

T065001: Introduction to Formal Languages

Lecture 3: Finite-state automata, regular languages, nondeterminism (2)

Chapter 1.2 in Sipser's textbook

2025-04-28

(Lecture slides by Yih-Kuen Tsay)

Nondeterminism

- In a *nondeterministic* machine, several choices may exist for the next state after reading the next input symbol in a given state.
- The difference between a deterministic finite automaton (DFA) and a nondeterministic finite automaton (NFA):

	# of next states (per symbol)	input symbols
DFA	1	from Σ
NFA	0, 1, or more	from $\Sigma \cup \{\varepsilon\}$

Nondeterminism (cont.)

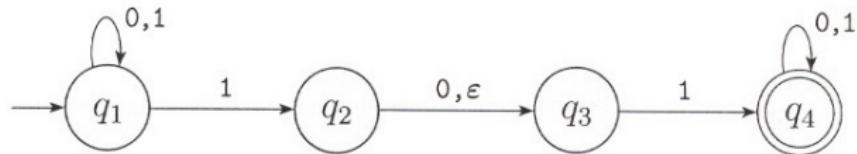
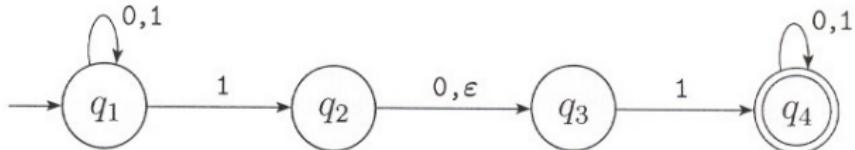


FIGURE 1.27

The nondeterministic finite automaton N_1

Nondeterminism (cont.)



FIGURE

1.27

The nondeterministic finite automaton N_1

Note: N_1 accepts all strings over the alphabet $\{0, 1\}$ that contain 101 or 11 as a substring.

How Does an NFA Compute?

1. If there are multiple choices for the next state, given the next input symbol, the machine splits into multiple copies, all moving to their respective next states in parallel.
2. Additional copies are also created if there are exiting arrows labeled with ϵ , one copy for each of such arrows. All copies move to their respective next states in parallel, but without consuming any input.
3. If *any* copy is in an accept state at the end of the input, the machine accepts the input string.
4. If there are input symbols remaining, the preceding steps are repeated.

Deterministic vs. Nondeterministic Comp.

