2 - Regular Languages and Finite State Automata

2.1 - Regular Expressions and Regular Languages

- **Definition :** The family of **regular languages** over an alphabet Σ is the least family of languages over Σ with the following properties :
    - ✓ The empty set ø is a regular language.
    - ✓ The singleton {ε} is a regular language.
    - ✓ For every element a from Σ, the singleton {a} is a regular language.
    - ✓ If L is a regular language , its **Kleene closure** L^* is a regular language.
    - ✓ If L_1 and L_2 are regular languages, their **concatenation** $L_1 . L_2$ and their **union** $L_1 \cup L_2$ are regular languages.
- A **regular expression** is the expression of a regular language from elementary languages with the operations of Kleene closure, concatenation and disjunction.
- **Theorem :** the class of regular languages over an alphabet Σ is closed under **intersection**, **complementation** and **reverse operation**.

2.1 - Regular expressions an Regular languages

1. Determine regular expressions representing the following languages :

   a) The car numbers in France (from one to four digits followed by two or three capital letters and a departmental code from 01 to 98),

   b) The decimal numbers in scientific notation (a positive or negative mantisse with one non null digit before the point followed by the symbol E and a positive or negative exponent (except 0),

   c) All strings over the alphabet *{a,b, c}* that include one substring *ab* and one substring *ac* at least,

   d) All strings over the alphabet *{a,b,c}* that do not include the substring *ac*.

2.2 - Deterministic Finite State Automata

- A **Finite State Automaton** (FSA) is a MT with bounded transitions :
  - ✓ The movements of the pointer are only forward movements.
  - ✓ The tape can only be read.

  To summarize, transitions have the form: (q_1, a, a, R, q_2)

✓ **Definition :** a Deterministic Finite State Automaton is a 5-uple $(Q, \Sigma, q_0, F, \tau)$ such that :

- ✓ Q is a finite set of states of the control unit.
- ✓ Σ is a finite input tape alphabet of symbols.
- ✓ q_0 is a particular element of Q, the start state of the automaton.
- ✓ F is a subset of Q, the accepting states of the automaton.
- ✓ τ is a transition function which associates a source state *q* from Q and an input symbol *a* from Σ to a target state from Q denoted $\tau(q,a)$.

2.2 - Deterministic Finite State Automata

- The **principle of a Finite State Automaton** :

  ✓ **Initialisation**: Initially, a string *s* from Σ^* is written on the tape from its beginning. The other positions are filled with the blank symbol $\perp$, which does not belong to Σ. The pointer indicates the beginning of the tape.
  The control unit is initially in the state q_0.

  ✓ **Transition step** : a transition can occur if the control unit is in a state q_1, if the pointer indicates the symbol *a* on the tape, and if the transition function τ is defined for (q_1,a). In this case, the pointer moves to the next position. Finally, the control unit moves to the state $\tau(q_1,a)$.

  ✓ **Termination** :If in a configuration of the FSA, no transition is possible, the FSA stops. At this moment, if the unit control is in accepting state and if the pointer indicates the end of *s*, the **input word** *s* is said to be **accepted** by the FSA.

2.2 - Deterministic Finite State Automata

- **Definition :** a **computation** on an automaton is a (possibly empty) sequence of transitions in the form : $q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} \dots \xrightarrow{a_n} q_n$ We denote : $q_0 \xrightarrow{a_1 a_2 \cdots a_n} {}^{*} q_n$

- The string $a_1 a_2 \dots a_n$ is **recognized** by the automaton if q_n is an accepting state.

- The language recognized by the automaton is the set of all strings recognized by the automaton.

