CS311: Computational Theory

Lecture 3: Regular Languages – Ch 1 – Cont'd

Lecture Learning Objectives

1. *Understand Regular Languages and Regular Expressions*
2. *Express Regular Languages using DFAs, and NFAs.*
3. *Convert among equivalently powerful notations for a language, including among DFAs, NFAs, and regular expressions.*

Regular Languages

1. *The language recognized by a finite automaton M is denoted by L(M).*
2. *A regular language is a language for which there exists a recognizing finite automaton.*

Two DFA Questions

1. *Given the description of a finite automaton $M = (Q, \Sigma, \delta, q, F)$, what is the language $L(M)$ that it recognizes?*
2. *In general, what kind of languages can be recognized by finite automata? (What are the regular languages?)*

Complement of a Regular Language

1. *Swap the accepting and non-accept states of M to get M'.*
2. *The complement of a regular language is regular.*

FORMAL DEFINITION OF A REGULAR EXPRESSION

*Say that R is a **regular expression** if R is:*

1. *a for some a in the alphabet Σ,*
2. *ε,*
3. *∅,*

1. a represent the languages {a}

2. ε represent the languages {ε}

3. ∅ represents the empty language

4. *($R_1 \cup R_2$), where R_1 and R_2 are regular expressions,*
5. *($R_1 \circ R_2$), where R_1 and R_2 are regular expressions,*
6. *(R_1^*), where R_1 is a regular expression.*

4, 5, and 6, the expressions represent the languages obtained by taking the union or concatenation of the languages R_1 and R_2, or the star of the language R_1, respectively.

RE Properties

- $R^{+} \cup \varepsilon = R^{*}$
- $R \cup \emptyset = R$.
- $R \circ \varepsilon = R$.
- $R \cup \varepsilon$ *may not equal* R
- $R \circ \emptyset$ *may not equal* R

Definitions

THEOREM: A language is regular if and only if some regular expression describes it.

Lemma: If a language is described by a regular expression, then it is regular.

