CS5371
Theory of Computation

Lecture 4: Automata Theory II
(DFA = NFA, Regular Language)

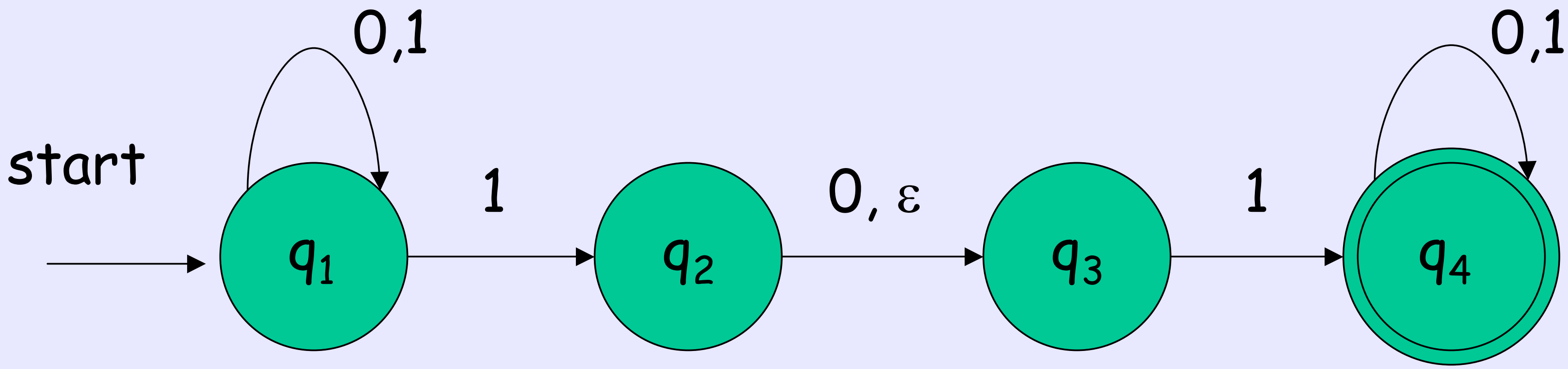
Objectives

- Give a formal definition of the non-deterministic finite automaton (NFA) and its computation
- Show that DFA = NFA in terms of string decision power
- Properties of language recognized by DFA (or NFA)

Formal Definition of NFA

- An NFA is a 5-tuple $(Q, \Sigma, \delta, q_{start}, F)$, where
  - Q is a set consisting finite number of states
  - Σ is an alphabet consisting finite number of characters
  - $\delta: Q \times \Sigma_\varepsilon \rightarrow 2^Q$ is the transition function
  - q_{start} is the start state
  - F is the set of accepting states
- Here, we let $\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$

Formal Definition of NFA



$Q = \{q_1, q_2, q_3, q_4\}$, $\Sigma = \{ 0, 1 \}$,

$q_{start} = q_1$, $F = \{ q_4 \}$,

$\delta(q_1, 0) = \{q_1\}$, $\delta(q_1, 1) = \{q_1, q_2\}$, $\delta(q_1, \varepsilon) = \{ \}$, ...

Formal Definition of NFA's Computation

- Let $M = (Q, \Sigma, \delta, q_{start}, F)$ be an NFA
- Let w be a string over the alphabet Σ
- Then, M **accepts** w if we can write $w = w_1 w_2 \ldots w_n$ such that each $w_i \in \Sigma_\varepsilon$ and a sequence of states $r_0, r_1, \ldots, r_n$ in Q exists with the three conditions:
  - $r_0 = q_{start}$
  - $r_{i+1} \in \delta(r_i, w_{i+1})$
  - $r_n \in F$

compare this with DFA

DFA = NFA
(in terms of string decision power)

