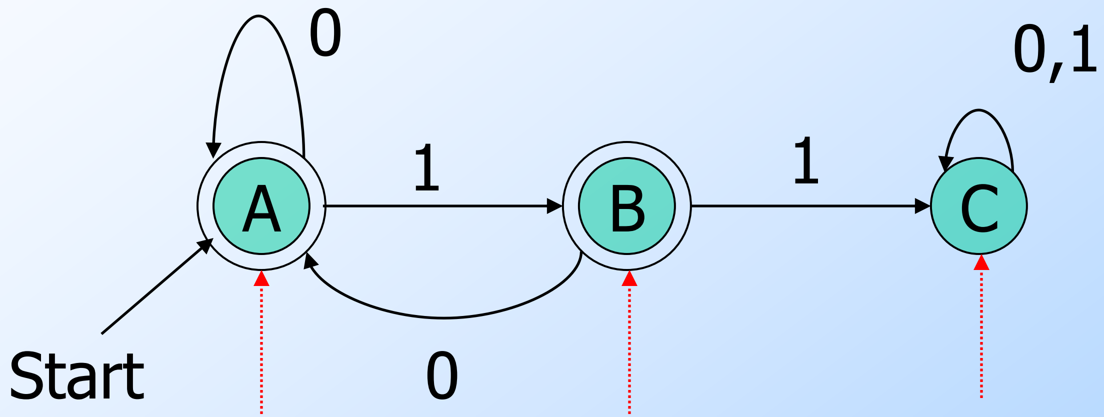
- Takes two arguments: a state and an input symbol.
- $\delta(q, a)$ = the state that the DFA goes to when it is in state *q* and input *a* is received.

Graph Representation of DFA's

- Nodes = states.
- Arcs represent transition function.
  - Arc from state p to state q labeled by all those input symbols that have transitions from p to q.
- Arrow labeled "Start" to the start state.
- Final states indicated by double circles.

Example: Graph of a DFA

Accepts all strings without two consecutive 1's.



Previous string OK, does not end in 1.

Previous String OK, ends in a single 1.

Consecutive 1's have been seen.

$$A = (\{A,B,C\}, \{0,1\}, \delta, A, \{A,B\})$$

Alternative Representation: Transition Table

Final states starred

Columns = input symbols

	0	1
→ * A	A	B
* B	A	C
C	C	C

Arrow for start state

Rows = states

Extended Transition Function

- We describe the effect of a string of inputs on a DFA by extending δ to a state and a string.
- Induction on length of string.
- Basis: $\delta(q, \epsilon) = q$
- Induction: $\delta(q,wa) = \delta(\delta(q,w),a)$
  - w is a string; a is an input symbol.

Extended δ: Intuition

- Convention:
  - … w, x, y, z are strings.
  - a, b, c,… are single symbols.
- Extended δ is computed for state q and inputs $a_1a_2 \ldots a_n$ by following a path in the transition graph, starting at q and selecting the arcs with labels $a_1, a_2, \ldots, a_n$ in turn.

Example: Extended Delta

	0	1
A	A	B
B	A	C
C	C	C

$\delta(B,011) = \delta(\delta(B,01),1) = \delta(\delta(\delta(B,0),1),1) =$

$\delta(\delta(A,1),1) = \delta(B,1) = C$

Delta-hat

- Some people denote the extended δ with a "hat" to distinguish it from δ itself.
- Not needed, because both agree when the string is a single symbol.
- $\hat{\delta}(q, a) = \delta(\hat{\delta}(q, \epsilon), a) = \delta(q, a)$

