CS301: Theory of Computation
Section 1.2 - Nondeterminism

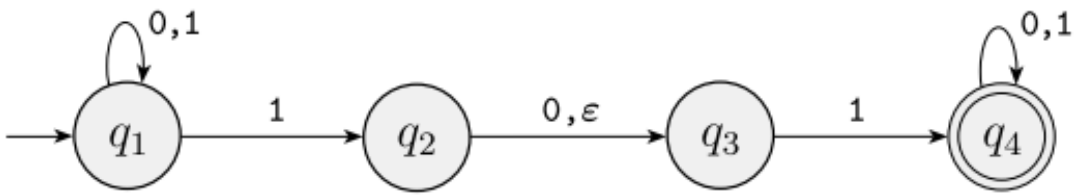
Daniel White

Gettysburg College
Computer Science

Spring 2024

When a (deterministic) finite automaton is in a given state and reads the next input symbol, we know exactly what next state will be. This is known as **deterministic** computation.

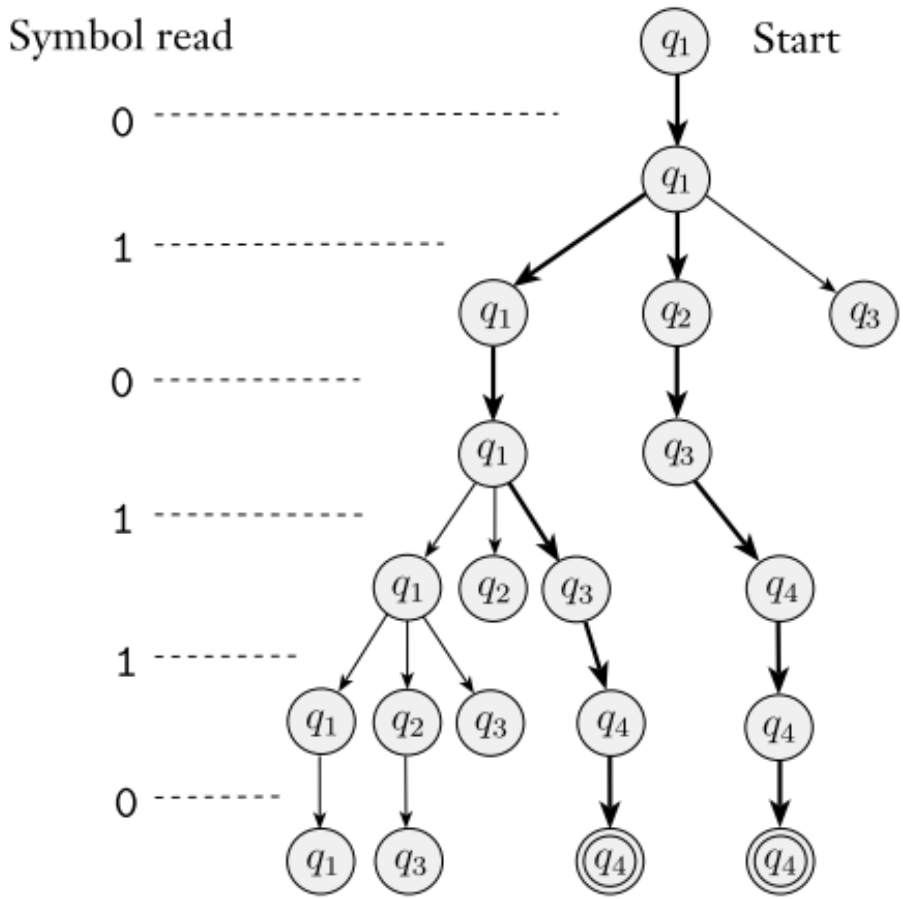
In a **nondeterministic** machine, there may be several choices for the next state at any given point. For example, consider the state diagram of the machine N_1 below.



exercise: List all the ways N_1 fails to be a deterministic finite automaton. (And what is ε doing?)