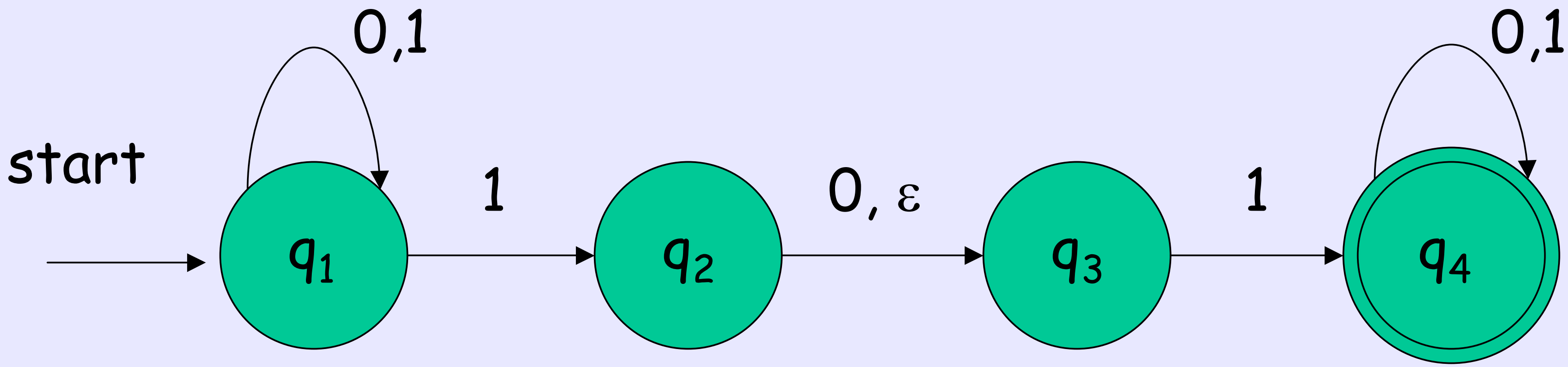
Theorem: (1) If a language L can be recognized by a DFA, then there exists an NFA that can recognize L; (2) If a language L' can be recognized by an NFA, then there exists a DFA that recognizes L'.

Proof: For (1), it is easy. (why?)
For (2), how to prove?

DFA = NFA (Proof Idea)

- We prove (2) by showing that: Given a language L′ recognized by an NFA, we can always find a DFA that recognizes L′ (what kind of proof technique?)
- To help our discussion, we define the following:
  - For any string w, let R(w) denote "the set of states that NFA can exactly reach" after reading all characters of w.

DFA = NFA (Proof Idea)



E.g., $R(0) = \{q_1\}$, $R(1) = \{q_1, q_2, q_3\}$,

$R(00) = \{q_1\}$

$R(11) = \{q_1, q_2, q_3, q_4\}$

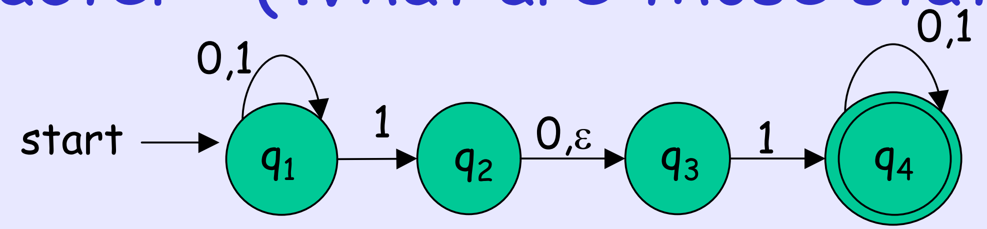
DFA = NFA (Proof Idea)

If we are the DFA simulating the NFA

- At any time when part of the input string is processed, say we have read w', we MUST need to know exactly what is R(w')… Otherwise,
  - if we miss a state of R(w'), what bad things may happen?
  - if we have an extra state, what bad things may happen?