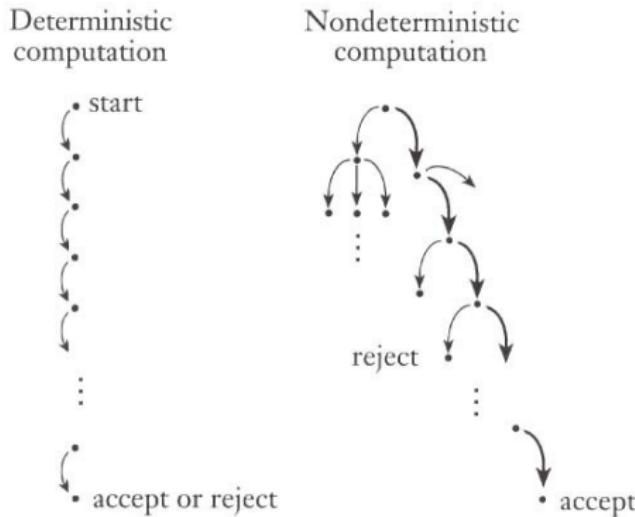


FIGURE 1.28

Deterministic and nondeterministic computations with an accepting branch

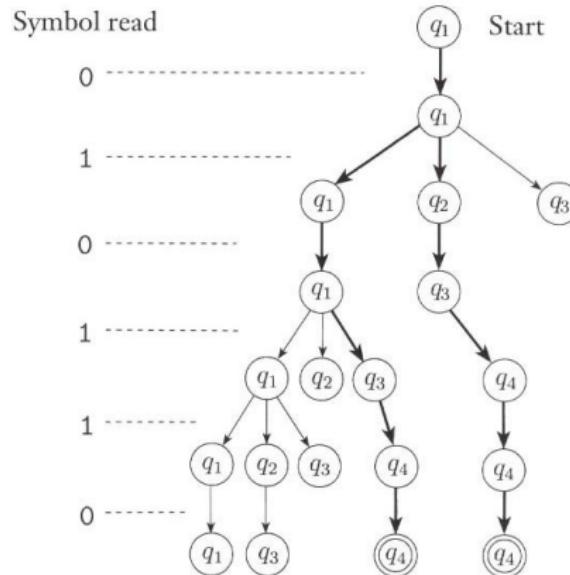
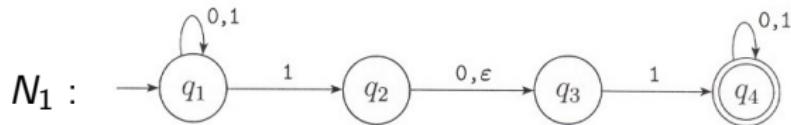


FIGURE 1.29

The computation of N_1 on input 010110

Nondeterminism (cont.)

- ➊ Nondeterminism is a useful concept that has had great impact on computation theory.
- ➋ As we will show, *every NFA can be converted into an equivalent DFA.*
- ➌ However, constructing NFAs is sometimes easier than directly constructing DFAs. An NFA may be much smaller than its deterministic counterpart, or its functioning may be easier to understand.

Example NFA

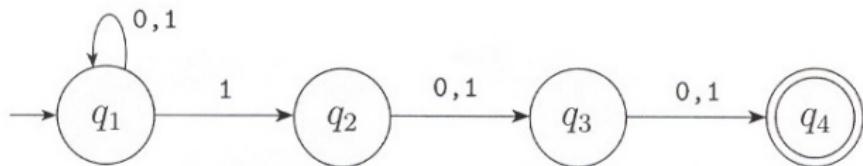


FIGURE 1.31
The NFA N_2 recognizing A

Example NFA

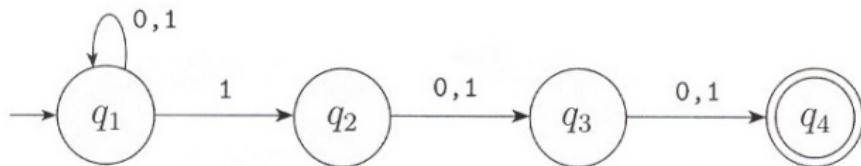


FIGURE 1.31
The NFA N_2 recognizing A

Note: A is the set of all strings over $\{0, 1\}$ containing a 1 in the last third position.

Example NFA (cont.)

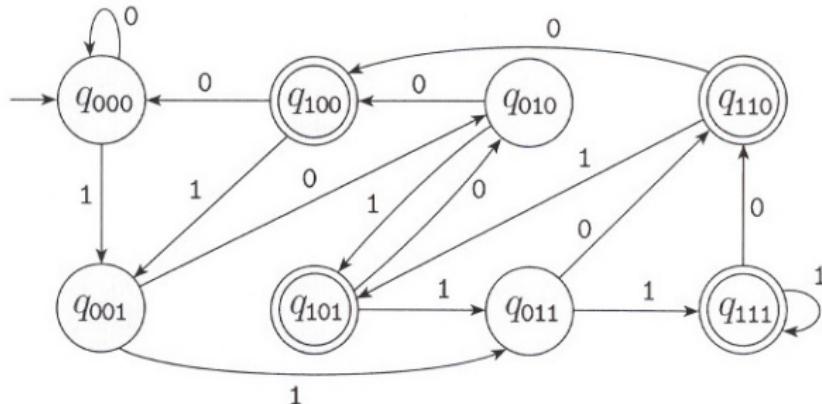


FIGURE 1.32
A DFA recognizing A

Note: A is the set of all strings over $\{0, 1\}$ containing a 1 in the last third position.

Example NFA (cont.)