Extended deltas

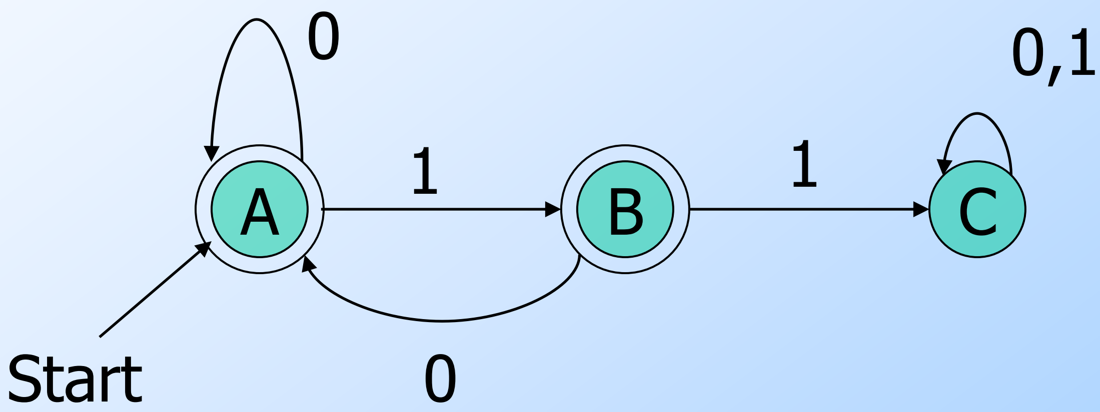
Language of a DFA

- Automata of all kinds define languages.
- If A is an automaton, L(A) is its language.
- For a DFA A, L(A) is the set of strings labeling paths from the start state to a final state.
- Formally: L(A) = the set of strings w such that $\delta(q_0, w)$ is in F.

$$L(A) = \{w \mid \hat{\delta}(q_0, w) \text{ is in } F\}$$

Example: String in a Language

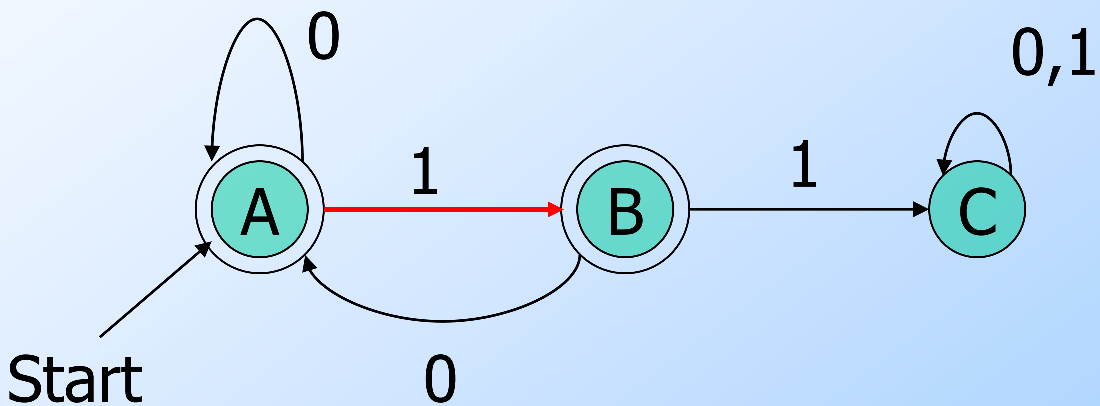
String 101 is in the language of the DFA below.
Start at A.



Example: String in a Language

String 101 is in the language of the DFA below.

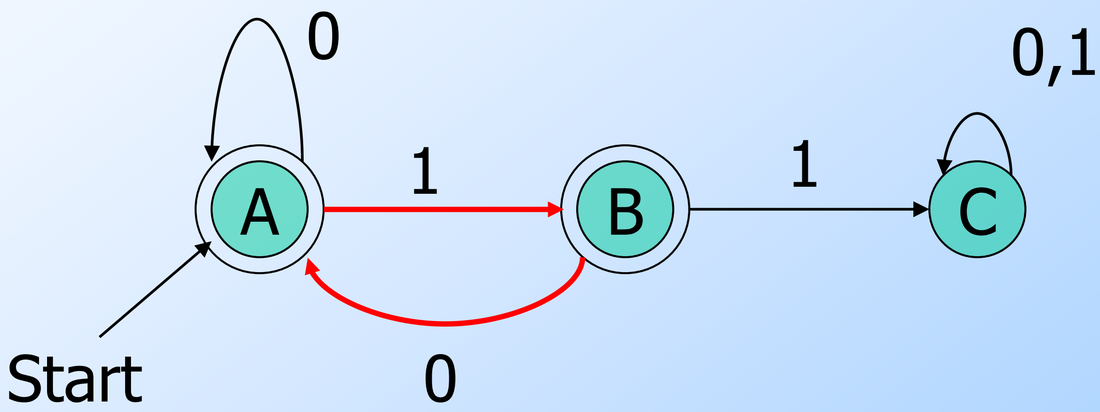
Follow arc labeled 1.



Example: String in a Language

String 101 is in the language of the DFA below.