DFA = NFA (Proof Idea)

- On the other hand, $R(w')$ is what we only need to know
  - Because if we know $R(w')$, we know exactly the set of states NFA can exactly reach after reading one more character (What are those states??)
- E.g.,



$R(w') = \{q_1, q_3\}$, $R(w'0) = ??$ $R(w'1) = ??$

DFA = NFA (Proof Idea)

- By looking at $R(w')$, how can we determine if the NFA accepts w'?
  - Question: If q is an accepting state, and we know that $q \in R(w')$, will the NFA accepts w'?
  - Answer: Yes, since $q \in R(w')$ means that by reading w', there is some way we can reach the accepting state q in NFA. By definition, w' is accepted
- In fact, w' is accepted if and only if some accepting state q is in $R(w')$

DFA = NFA (Proof Idea)

- If we can list out the R(w)'s for all w, we can simulate the computation of NFA
- However, there are infinite number of strings w_1, w_2, … (what could we do?)
- How about the number of possible set of states, $R(w_1)$, $R(w_2)$, …, that are just reachable by an NFA?
  - Are there infinite of them?

DFA = NFA (Formal Proof)

- Let $N = (Q, \Sigma, \delta, q_{start}, F)$ be the NFA recognizing some language A
- We construct a DFA $D = (Q', \Sigma, \delta', q_{start}', F')$ recognizing A as follows
- $Q' = 2^Q$

each state of D corresponds to a particular R(w)

- q_{start}' = the state corresponding to $R(\varepsilon)$

  $= E(q_{start})$

where $E(X) = \{X\} \cup$ the set of states that NFA N can reach from X by following only ε arrows

DFA = NFA (Formal Proof)

- $F' = \{ Y \in Q' \mid Y$ contains an accept state of N $\}$

D accepts if one of the possible states that N can now be in is an accept state

- For $Y \in Q'$ and $a \in \Sigma$,

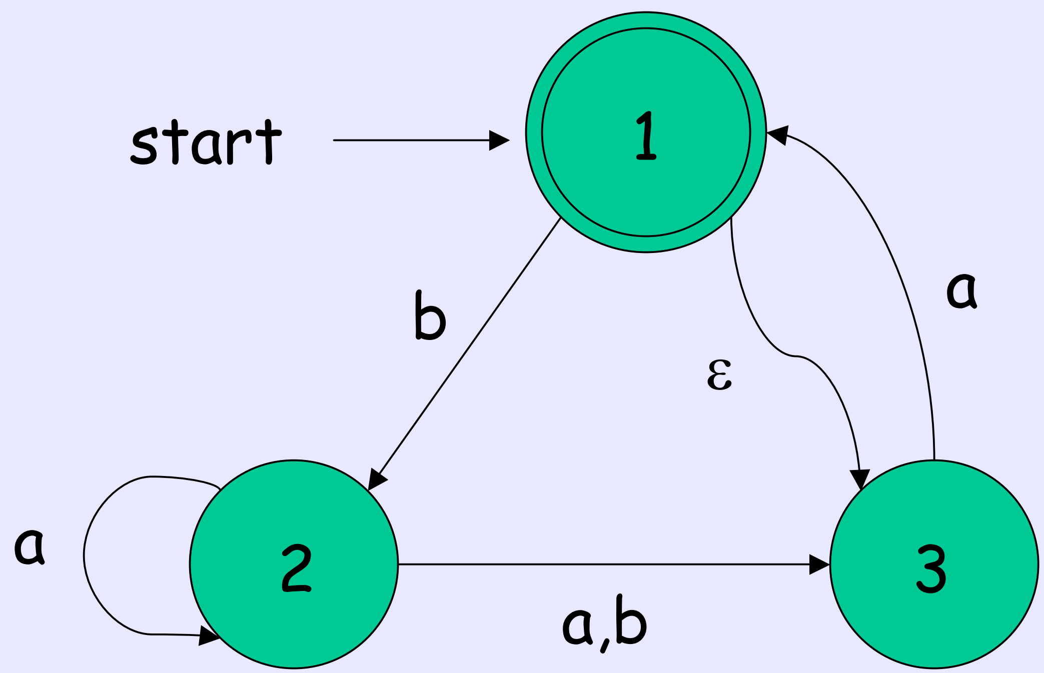
  $\delta'(Y, a) = \{ q \mid q \in E(\delta(y,a))$ for some $y \in Y \}$

The reason why $\delta'(Y,a)$ is defined in this way is because: If N is in one of the states in Y, after reading the character a, N can be in any of the states in $\delta(y,a)$, so that N can be in any states in $E(\delta(y,a))$