2.2 - Deterministic Finite State Automata

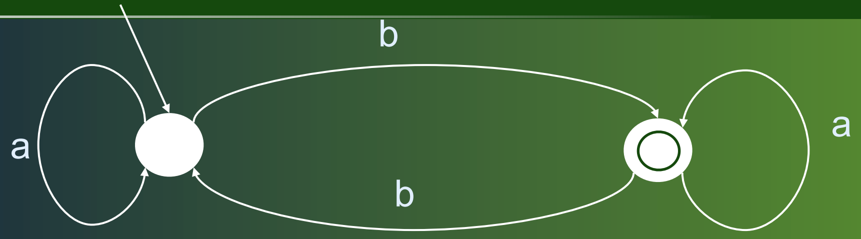
- **Algorithm of recognition**

```
function D-RECOGNIZE (tape, automaton)
index ← beginning_of (tape)
current-state ← initial_state_of (automaton)
loop
    if index = end_of_input (tape) then
        if current-state ∈ accept_states (automaton) then
            return accept
        else
            return reject
        endif
    elseif transition_table (automaton)[current-state, tape[index]]) = empty then
        return reject
    else
        current-state ← transition_table (automaton)[current-state, tape[index]])
        index ← index + 1
    endif
endloop
```

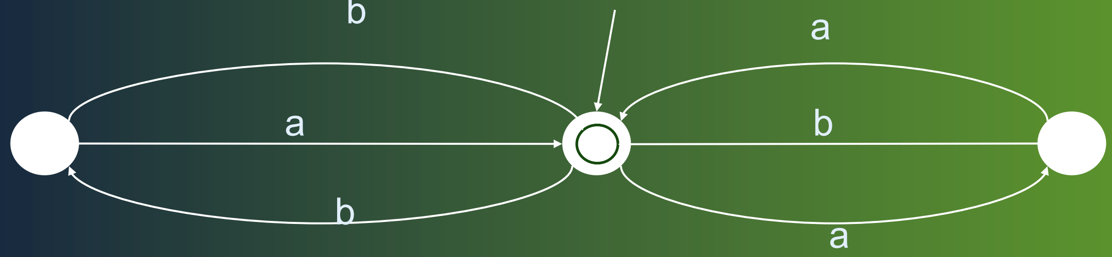
2.2 - Deterministic Finite State Automata

1. Describe the languages recognized by the following FSA:

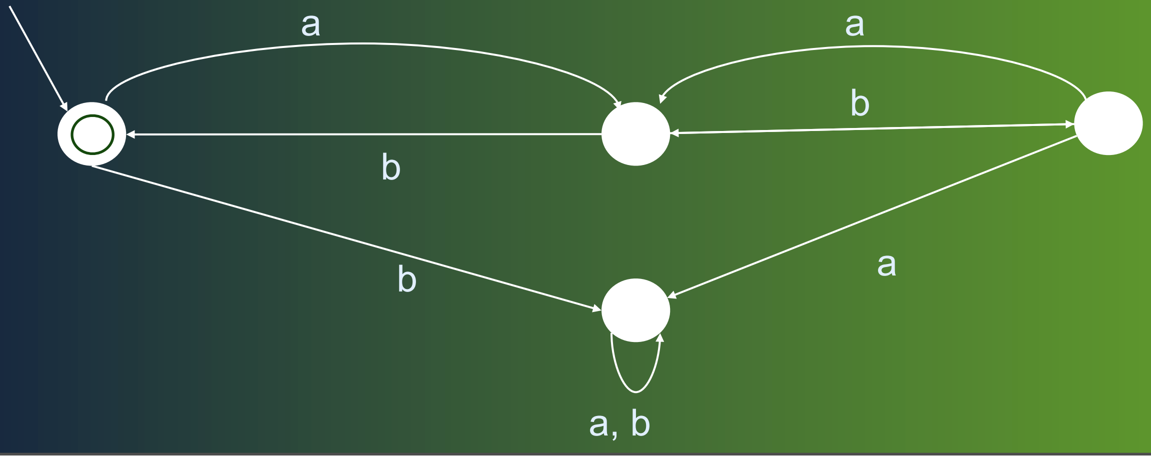
a)



b)



c)



2.2 - Deterministic Finite State Automata

2. Determine a regular expression and an automaton recognizing the following languages :

a) The set of natural numbers in base 10.

b) $\{0^n 1^m \mid n, m \in \mathbb{N}\}$.

c) The set of strings of digits including 11. The alphabet is {0,1,…,9}.

d) The set of dates in the format *day/month,* where *day* and *month* are natural numbers with two digits; *month* is between 01 and 12 and *day* must be compatible with *month* (for instance, if *month* = 02 then 01 ≤ *day* ≤ 29).

e) The set of floating numbers. A floating number is written as a mantisse followed by an exponent. The mantisse is composed of a facultative sign followed by a whole part and possibly a decimal which starts with a point. The exponent is composed of the letter E followed by a facultative sign and a natural number.

f) The language of comments in computer programs. A comment is a sequence of characters between /* and */. Such sequence cannot include /* and */ except if these are preceded by the escape character %.