Regular Expression to NFA

Claim: If L=L(e) for some RE e, then L= L(M) for some NFA M

Construction: Use inductive definition

1. $R=a$, with $a \in \Sigma$,

2. $R=\varepsilon$,

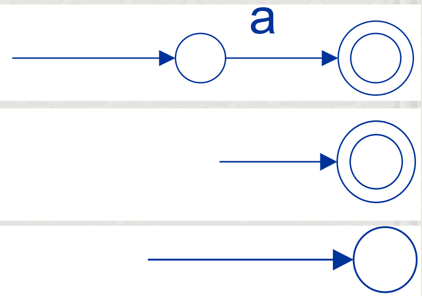
3. $R=\emptyset$,

4. $R = (R_1 \cup R_2)$, with R_1 and R_2 regular expressions

5. $R = (R_1 \circ R_2)$, with R_1 and R_2 regular expressions

6. $R=(R_1{}^*)$, with R_1 a regular expression

1. **2.** **3.**



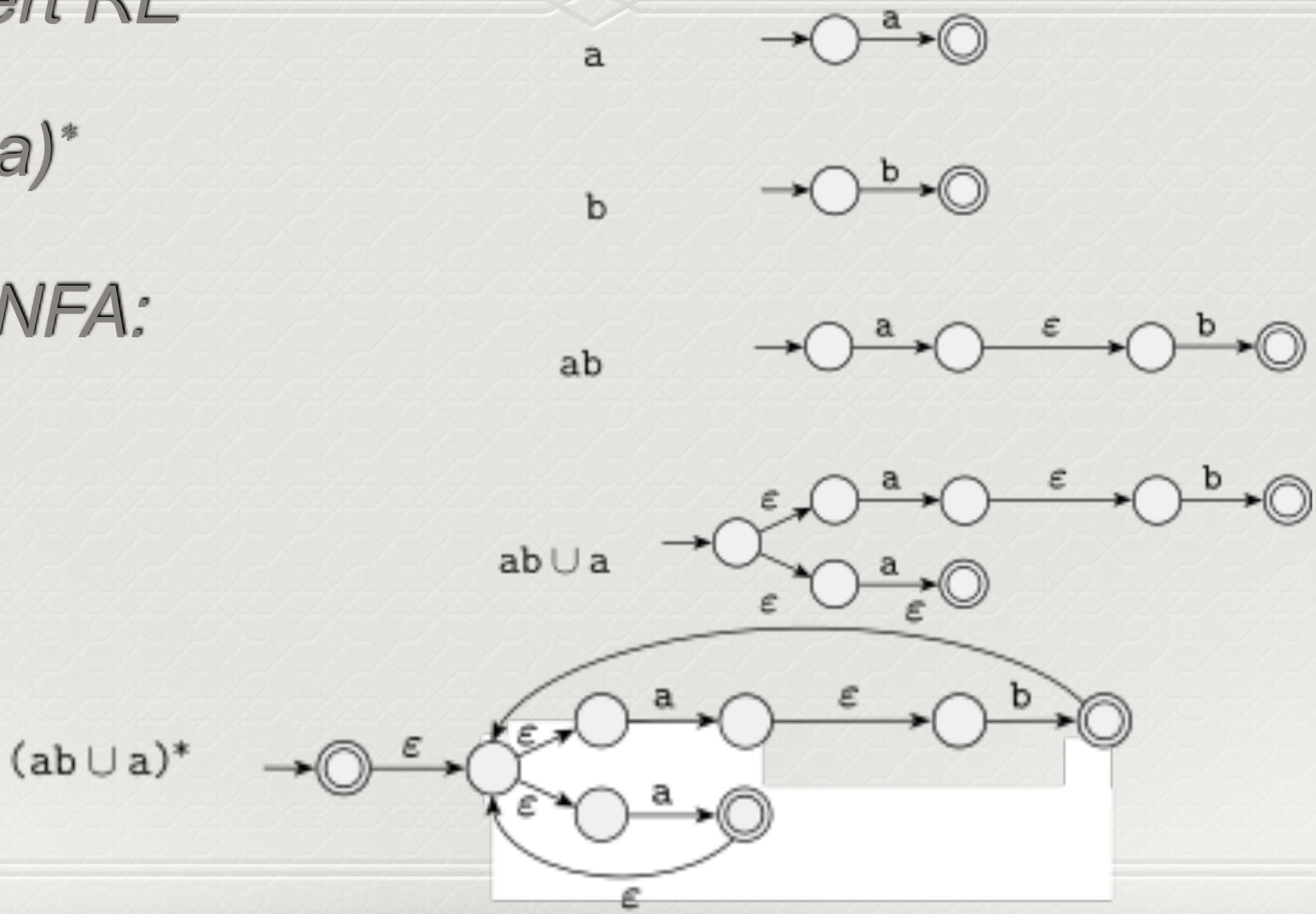
4,5,6: similar to closure of RL under regular operations.

Example

Convert RE

*(ab ∪ a)**

to an NFA:



Terminology: Closure

A set is defined to be closed under an operation if that operation on members of the set always produces a member of the same set. (adapted from wikipedia)

E.g.:The integers are closed under addition, multiplication.

- *The integers are not closed under division*
- *Σ^* is closed under concatenation*
- *A set can be defined by closure -- Σ^* is called the (Kleene) closure of Σ under concatenation.*

Terminology: Regular Operations

The regular operations are:

1. *Union*
2. *Concatenation*
3. *Star (Kleene Closure): For a language A,*

$A^* = \{w_1 w_2 w_3 \ldots w_k \mid k \geq 0, \text{ and each } w_i \in A\}$

Closure Properties

Set of regular languages is closed under:

– Union

– Concatenation

– Star (Kleene Closure)

Union of Two Languages

Theorem 1.12: *If A_1 and A_2 are regular languages, then so is $A_1 \cup A_2$.*

(The regular languages are 'closed' under the union operation.)

Proof idea: A_1 and A_2 are regular, hence there are two DFA M_1 and M_2, with A_1=$L(M_1)$ and A_2=$L(M_2)$. Out of these two DFA, we will make a third automaton M_3 such that $L(M_3) = A_1 \cup A_2$.

How do we combine DFA?

Q: Can we design a DFA that somehow "simulates" them both and accepts when at least one of them accepts?

Ans: Yes, through a clever construction.

Proof Union-Theorem (1)

$M_1=(Q_1, \Sigma, \delta_1, q_1, F_1)$ and $M_2=(Q_2, \Sigma, \delta_2, q_2, F_2)$ Define $M_3 = (Q_3, \Sigma, \delta_3, q_3, F_3)$ by:

$Q_3 = Q_1 \times Q_2 = \{(r_1,r_2) \mid r_1 \in Q_1 \text{ and } r_2 \in Q_2\}$

$\delta_3((r_1,r_2),a) = (\delta_1(r_1,a), \delta_2(r_2,a))$

$q_3 = (q_1,q_2)$

$F_3 = \{(r_1,r_2) \mid r_1 \in F_1 \text{ or } r_2 \in F_2\}$

Proof Union-Theorem (2)

The automaton $M_3 = (Q_3, \sum, \delta_3, q_3, F_3)$ runs M_1 and M_2 in 'parallel' on a string w.