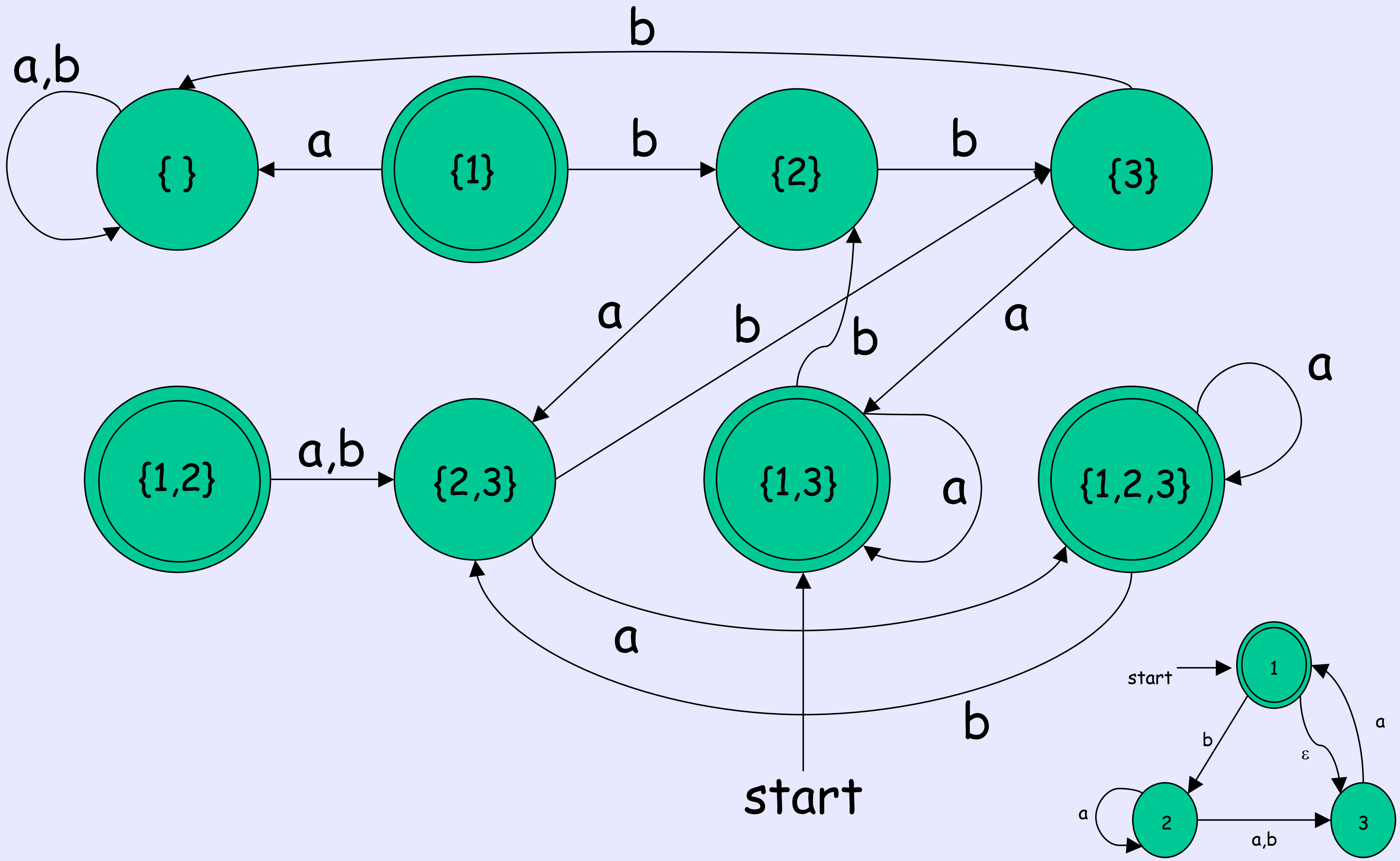
DFA = NFA (Formal Proof)

- At every step in D's computation, D clearly enters a state that corresponds to the subset of states N can exactly reach at that point. Thus, the DFA D recognizes the same language as the NFA N. Our proof completes.