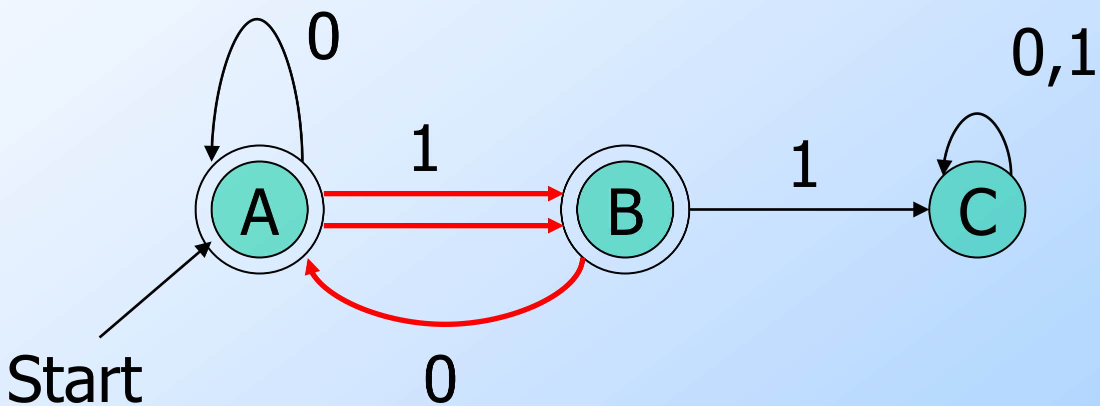
Then arc labeled 0 from current state B.



Example: String in a Language

String 101 is in the language of the DFA below.

Finally arc labeled 1 from current state A. Result is an accepting state, so 101 is in the language.



Example – Concluded

- The language of our example DFA is:

{w | w is in {0,1}* and w does not have two consecutive 1's}

Such that…

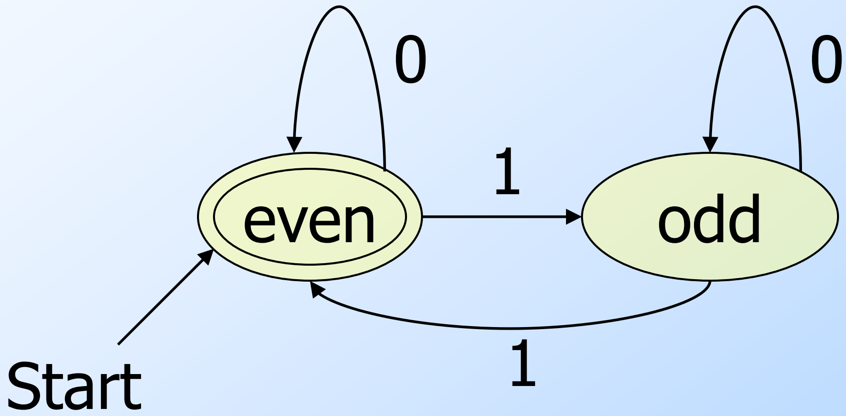
These conditions about w are true.

Read a *set former* as "The set of strings w…

Example 4: Set of all strings that contain the string aabb in it

Example 5: An Even Number of 1's

Example 5: An Even Number of 1's

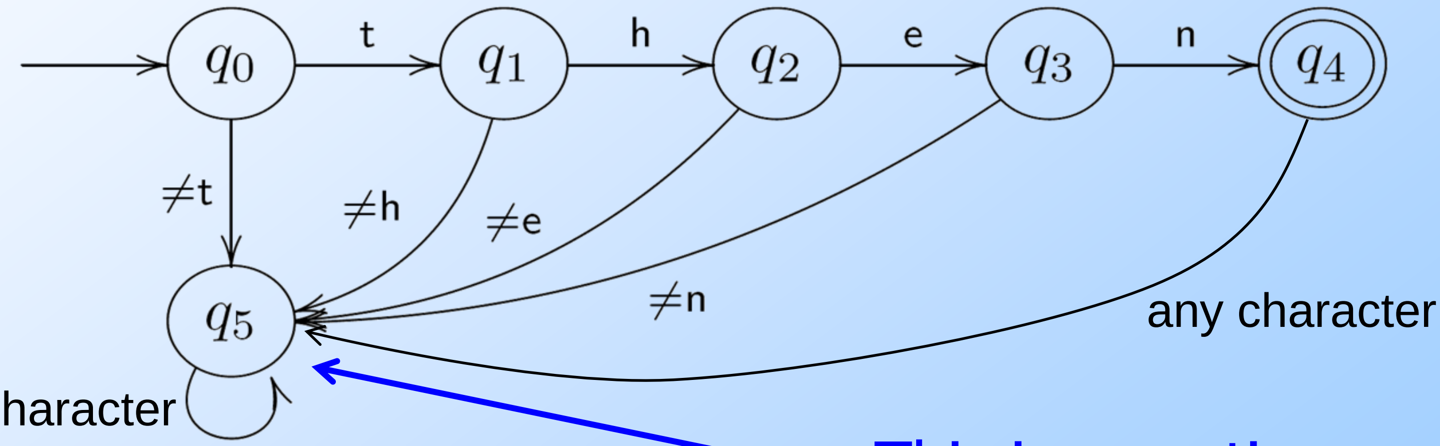


Password/Keyword Example

It reads the password and accepts it if it is “then”