In the end, the final state (r_1, r_2) 'knows'
if $w \in L_1$ (via $r_1 \in F_1$?) and if $w \in L_2$ (via $r_2 \in F_2$?)

The accepting states F_3 of M_3 are such that $w \in L(M_3)$ if and only if $w \in L_1$ or $w \in L_2$, for: $F_3 = \{(r_1, r_2) \mid r_1 \in F_1 \text{ or } r_2 \in F_2\}$.

Concatenation of L_1 and L_2

Definition: $L_1 \circ L_2 = \{ xy \mid x \in L_1 \text{ and } y \in L_2 \}$ *Example:* $\{a,b\}$

$\circ \{0,11\} = \{a0,a11,b0,b11\}$

Theorem 1.13: *If L_1 and L_2 are regular languages, then so is $L_1 \circ L_2$.*

(The regular languages are 'closed' under concatenation.)

Proving Concatenation Theorem

Consider the concatenation: {1,01,11,001,011,...} ◦ {0,000,00000,...} (That is: the bit strings that end with a "1", followed by an odd number of 0's.)

Problem is: given a string w, how does the automaton know where the L_1 part stops and the L_2 substring starts?

We need an M with 'lucky guesses'.

Non-Determinism

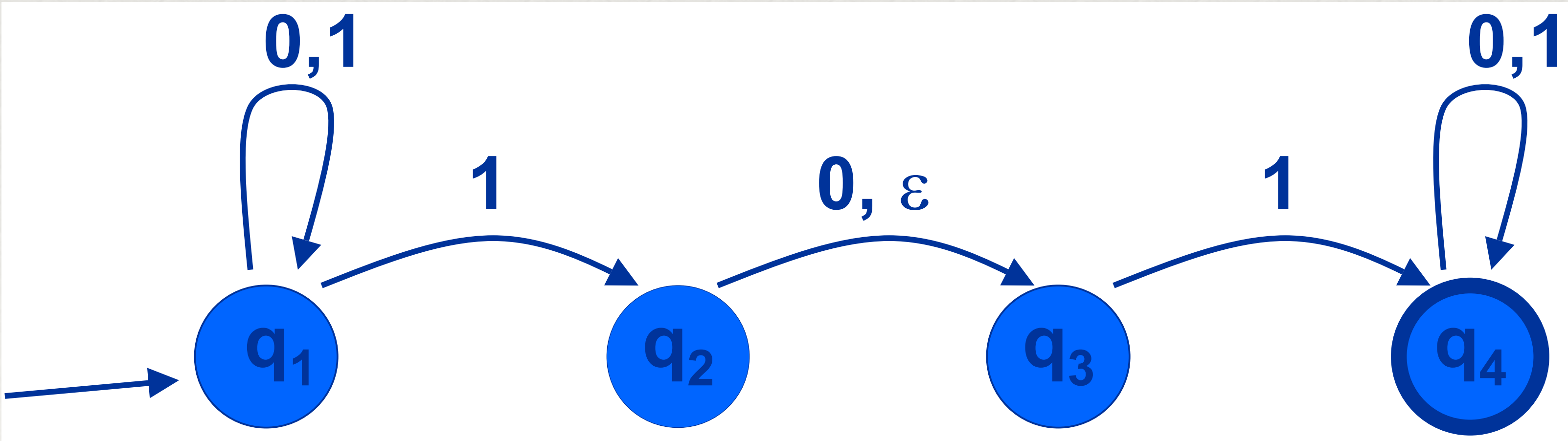
Nondeterministic machines are capable of being lucky, no matter how small the probability.