Constructing DFA from NFA (Example)



Constructing DFA from NFA (Example)



Properties of Language Recognized by DFA or NFA

Theorem: If A and B are languages recognized by DFAs, then the language

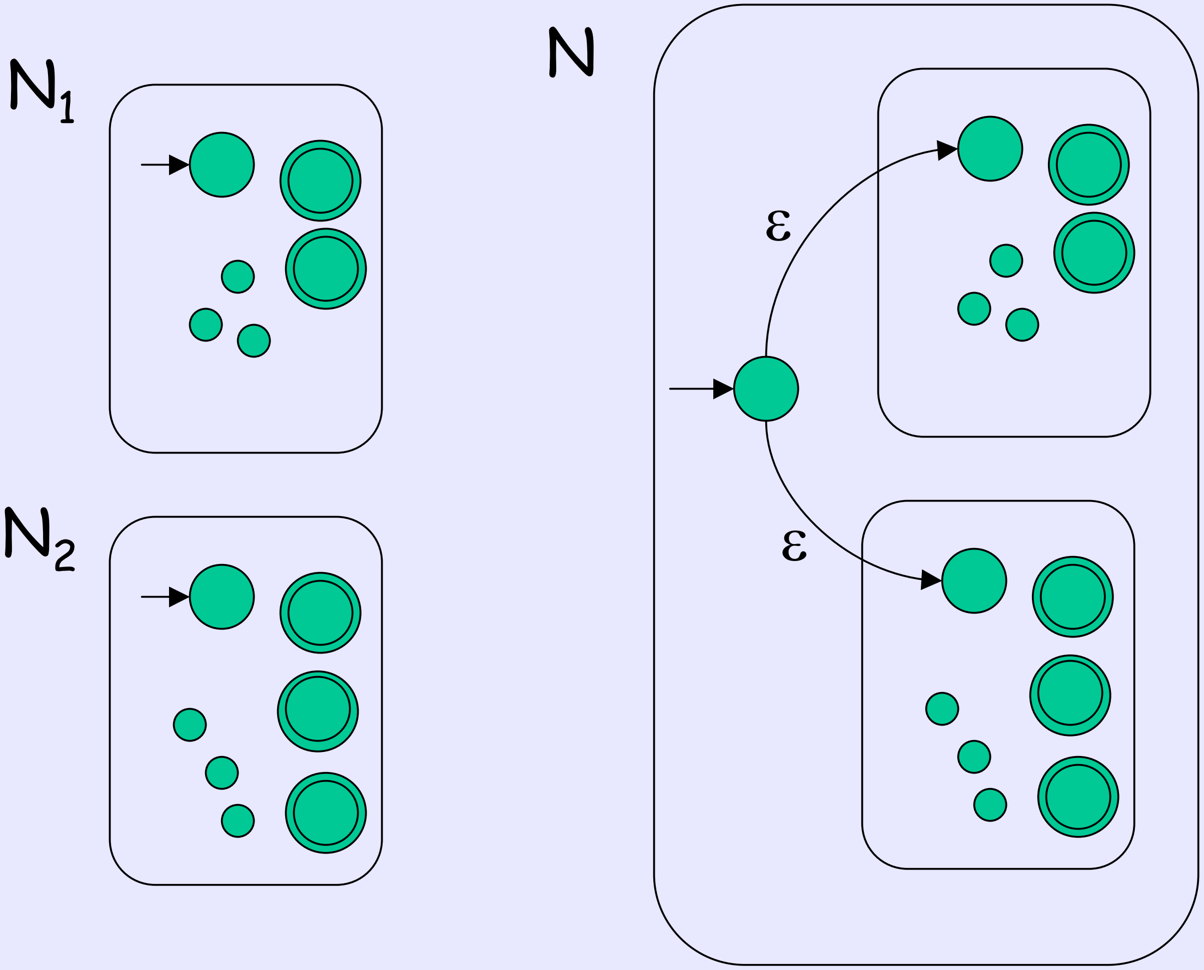
$$A \cup B = \{ x \mid x \in A \text{ or } x \in B \}$$

can also be recognized by a DFA.