The machine N_1 is an example of a **nondeterministic finite automaton (NFA)**. So, how does it compute?

- If there are multiple ways to proceed (*i.e.* multiple transition arrows to follow), the machine splits into multiple copies of itself and follows all possibilities in parallel.
- If the next input symbol doesn't appear on any transitions from the current state, that branch of computation dies.
- If a state with any exiting ε-transitions is encountered, the machine copies itself and the copies follow those transitions without reading an input symbol.
- If *any branch* of computation terminates at an accept state, the machine accepts the input string. Otherwise, rejects.



homework: 1.7ab

Definition 1.37 (pg. 53)

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

1. Q is a finite set called the **states**,
2. Σ is a finite set called the **alphabet**,
3. $\delta : Q \times \Sigma_\varepsilon \to \mathcal{P}(Q)$ is the **transition function**,
4. $q_0 \in Q$ is the **start state**, and
5. $F \subseteq Q$ is the **set of accept states**.

exercise: Express the machine N_1 from the previous slides formally as a NFA.

Definition (pg. 54)

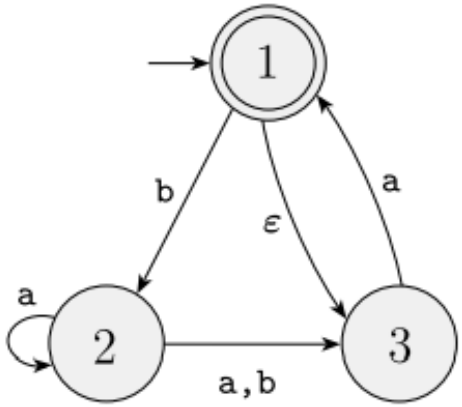
Say that two machines are **equivalent** if they recognize the same language.

Theorem 1.39 (pg. 55)

Every NFA is equivalent to some DFA.

Proof. Let A be a language recognize by some NFA, N. We must show there exists some DFA, M, that recognizes A as well. We will do so by construction, defining a DFA, M, whose state set simulates all possible parallel states encounterable by N . . .

exercise: Finish this proof. *(pgs. 55-56)*



exercise: Convert N_4 (above) into a DFA using the ideas found in the proof of Theorem 1.39.

homework: 1.11 & convert N_1 into a DFA

Corollary 1.40 (pg. 56)

A language is regular if and only if some NFA recognizes it.

The previous section concluded on a cliff-hanger where we hoped to prove that the class of regular languages is closed under the union, concatenation, and star operations.

NFAs provide us with ways to both clean up the proof for the union case, and also to approach the other two operations.

Definition 1.23 (pg. 44)

Let A and B be languages. We define the regular operations as follows:

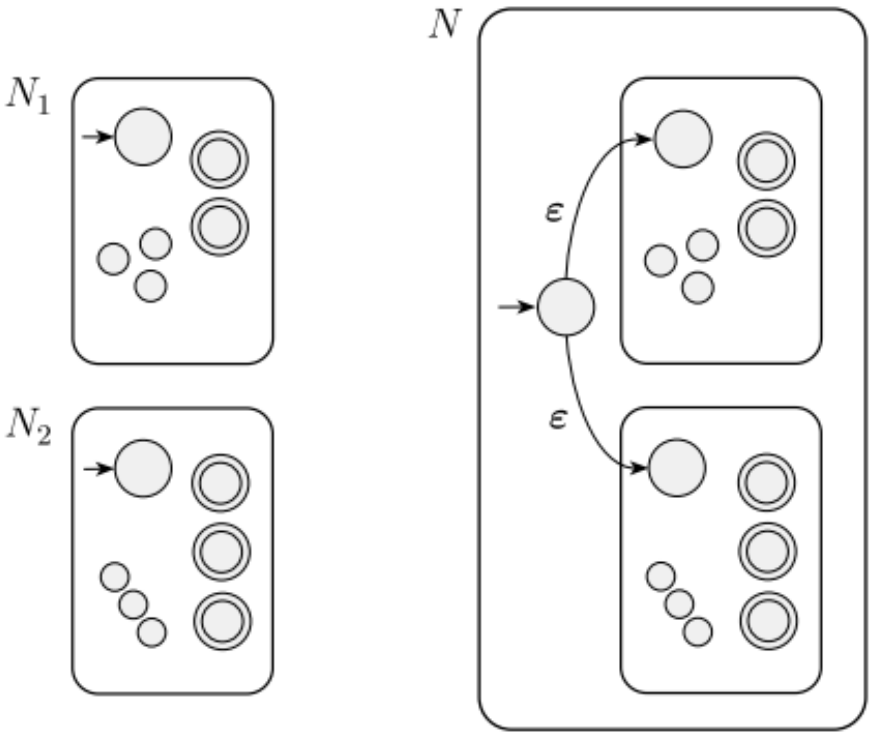
1. **union:** $A \cup B = \{x \mid x \in A \text{ or } x \in B\}$
2. **concatenation:** $A \circ B = \{xy \mid x \in A \text{ and } y \in B\}$
3. **star:** $A^* = \{x_1 x_2 \dots x_k \mid k \geq 0 \text{ and each } x_i \in A\}$