A nondeterministic finite automaton has transition rules/possibilities like

A Nondeterministic Automaton

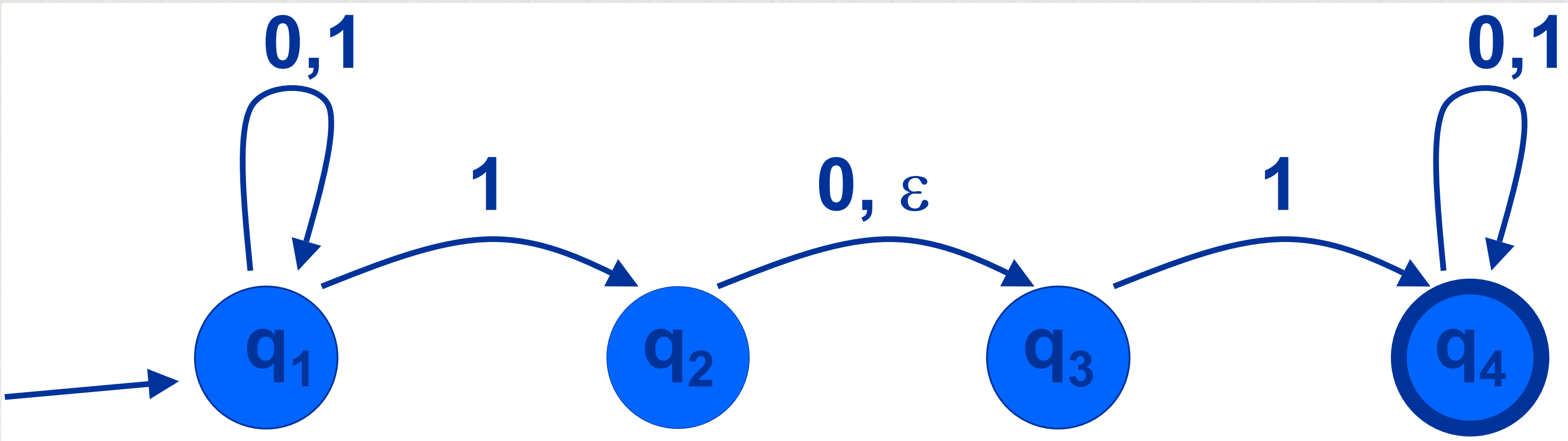
This automaton accepts "0110", because there is a possible path that leads to an accepting state, namely:

$q_1 \rightarrow q_1 \rightarrow q_2 \rightarrow q_3 \rightarrow q_4 \rightarrow q_4$



A Nondeterministic Automaton

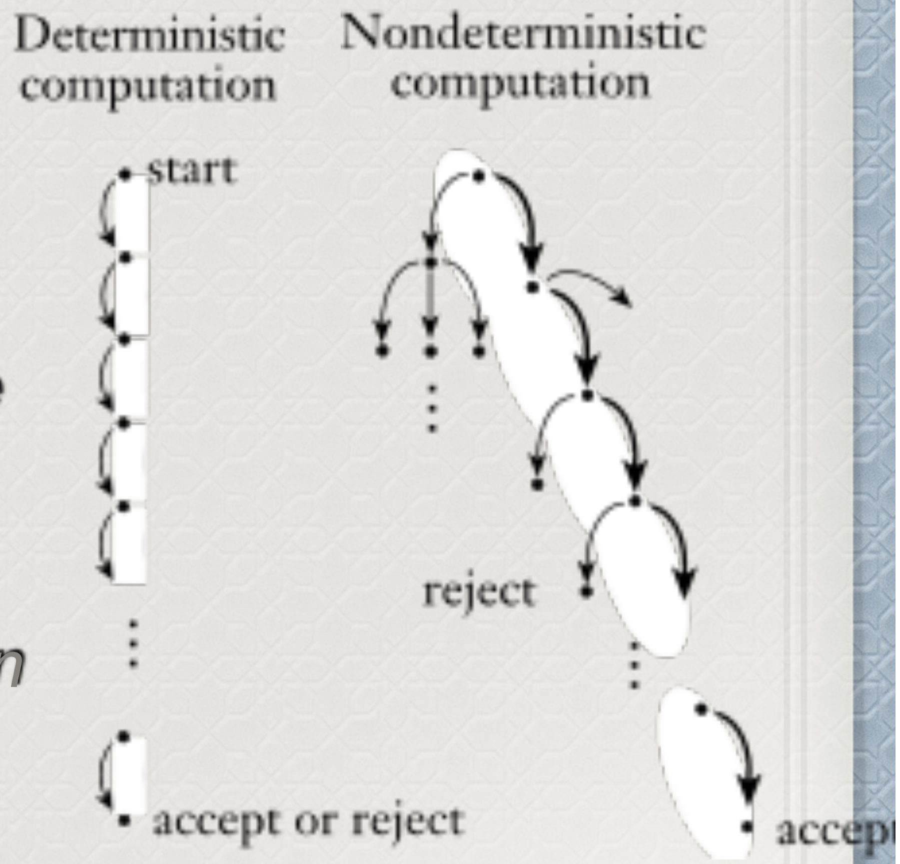
The string 1 gets rejected: on "1" the automaton can only reach: $\{q_1, q_2, q_3\}$.