Proof: Let N_1 be DFA recognizing A,
and N_2 be DFA recognizing B.

Construct NFA N that recognizes $A \cup B$.

Proof (Informal)



Properties of Language Recognized by DFA or NFA

Theorem: If A and B are languages recognized by DFAs, then the language

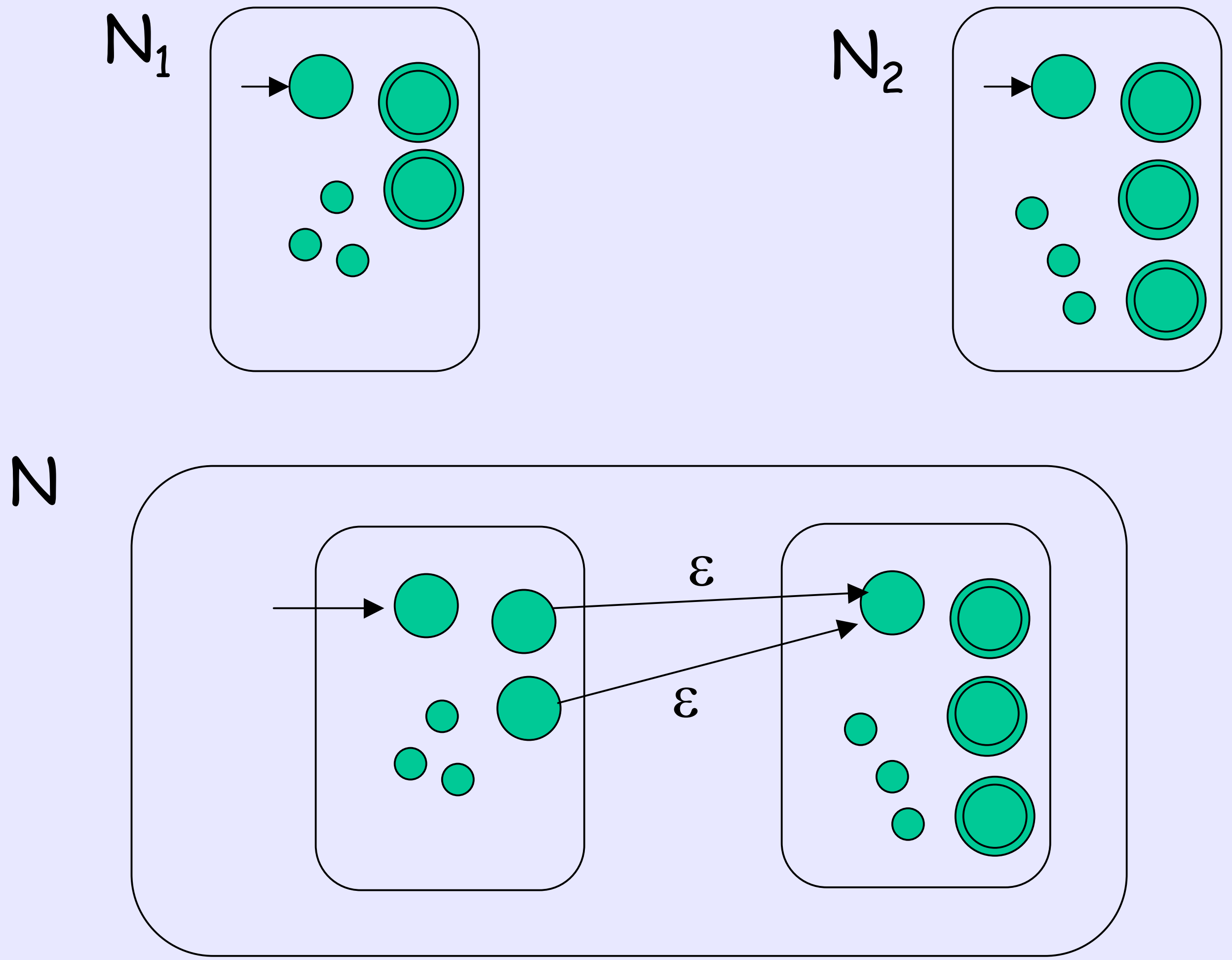
$$A \circ B = \{ xy \mid x \in A \text{ and } y \in B \}$$

can also be recognized by a DFA.