2.3 - Non Deterministic Finite State Automata

- **Definition :** a Non Deterministic Finite State Automaton over an alphabet Σ, is a 5-uple $(Q, \Sigma, q_0, F, \tau)$ such that :

  ✓ Q is a finite set of states of the control unit.

  ✓ $\Sigma \cup \{\varepsilon\}$ is a finite input alphabet of symbols. The special symbol ε is the empty word, that is the unit for concatenation of words from Σ^* .

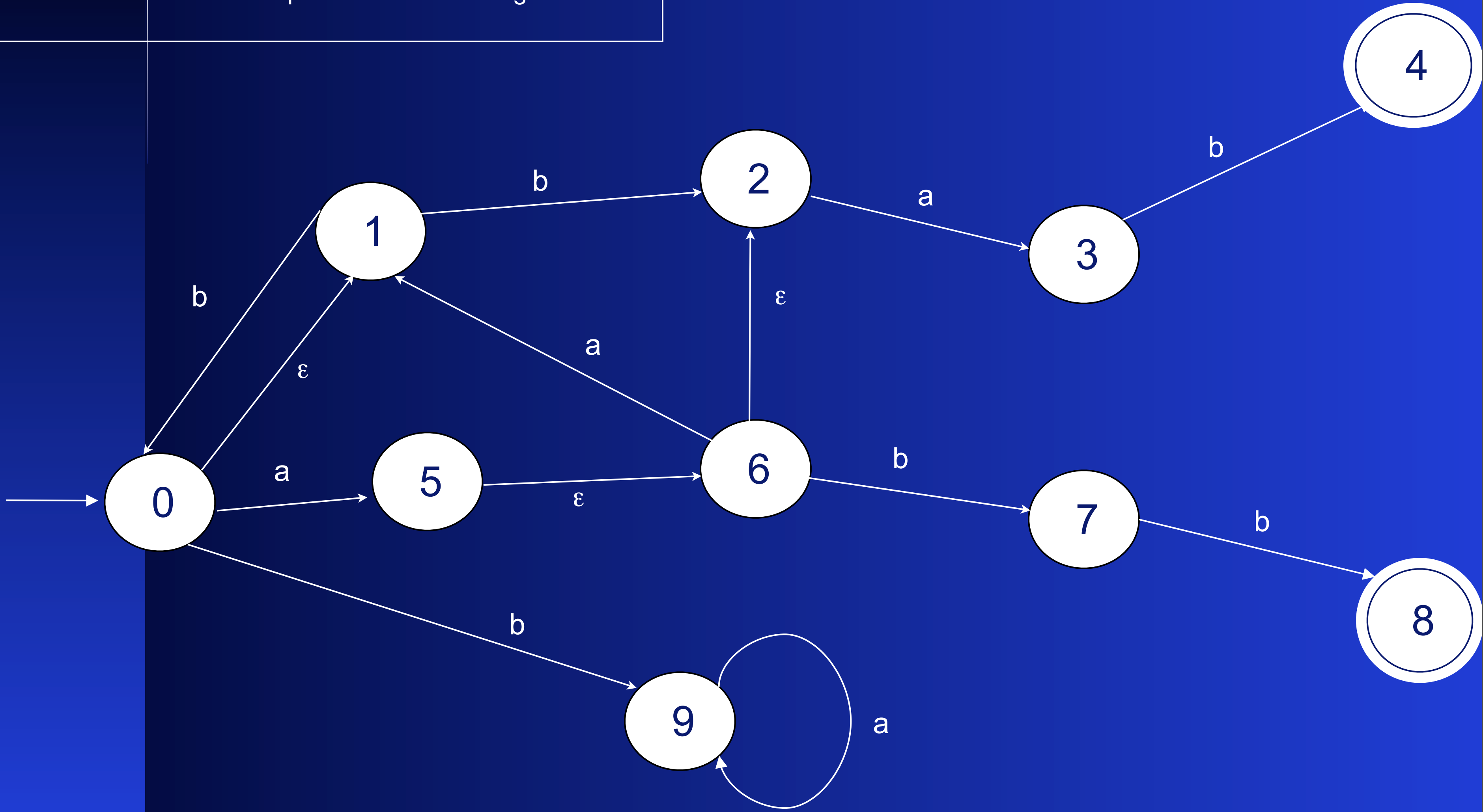
  ✓ q_0 is a particular element of Q, the start state of the automaton.