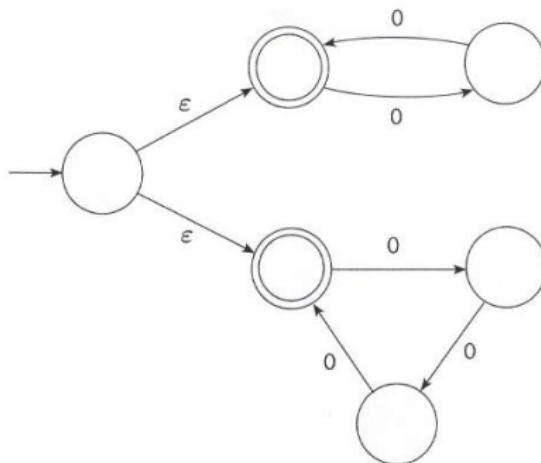


FIGURE 1.34

The NFA N_3

Example NFA (cont.)

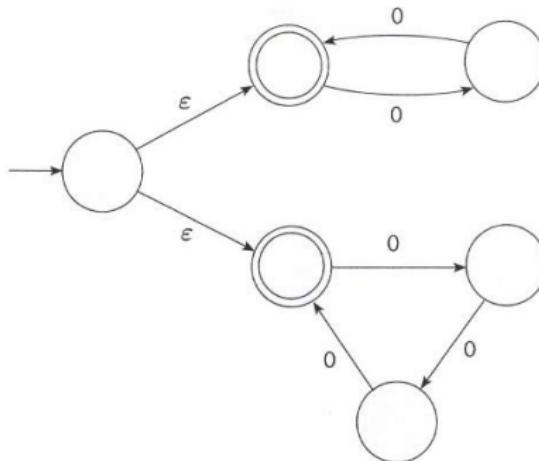


FIGURE 1.34

The NFA N_3

Note: N_3 accepts all strings of the form 0^k where k is a multiple of 2 or 3.

Example NFA (cont.)

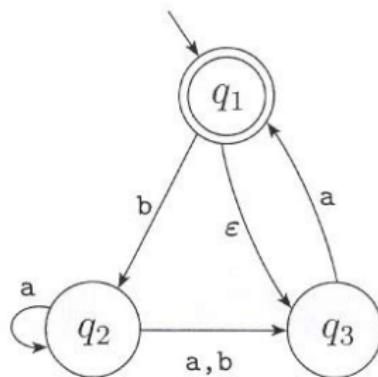


FIGURE 1.36
The NFA N_4

Does N_4 accept ε ?

Example NFA (cont.)

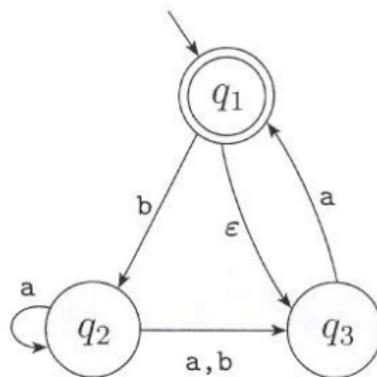


FIGURE 1.36

The NFA N_4

Does N_4 accept ϵ ? How about babaa?

Example NFA (cont.)

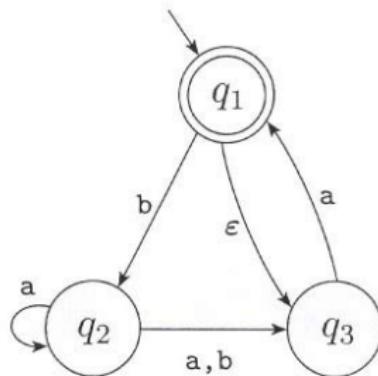


FIGURE 1.36

The NFA N_4

Does N_4 accept ϵ ? How about babaa?

(Yes and yes.)

Definition of an NFA

- The transition function of an NFA takes a state and an input symbol *or the empty string* and produces *a set of possible next states*.
- Let $\mathcal{P}(Q)$ be the power set of Q and let Σ_ε denote $\Sigma \cup \{\varepsilon\}$.