homework: rigorize the upcoming proofs *(pgs. 58-63)*

Theorem 1.45 (pg. 59)

The class of regular languages is closed under the union operation.

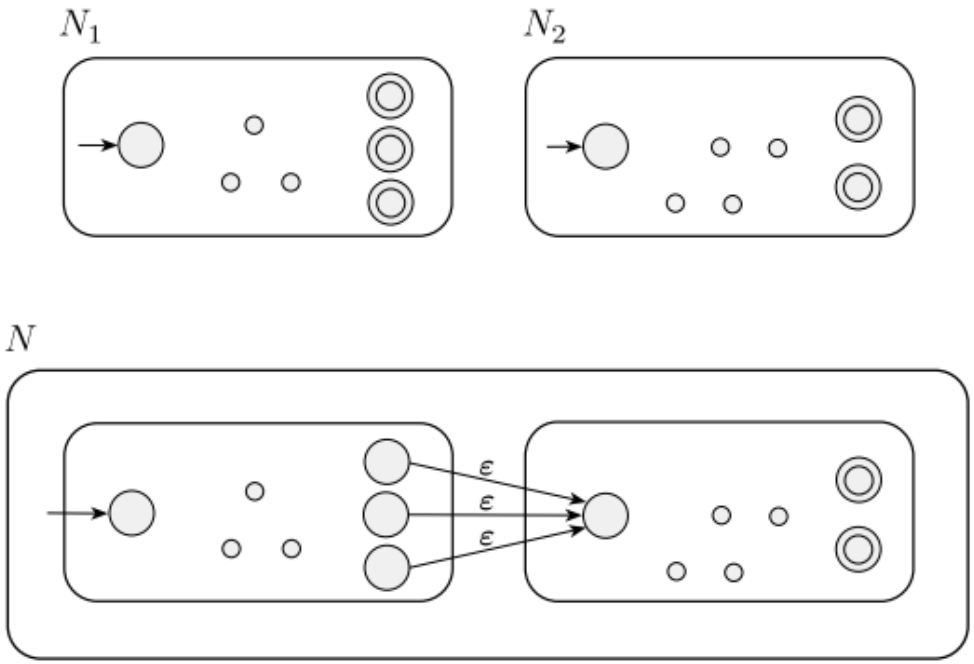
Proof. Let N_1 and N_2 be DFAs recognizing languages A_1 and A_2, respectively ...



Theorem 1.47 (pg. 60)

The class of regular languages is closed under the concatenation operation.

Proof. Let N_1 and N_2 be DFAs recognizing languages A_1 and A_2, respectively . . .



Theorem 1.49 (pg. 62)

The class of regular languages is closed under the star operation.

Proof. Let N_1 be a DFA recognizing language A_1 ...