  ✓ F is a subset of Q, the accepting states of the automaton.

  ✓ τ is a transition relation which associates a source state q_1 and an input symbol *a* with a target state q_2.This is denoted: $\tau(q_1,a, q_2)$.

2.4 - Determinization of Finite State Automata

Automat0 : an example of NFSA to recognized



2.3 - Non Deterministic Finite State Automata

An algorithm of recognition :

```
function ND-RECOGNIZE (tape, automaton)
agenda ← { }
index ← beginning_of (tape)
current-state ← initial_state_of (automaton)
loop
        if index = end_of_input (tape)  and current-state ∈ accept_states (automaton)  then
                return accept
        else
                forall state  ∈  transition_table (automaton)[current-state, tape[index]])
                        agenda ← agenda ∪ {(state, index + 1)}
                endforall
                forall state  ∈  transition_table (automaton)[current-state, ε])
                        agenda ← agenda ∪ {(state, index )}
                endforall
                if agenda = { } then
                        return reject
                else
                        (current-state, index) ← choose_a_new_element(agenda)
                endif
        endif
endloop
```

2.4 - Determinization of Finite State Automata

- The complexity of the recognition problem with a given DFSA is **linear-time** in the length of the input word and it does not depend on the size of the automaton.

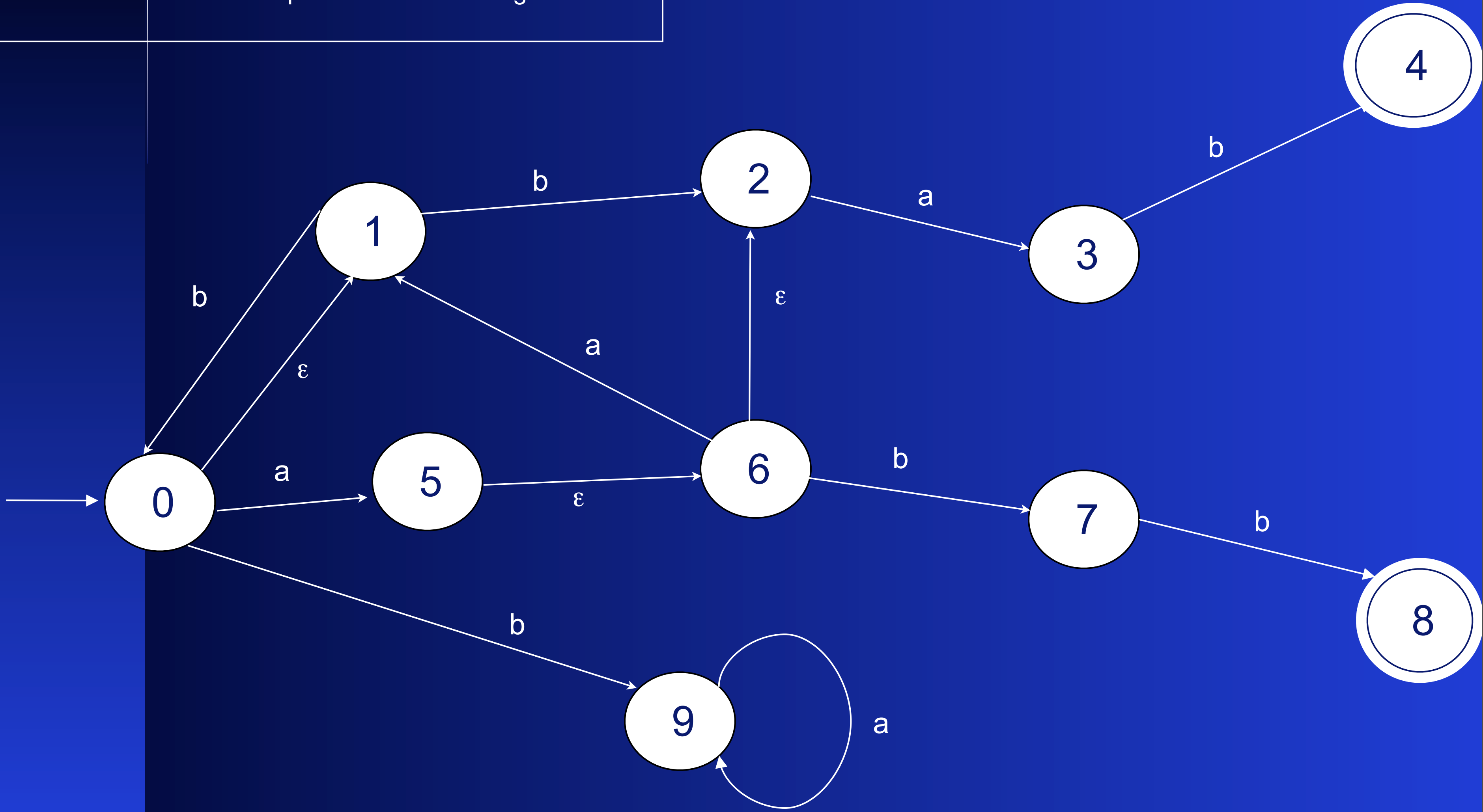
- **Theorem** : for any NFSA, there exists an equivalent DFSA, that is an automaton recognizing the same language