Example: Let $Q = \{q_0, q_1, q_2\}$. Then

$$\mathcal{P}(Q) = \{\emptyset, \{q_0\}, \{q_1\}, \{q_2\}, \{q_0, q_1\}, \{q_0, q_2\}, \{q_1, q_2\}, \{q_0, q_1, q_2\}\}.$$

Definition of an NFA

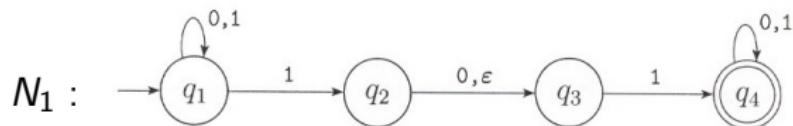
- The transition function of an NFA takes a state and an input symbol *or the empty string* and produces *a set of possible next states*.
- Let $\mathcal{P}(Q)$ be the power set of Q and let Σ_ϵ denote $\Sigma \cup \{\epsilon\}$.

Definition (1.37)

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

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

Example NFA (cont.)



Formally, $N_1 = (Q, \Sigma, \delta, q_1, F)$, where

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

	0	1	ε
q_1	$\{q_1\}$	$\{q_1, q_2\}$	\emptyset
q_2	$\{q_3\}$	\emptyset	$\{q_3\}$
q_3	\emptyset	$\{q_4\}$	\emptyset
q_4	$\{q_4\}$	$\{q_4\}$	\emptyset

3. δ is given as
4. q_1 is the start state, and
5. $F = \{q_4\}$.

Formal Def. of Nondeterministic Comp.

- ➊ Let $N = (Q, \Sigma, \delta, q_0, F)$ be an NFA and w be a string over Σ .
- ➋ We say that N *accepts* w if we can write $w = y_1y_2\dots y_m$, where $y_i \in \Sigma_\epsilon$, and a sequence of states r_0, r_1, \dots, r_m exists such that
 1. $r_0 = q_0$,
 2. $r_{i+1} \in \delta(r_i, y_{i+1})$, for $i = 0, 1, \dots, m - 1$, and
 3. $r_m \in F$.

Formal Def. of Nondeterministic Comp.

- ➊ Let $N = (Q, \Sigma, \delta, q_0, F)$ be an NFA and w be a string over Σ .
- ➋ We say that N *accepts* w if we can write $w = y_1y_2\dots y_m$, where $y_i \in \Sigma_\epsilon$, and a sequence of states r_0, r_1, \dots, r_m exists such that
 1. $r_0 = q_0$,
 2. $r_{i+1} \in \delta(r_i, y_{i+1})$, for $i = 0, 1, \dots, m - 1$, and
 3. $r_m \in F$.

Notation:

The set of all strings that are accepted by an NFA N is denoted by $L(N)$. We also say that N *recognizes* this language.

Equivalence of NFA and DFA

Two machines are *equivalent* if they recognize the same language.
(I.e., “ M_1 and M_2 are equivalent” means “ $L(M_1) = L(M_2)$ ”.)

Equivalence of NFA and DFA

Two machines are *equivalent* if they recognize the same language.
(I.e., “ M_1 and M_2 are equivalent” means “ $L(M_1) = L(M_2)$ ”.)

Theorem (1.39)

Every nondeterministic finite automaton has an equivalent deterministic finite automaton.

Corollary (1.40)

A language is regular if and only if some nondeterministic finite automaton recognizes it.

Equivalence of NFA and DFA (cont.)

Theorem (1.39)

Every NFA has an equivalent DFA.

- ➊ The idea is to convert a given NFA into an equivalent DFA that *simulates* the NFA.
- ➋ An NFA can be in one of several possible states, as it reads the input.
- ➌ If k is the number of states of the NFA, it has 2^k subsets of states. Each subset corresponds to one of the possibilities that the simulating DFA must remember.

Equivalence of NFA and DFA (cont.)

Theorem (1.39)

Every NFA has an equivalent DFA.

- 💡 Let $N = (Q, \Sigma, \delta, q_0, F)$ be an NFA recognizing some language A .

Equivalence of NFA and DFA (cont.)