Nondeterminism ~ Parallelism

For any (sub)string w, the nondeterministic automaton can be in a set of possible states.
If the final set contains an accepting state, then the automaton accepts the string.

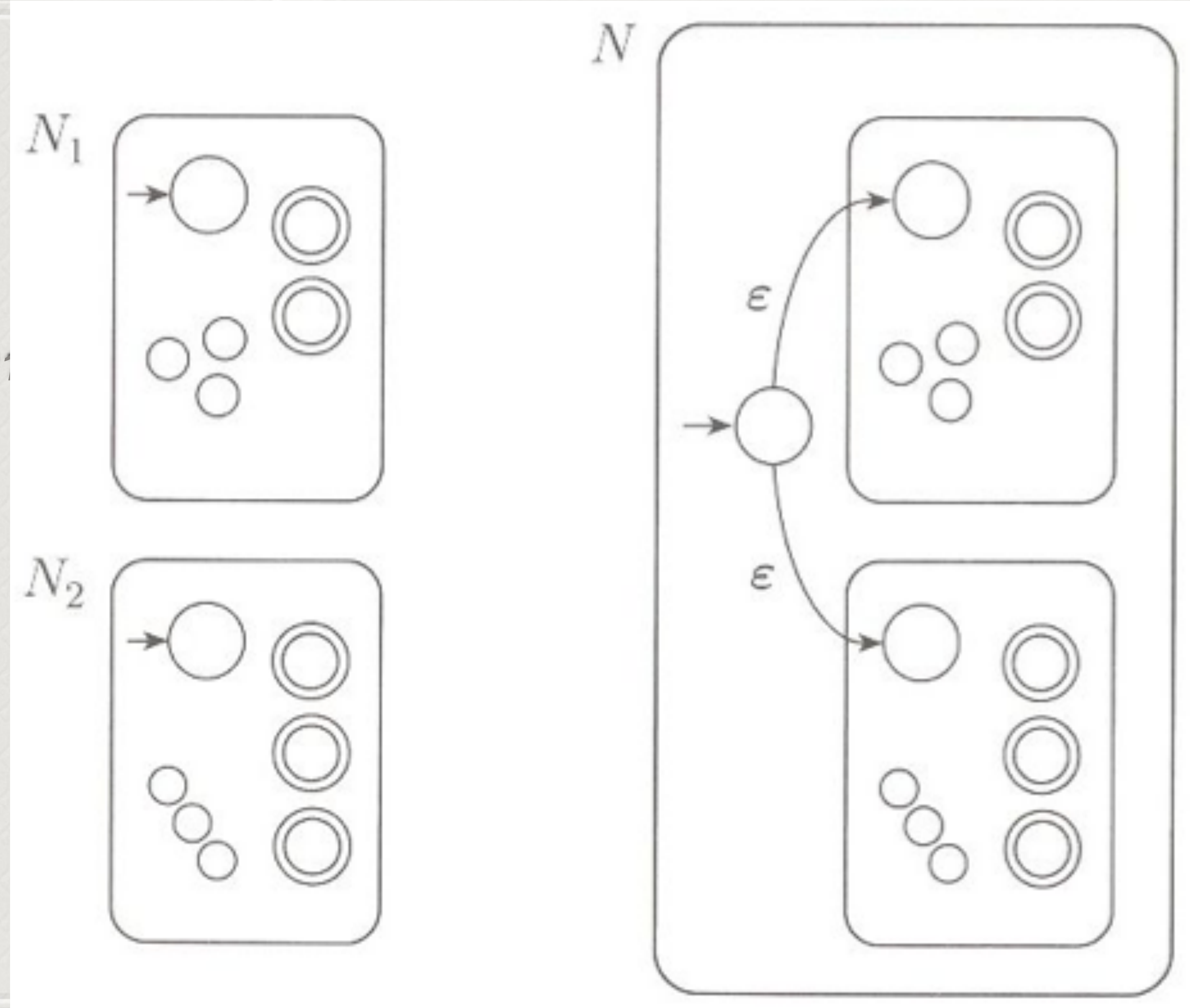
“The automaton processes the input in a parallel fashion. Its computational path is no longer a line, but a tree.”