2.4 - Determinization of Finite State Automata

- The principle of transforming a NFSA into a DFSA is the following:
  - *any state of the DFSA is the set of all NFSA states recognizing the same word of* Σ^**;*
  - *the initial state of the DFSA is the set of all NFSA states recognizing the empty word;*
  - *the final states of the DFSA are all states containing a NFSA final state;*
  - *if two DFSA states* s_1 *and* s_2 *are respectively associated with* w *and* w.a *words from* Σ^**, there is a transition from* s_1 *to* s_2*.*
- The DFSA is built state by state recursively from its initial state.

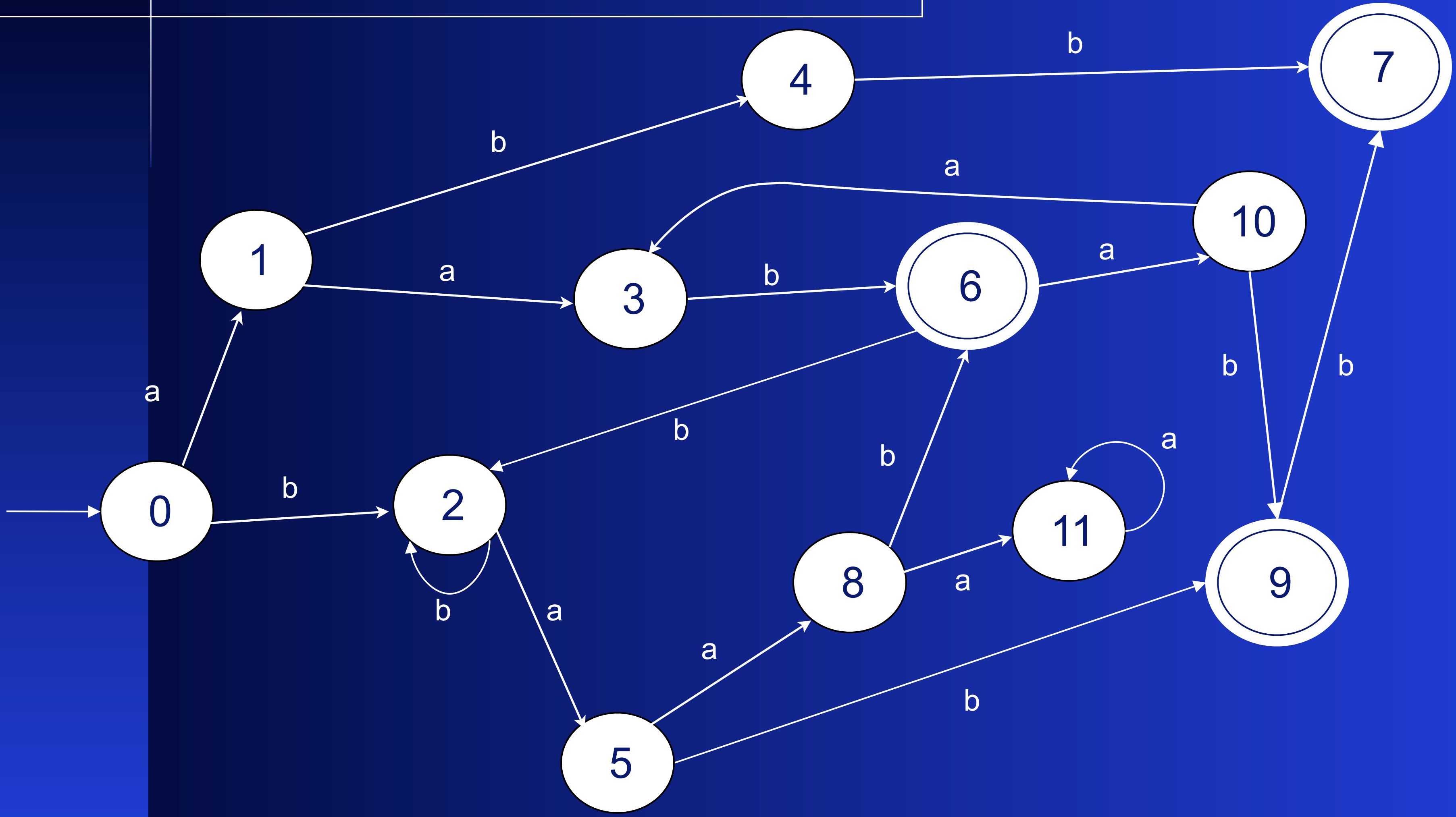
2.4 - Determinization of Finite State Automata

Automat0 : an example of NFSA to recognized



2.4 - Determinization of Finite State Automata

Automat1: a DFSA which results from determinizing *Automat0*.



2.5 - Minimization of Finite State Automata

- If we change the initial state of a FSA with any state *q*, the language recognized by the new FSA is **the language recognized from state** *q* in the former FSA.

- Every DFSA can be **minimized** by identifying its **interchangeable** states, the states from which the same languages are recognized.