Theorem (1.39)

Every NFA has an equivalent DFA.

- ➊ Let $N = (Q, \Sigma, \delta, q_0, F)$ be an NFA recognizing some language A .
- ➋ Construct a DFA $M = (Q', \Sigma, \delta', q'_0, F')$ that also recognizes A as follows.
(First consider the easier case where N has no ε arrows, then extend the construction to also allow them.)

Equivalence of NFA and DFA (cont.)

1. $Q' = \mathcal{P}(Q)$.
2. For $R \in Q'$ and $a \in \Sigma$, let $\delta'(R, a) = \bigcup_{r \in R} \delta(r, a)$.
3. $q'_0 = \{q_0\}$.
4. $F' = \{R \in Q' \mid R \text{ contains some element of } F\}$.

Equivalence of NFA and DFA (cont.)

1. $Q' = \mathcal{P}(Q)$.
 2. For $R \in Q'$ and $a \in \Sigma$, let $\delta'(R, a) = \bigcup_{r \in R} \delta(r, a)$.
 3. $q'_0 = \{q_0\}$.
 4. $F' = \{R \in Q' \mid R \text{ contains some element of } F\}$.
-  To allow ε arrows, define for $R \subseteq Q$,
- $$E(R) = \{q \mid q \text{ can be reached from } R \text{ by following 0 or more } \varepsilon \text{ arrows}\}$$
-  Replace $\delta(r, a)$ with $E(\delta(r, a))$ and set q'_0 to be $E(\{q_0\})$ in the construction of M .

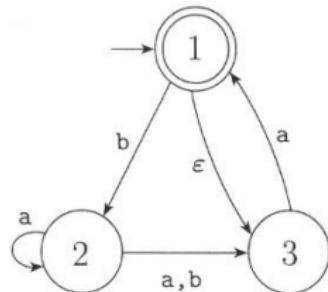
Equivalence of NFA and DFA (cont.)

1. $Q' = \mathcal{P}(Q)$.
 2. For $R \in Q'$ and $a \in \Sigma$, let $\delta'(R, a) = \bigcup_{r \in R} \delta(r, a)$.
 3. $q'_0 = \{q_0\}$.
 4. $F' = \{R \in Q' \mid R \text{ contains some element of } F\}$.
-  To allow ε arrows, define for $R \subseteq Q$,
- $$E(R) = \{q \mid q \text{ can be reached from } R \text{ by following 0 or more } \varepsilon \text{ arrows}\}$$
-  Replace $\delta(r, a)$ with $E(\delta(r, a))$ and set q'_0 to be $E(\{q_0\})$ in the construction of M .

The construction works because at every step, M enters a state that corresponds to the subset of states that N could be in at that point.

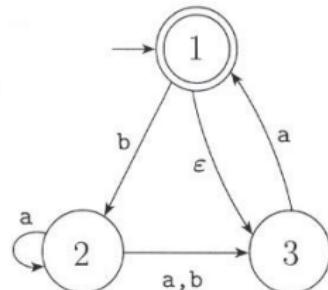
Example:

The NFA N_4 from earlier today:



Example:

The NFA N_4 from earlier today:



The construction in the proof of Theorem 1.39 gives:

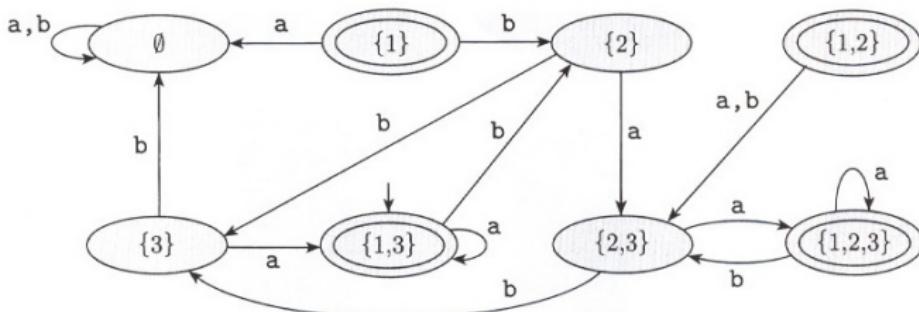


FIGURE 1.43

A DFA D that is equivalent to the NFA N_4

Equivalence of NFA and DFA (cont.)