Password/Keyword Example

It reads the word and accepts it if it stops in an accepting state

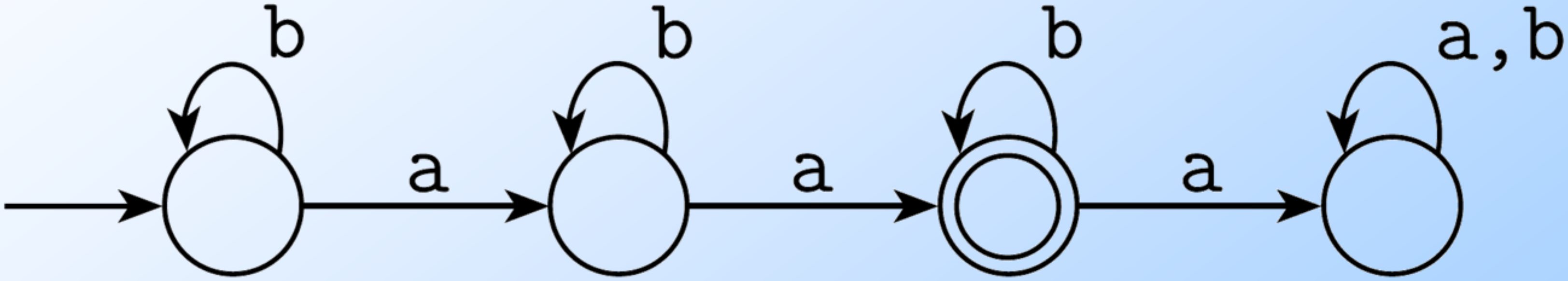


Only the word then is accepted

This is sometimes called a **dead** state.