2.5 - Minimization of Finite State Automata

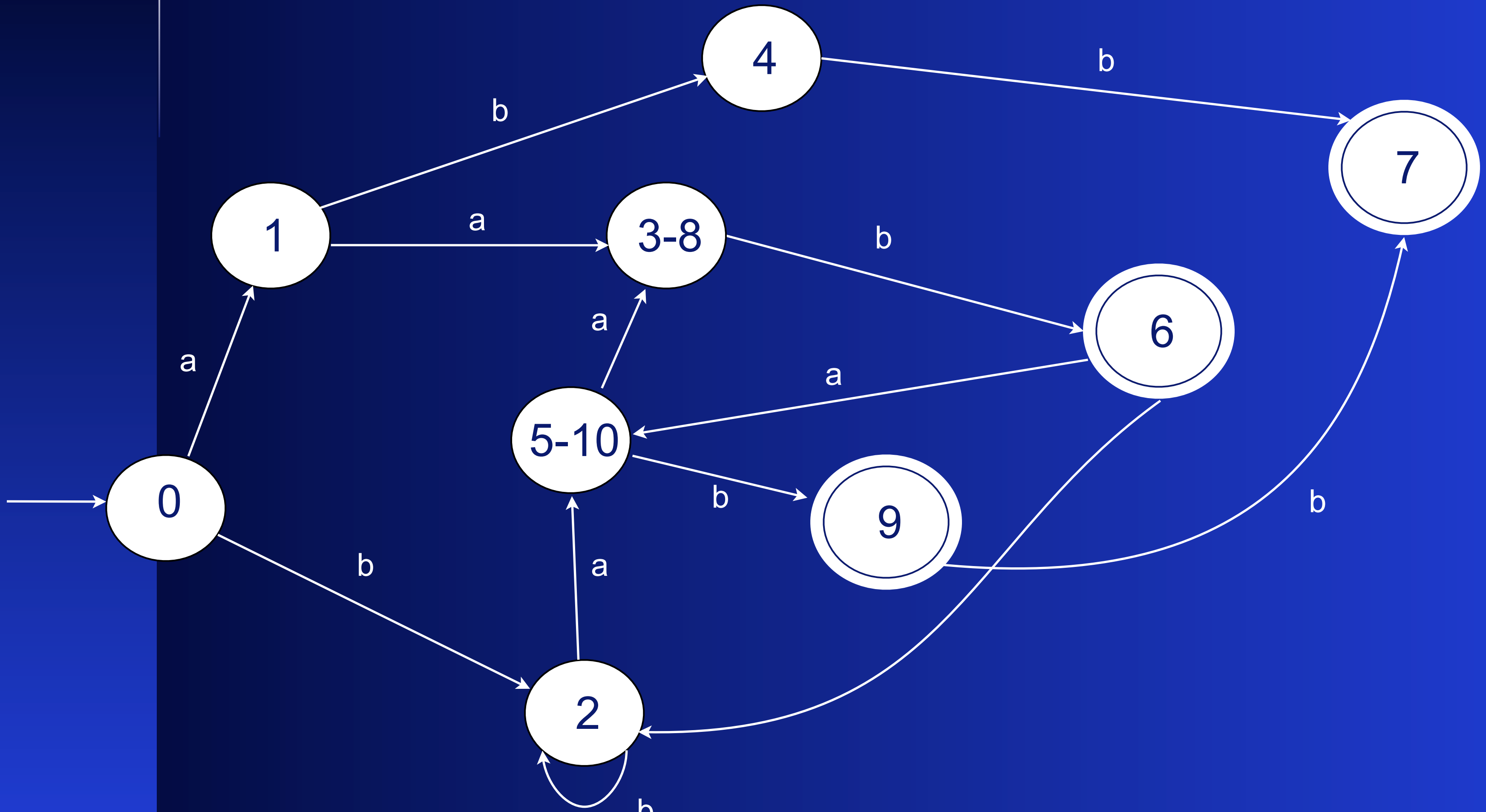
- For every DFSA A_1, the minimal automaton A_2 equivalent to A_1 is defined as follows:
    - *The states of A_2 are the equivalence classes of A_1 interchangeable states;*
    - *The initial state of A_2 is the equivalence class of the A_1 initial state;*
    - *The final states of A_2 are the equivalence classes of the A_1 final states;*
    - *For any transition in A_1 on a symbol a from a state s_1 to a state s_2, there is a corresponding transition on a in A_2 from the equivalence class of s_1 to the equivalence class of s_2*

2.5 - Minimization of Finite State Automata

- The minimal automaton is built recursively by approximating the interchangeability classes step by step, according to Moore's algorithm (1956):
  - The partition between equivalence classes is initialized with two classes : the class of final states and the class of non final states plus the garbage state.
  - Then, the partition is refined step by step. At each step, each class is partitioned according to the input transitions from other classes.
  - The process stops when a fix point is reached.
- The complexity of the algorithm is quadratic in time in the size of the automaton.
- There is an exponential algorithm which consists in determinizing the reverse automaton.

2.5 - Minimization of Finite State Automata

Automat2: resulting from the minimization of *Automat1*.