Remark: The DFA we just obtained can be simplified by removing unnecessary states:

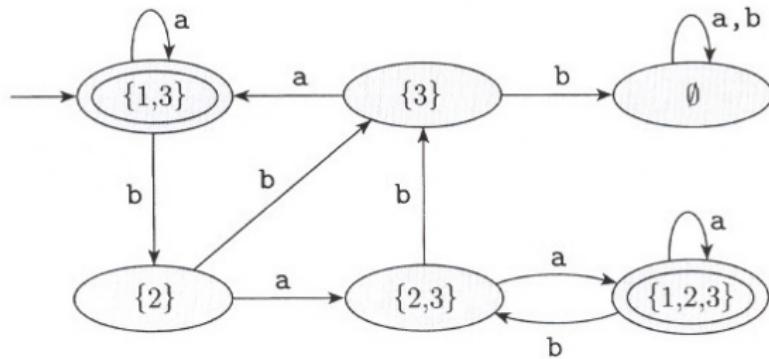


FIGURE 1.44
DFA D after removing unnecessary states

(From last week)

Definition (1.23)

Let A and B be languages. The three *regular operations* are defined as follows:

-  **Union:** $A \cup B = \{x \mid x \in A \text{ or } x \in B\}$.
-  **Concatenation:** $A \circ B = \{xy \mid x \in A \text{ and } y \in B\}$.
-  **Star:** $A^* = \{x_1x_2\dots x_k \mid k \geq 0 \text{ and each } x_i \in A\}$.

EXAMPLE 1.24

Let the alphabet Σ be the standard 26 letters $\{a, b, \dots, z\}$. If $A = \{\text{good}, \text{bad}\}$ and $B = \{\text{boy}, \text{girl}\}$, then

(From last week)

Definition (1.23)

Let A and B be languages. The three *regular operations* are defined as follows:

-  **Union:** $A \cup B = \{x \mid x \in A \text{ or } x \in B\}$.
-  **Concatenation:** $A \circ B = \{xy \mid x \in A \text{ and } y \in B\}$.
-  **Star:** $A^* = \{x_1 x_2 \dots x_k \mid k \geq 0 \text{ and each } x_i \in A\}$.

EXAMPLE 1.24

Let the alphabet Σ be the standard 26 letters $\{a, b, \dots, z\}$. If $A = \{\text{good}, \text{bad}\}$ and $B = \{\text{boy}, \text{girl}\}$, then

$$A \cup B = \{\text{good}, \text{bad}, \text{boy}, \text{girl}\},$$

$$A \circ B = \{\text{goodboy}, \text{goodgirl}, \text{badboy}, \text{badgirl}\}, \text{ and}$$

$$A^* = \{\varepsilon, \text{good}, \text{bad}, \text{goodgood}, \text{goodbad}, \text{badgood}, \text{badbad}, \\ \text{goodgoodgood}, \text{goodgoodbad}, \text{goodbadgood}, \text{goodbadbad}, \dots\}.$$

(From last week)

Theorem (1.25)

The class of regular languages is closed under the union operation. In other words, if A_1 and A_2 are regular languages, so is $A_1 \cup A_2$.

- ➊ Suppose $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognizes A_1 and $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ recognizes A_2 .
- ➋ Construct $M = (Q, \Sigma, \delta, q_0, F)$ to recognize $A_1 \cup A_2$:
 1. $Q = \{(r_1, r_2) \mid r_1 \in Q_1 \text{ and } r_2 \in Q_2\}$.
 2. Σ is the same. (Generalization is possible.)
 3. For each $(r_1, r_2) \in Q$ and each $a \in \Sigma$, let $\delta((r_1, r_2), a) = (\delta_1(r_1, a), \delta_2(r_2, a))$.
 4. $q_0 = (q_1, q_2)$.
 5. $F = \{(r_1, r_2) \mid r_1 \in F_1 \text{ or } r_2 \in F_2\}$.

Closedness under Union

We can use nondeterminism to get an alternative proof of Theorem 1.25:

Closedness under Union

We can use nondeterminism to get an alternative proof of Theorem 1.25:

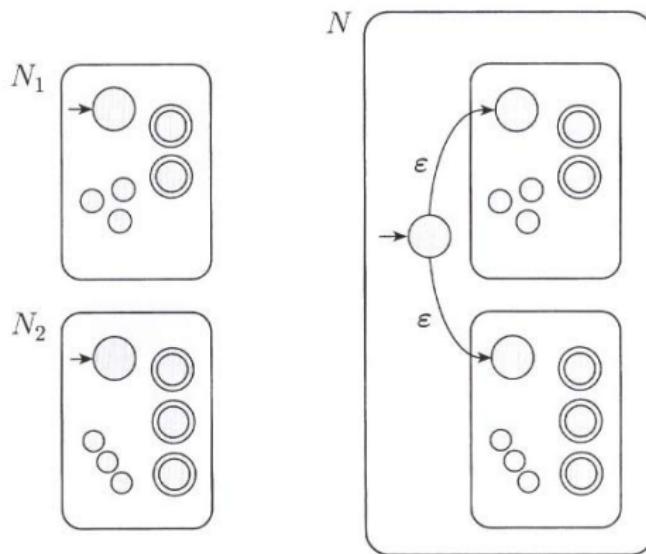


FIGURE 1.46

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

Closedness under Union (cont.)

Theorem (1.45)

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

Proof: Let $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ be an NFA that recognizes A_1 , and $N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ an NFA that recognizes A_2 .

Construct an NFA $N = (Q, \Sigma, \delta, q_0, F)$ that recognizes $A_1 \cup A_2$ as follows.

Closedness under Union (cont.)

Theorem (1.45)

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

Proof: Let $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ be an NFA that recognizes A_1 , and $N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ an NFA that recognizes A_2 .

Construct an NFA $N = (Q, \Sigma, \delta, q_0, F)$ that recognizes $A_1 \cup A_2$ as follows.

1. $Q = \{q_0\} \cup Q_1 \cup Q_2$.
2. $q_0 (\notin Q_1 \cup Q_2)$ is the start state.
3. For $q \in Q$ and $a \in \Sigma_\varepsilon$,

$$\delta(q, a) = \begin{cases} \delta_1(q, a) & q \in Q_1 \\ \delta_2(q, a) & q \in Q_2 \\ \{q_1, q_2\} & q = q_0 \text{ and } a = \varepsilon \\ \emptyset & q = q_0 \text{ and } a \neq \varepsilon \end{cases}$$

4. $F = F_1 \cup F_2$.

Closedness under Concatenation

We can also use nondeterminism to prove “closed under \circ ”:

Closedness under Concatenation

We can also use nondeterminism to prove “closed under \circ ”:

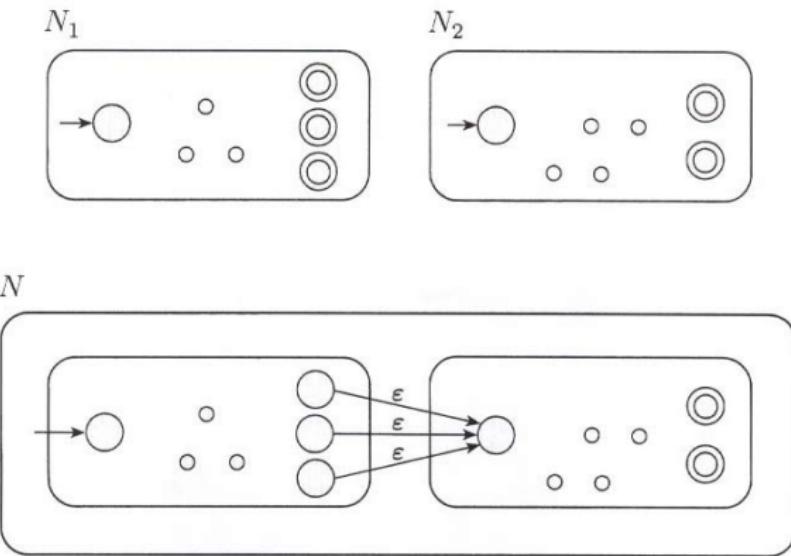


FIGURE 1.48
Construction of N to recognize $A_1 \circ A_2$

Closedness under Concatenation (cont.)