Proof: Let N_1 be DFA recognizing A, and N_2 be DFA recognizing B.

Construct NFA N that recognizes AB.

Proof (Informal)



Properties of Language Recognized by DFA or NFA

Theorem: If A is a language that can be recognized by a DFA, then the language

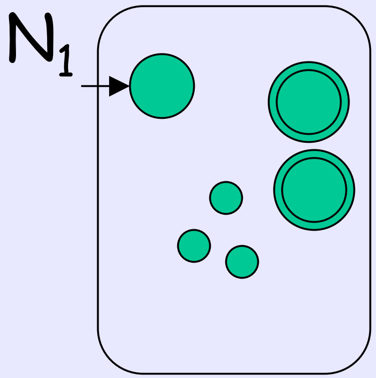
$A^* = \{ x_1 x_2 \ldots x_k \mid k \geq 0 \text{ and } x_i \in A \}$ can

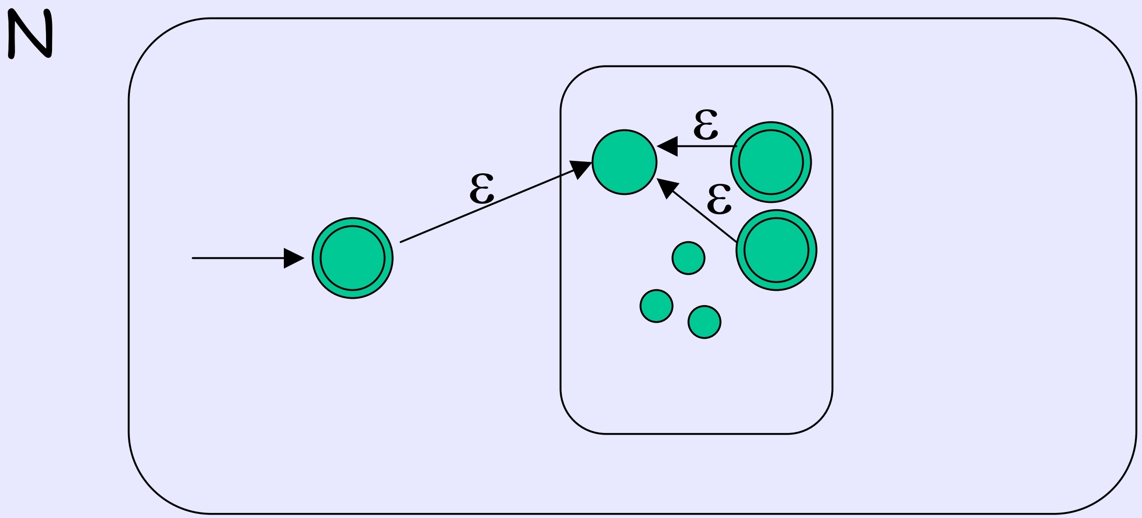
also be recognized by a DFA.

Proof: Let N_1 be DFA recognizing A.

Construct NFA N that recognizes A^*.

Proof (Informal)





Regular Language

- The Union, Concatenation, and Star operations are called regular operations
- Languages that can be recognized by DFA are called regular language

Practice at Home

- We have given informal construction of N, showing that the class of regular languages is closed under union operations

  That is, if we take two regular languages and perform union operations on them, the resulting language is also a regular language

- Can you give formal construction? That is, with

  $N_1 = (Q_1, \Sigma_1, \delta_1, q_1, F_1)$ and

  $N_2 = (Q_2, \Sigma_2, \delta_2, q_2, F_2)$,

  what are the values for the tuples in N?

Practice at Home

- Also, how about the formal constructions of N showing that the class of regular languages is closed under concatenation operation and is closed under star operations?

Next time

- Are there Non-Regular Languages?
- Introduce "Regular Expression" and show its relationship Regular Language