2.5 - Minimization of Finite State Automata

1. Consider the following transition table defining a DFSA with the initial state q_0 and the unique accepting state q_2. Minimize this automaton.

	q_0	q_1	q_2	q_3	q_4	q_5	q_6	q_7
0	q_1	q_6	q_0	q_2	q_7	q_2	q_6	q_6
1	q_5	q_2	q_2	q_6	q_5	q_6	q_4	q_2

2. Consider the following transition table defining a NFSA with the initial state q_0 and the unique accepting state q_9. Determinize and minimize this automaton.

	q_0	q_1	q_2	q_3	q_4	q_5	q_6	q_7	q_8	q_9
ε	q_2		q_0				q_9			q_6
0	q_1	q_3	q_4			q_6	q_7	q_9		
1		q_2	q_5	q_2	q_6		q_8		q_9	

2.6 - Finite State Automata and Regular Languages

- FSA can be combined to recognize the languages resulting from the following operations on their associated languages: **union**, **concatenation**, **Kleene closure**, **intersection** , **complementation**.

- **Theorem (Kleene)**: the class of the regular languages identifies with the class of the languages recognized by FSA.

    - First direction : any regular language is recognized by a FSA.

        Proof by induction on the structure of the regular expression defining the language; the induction step uses the previous theorems for the operations of union, concatenation and Kleene closure (Thompson method).

    - Second direction : any language recognized by a FSA is regular.

        Proof by addition of a new initial state and a new final state and by removing all intermediate states step by step (Brzozowski and Mc Cluskey algorithm).

2.6 - Finite State Automata and Regular Languages

- **Pumping Lemma** : let L be an infinite regular language. Then, there are strings x, y and z, such that $y \neq \varepsilon$ and $xy^nz \in L$ for any $n \geq 0$.

- The Pumping Lemma is used to prove that a language is not regular.

2.6 - Finite State Automata and Regular Languages

1. Build the minimal automaton recognizing the language defined by the regular expression (a(ca* | b*c)*bc*)*

2. Build a regular expression defining the language recognized by the automaton with the following transition table (the initial state is q_0 and the final states are q_2 and q_4):

	q_0	q_1	q_2	q_3	q_4	q_5
a	q_1	q_1	q_2	q_4		
b	q_2	q_3	q_3	q_3		
c		q_4	q_5			

3. Determine a regular expression and an automaton recognizing the following languages (determinize and minimize the automaton) :

a) The set of strings including 001 and 11 (the alphabet is {0,1}).

b) Date expressions in French like *"13 novembre 2005"*

c) Sentences in French that do not include the word *"moto".* (a sentence is sequence of words separated with one space)

2.6 - Finite State Automata and Regular Languages

4. Determine if the following languages are regular :

   a) The set of the words defined in the 2001 edition of the dictionary Larousse over the usual French alphabet plus hyphen.

   b) $\{(ab)^n(ba)^n \mid n \in \mathbb{N}\}$

   c) $\{(ab)^n a(ba)^n \mid n \in \mathbb{N}\}$

   d) The arithmetic expressions on natural numbers in infixed notation (the usual notation).

   e) The arithmetic expressions on natural numbers in polish notation (for instance, *(2-3)x4+10x5* is written + *x - 2 3 4 x 10 5*).

5. Three missionaries and three cannibals want to cross a river. There is only one boat which can carry two persons at most. A missionary cannot use the boat alone. A soon as there are more cannibals than missionaries at some place, the cannibals will eat the missionaries. Is there a way for the six persons to cross the river without being eaten.
   Hints : use an automaton where states represent the possible distributions of missionaries and cannibals between the two sides of the river and transitions represent the crossings of the river.

2.6 - Finite State Automata and Regular Automata

6. **The blind barman with boxing gloves**

A barman and a client play at the following game :
The barman blindfolds himself and gets boxing gloves. Then, he cannot see and determine if a glass is up or down. In front of the barman, there is a salver with four glasses put in square. These glasses can be up or down. Their direction is chosen initially by the client and unknown by the barman.
The barman can repeat the following operation ten times at most : he indicates the glasses that he wants to turn over; the client turns the salver and then, the barman turns the glasses over as he has indicated. If the glasses are all in the same direction, the barman wins.

a) Give an automaton the states of which are the different configurations of the salver and the transitions are the possible changes of configuration, their input symbols representing the indications of the barman.

b) From this automaton, deduce another automaton in which the accepting states are the configurations that are winning for the client.

c) Give an automaton that guarantees the victory to the barman whichever the choice of the client is.

d) Solve the problem again with three glasses in triangle and then with five glasses in pentagon.