Theorem (1.47)

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

Proof: Let $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ be an NFA that recognizes A_1 , and $N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ an NFA that recognizes A_2 .

Construct an NFA $N = (Q, \Sigma, \delta, q_1, F_2)$ that recognizes $A_1 \circ A_2$ as follows.

Closedness under Concatenation (cont.)

Theorem (1.47)

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

Proof: Let $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ be an NFA that recognizes A_1 , and $N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ an NFA that recognizes A_2 .

Construct an NFA $N = (Q, \Sigma, \delta, q_1, F_2)$ that recognizes $A_1 \circ A_2$ as follows.

1. $Q = Q_1 \cup Q_2$.
2. For $q \in Q$ and $a \in \Sigma_\epsilon$,

$$\delta(q, a) = \begin{cases} \delta_1(q, a) & q \in Q_1 \text{ but } q \notin F_1 \\ \delta_1(q, a) & q \in F_1 \text{ and } a \neq \epsilon \\ \delta_1(q, a) \cup \{q_2\} & q \in F_1 \text{ and } a = \epsilon \\ \delta_2(q, a) & q \in Q_2 \end{cases}$$

Closedness under Star

Finally, we can use nondeterminism to prove “closed under *”:

Closedness under Star

Finally, we can use nondeterminism to prove “closed under *”:

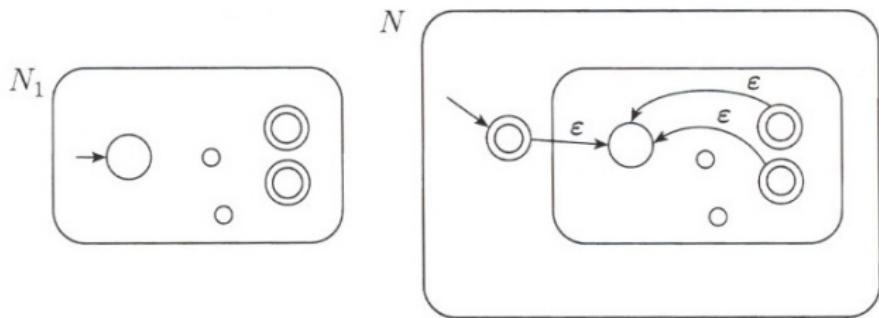


FIGURE 1.50
Construction of N to recognize A^*

Closedness under Star (cont.)

Theorem (1.49)

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

Proof: Let $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ be an NFA that recognizes A .

Construct an NFA $N = (Q, \Sigma, \delta, q_0, F)$ that recognizes A^* as follows.

Closedness under Star (cont.)

Theorem (1.49)

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

Proof: Let $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ be an NFA that recognizes A .

Construct an NFA $N = (Q, \Sigma, \delta, q_0, F)$ that recognizes A^* as follows.

1. $Q = \{q_0\} \cup Q_1$.
2. For $q \in Q$ and $a \in \Sigma_\epsilon$,

$$\delta(q, a) = \begin{cases} \delta_1(q, a) & q \in Q_1 \text{ but } q \notin F_1 \\ \delta_1(q, a) & q \in F_1 \text{ and } a \neq \epsilon \\ \delta_1(q, a) \cup \{q_1\} & q \in F_1 \text{ and } a = \epsilon \\ \{q_1\} & q = q_0 \text{ and } a = \epsilon \\ \emptyset & q = q_0 \text{ and } a \neq \epsilon \end{cases}$$

3. $F = \{q_0\} \cup F_1$.

Regular operations, summary

Theorem (1.45)

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

Theorem (1.47)

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

Theorem (1.49)

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