Closure Under Regular Operations

Union (new proof):

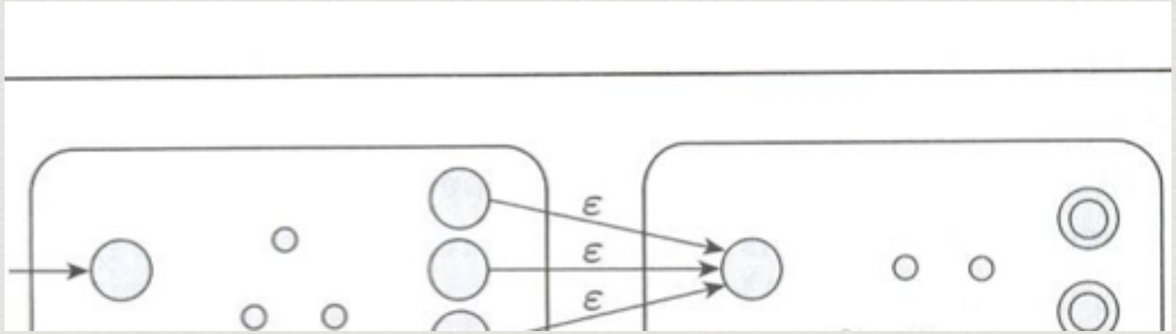
Construction of an NFA N to recognize $A_1 \cup A_2$

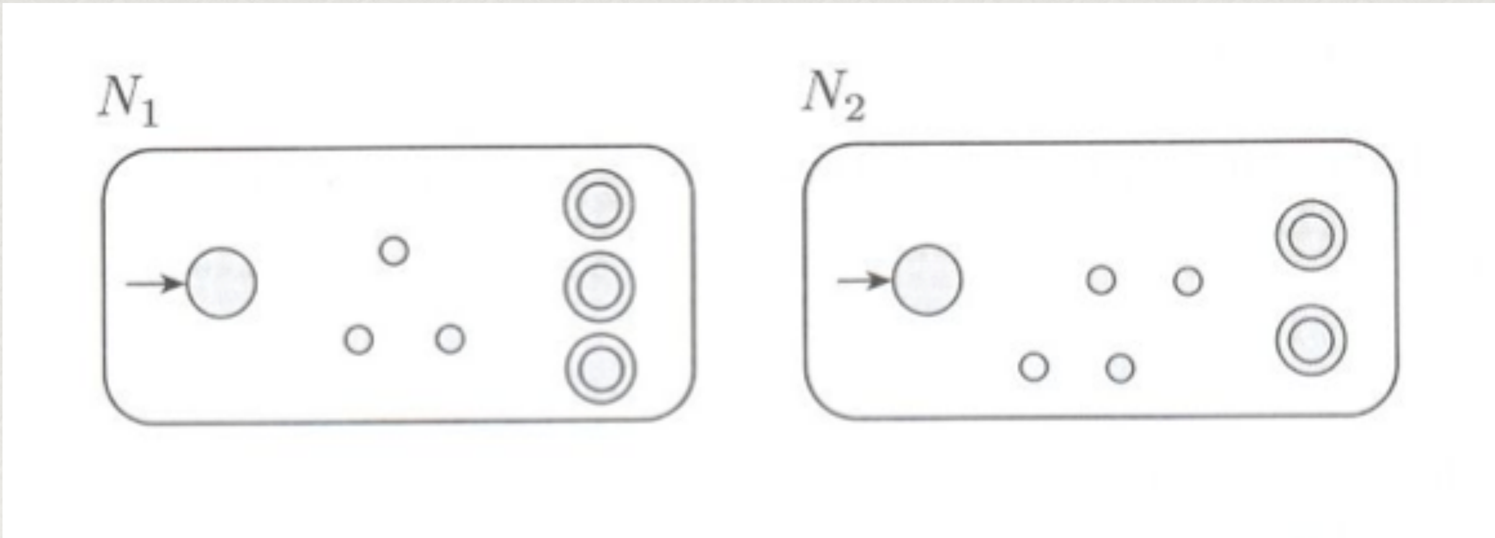


Closure under Regular Operations

Concatenation:

Construction of an NFA N to recognize $A_1 \circ A_2$





Closure under Regular Operations

Star:

Construction of an NFA N to recognize A^*