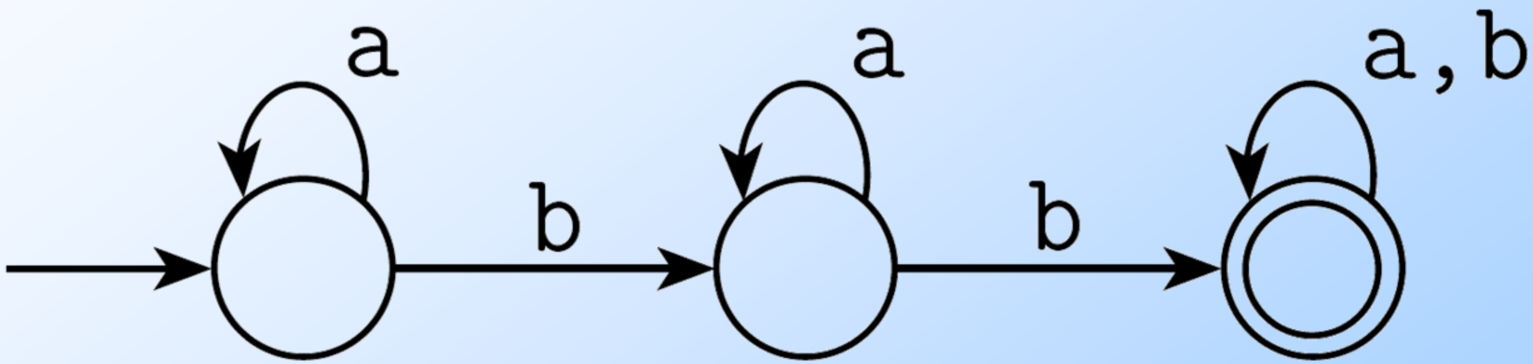
Exactly Two a’s

Exactly Two a’s



At Least Two b’s

At Least Two b's

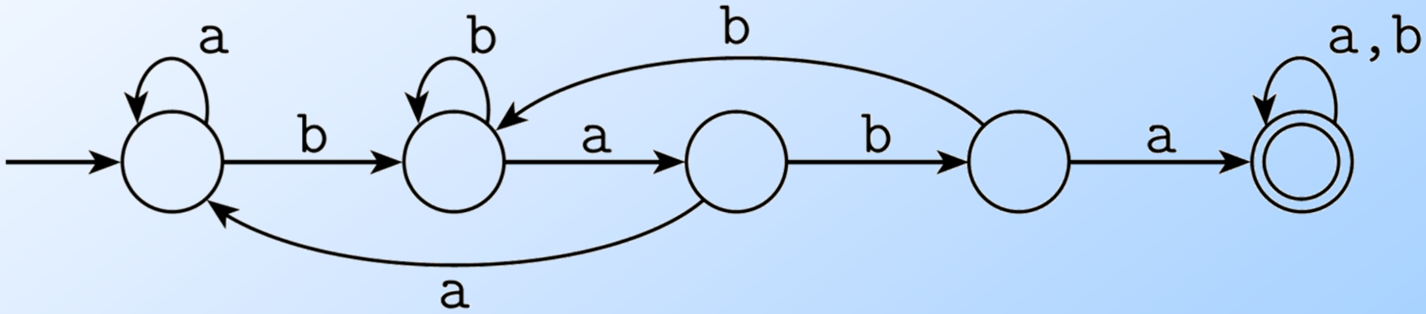


Containing Substrings or Not

- Contains baba:

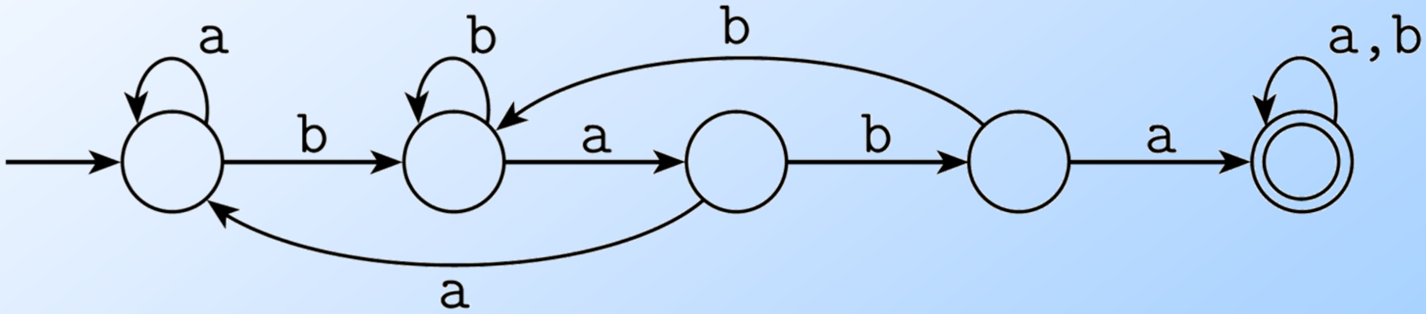
Containing Substrings or Not

- Contains baba:



Containing Substrings or Not

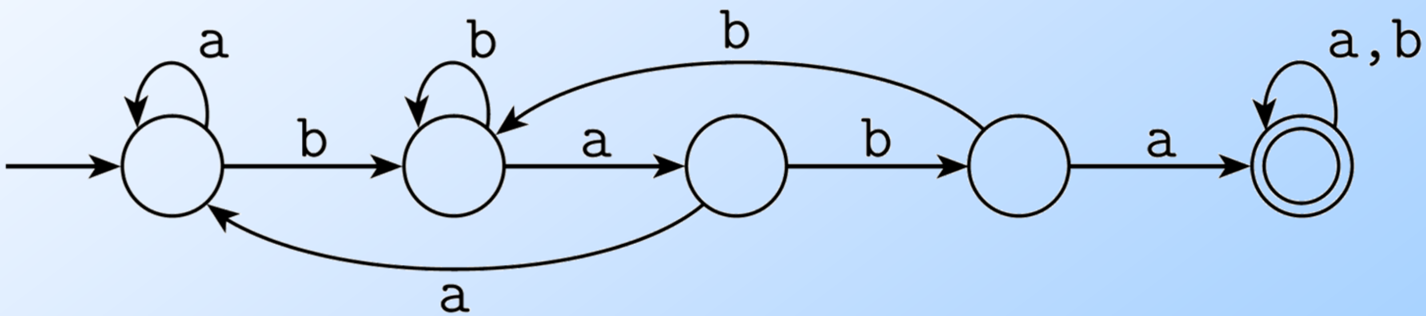
- Contains baba:



- Does not contain baba:

Containing Substrings or Not

- Contains baba:



- Does not contain baba: