

Closure under the Regular Operations

Application of NFA

- Now we use the NFA to show that the collection of regular languages is closed under regular operations union, concatenation, and star

Application of NFA

- Now **we use the NFA to show that** the collection of regular languages is closed under regular operations union, concatenation, and star
- Earlier we have **shown this closure for union** using a Cartesian product of DFA

Application of NFA

- Now we use the NFA to show that the collection of regular languages is closed under regular operations union, concatenation, and star
- Earlier we have shown this closure for union using a Cartesian product of DFA
- For uniformity reason we reconstruct that proof using NFA

Theorem 1.45

The class of regular languages is closed under the union operation

Theorem 1.45

The class of regular languages is closed under the union operation

Proof idea:

Theorem 1.45

The class of regular languages is closed under the union operation

Proof idea:

- Let regular languages A_1 and A_2 be recognized by NFA N_1 and N_2 , respectively

Theorem 1.45

The class of regular languages is closed under the union operation

Proof idea:

- Let regular languages A_1 and A_2 be recognized by NFA N_1 and N_2 , respectively
- To show that $A_1 \cup A_2$ is regular we will construct an NFA N that recognizes $A_1 \cup A_2$

Theorem 1.45

The class of regular languages is closed under the union operation

Proof idea:

- Let regular languages A_1 and A_2 be recognized by NFA N_1 and N_2 , respectively
- To show that $A_1 \cup A_2$ is regular we will construct an NFA N that recognizes $A_1 \cup A_2$
- N must accept its input if either N_1 or N_2 accepts its input. Hence, N must have a new state that will allow it to guess nondeterministically which of N_1 or N_2 accepts it

Theorem 1.45

The class of regular languages is closed under the union operation

Proof idea:

- Let regular languages A_1 and A_2 be recognized by NFA N_1 and N_2 , respectively
- To show that $A_1 \cup A_2$ is regular we will construct an NFA N that recognizes $A_1 \cup A_2$
- N must accept its input if either N_1 or N_2 accepts its input. Hence, N must have a new state that will allow it to guess nondeterministically which of N_1 or N_2 accepts it
- Guessing is implemented by ϵ transitions from the new state to the start states of N_1 and N_2 , as seen in Figure 1

An NFA recognizing $A_1 \cup A_2$

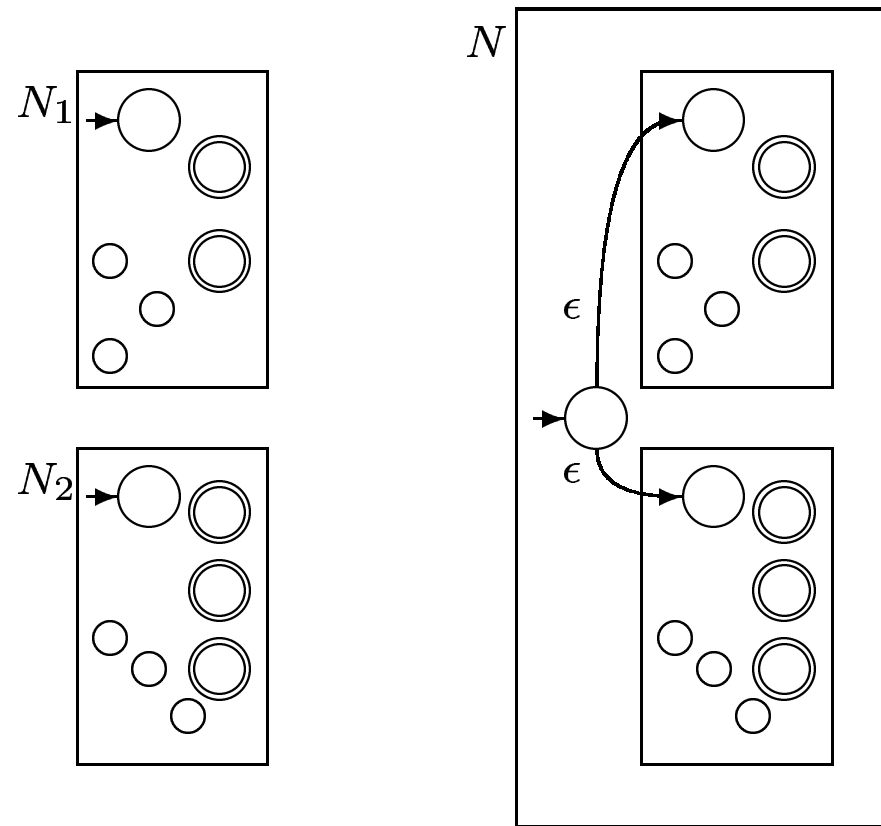


Figure 1: Construction of N to recognize $A_1 \cup A_2$

Proof

Let $N_1 = (Q_1, \Sigma, \delta_1, q_0^1, F_1)$, and $N_2 = (Q_2, \Sigma, \delta_2, q_0^2, F_2)$

Construct $N = (Q, \Sigma, \delta, q_0, F)$ to recognize $A_1 \cup A_2$ using the following procedure:

Construction procedure

1. $Q = \{q_0\} \cup Q_1 \cup Q_2$: That is, the states of N are all states on N_1 and N_2 with the addition of a new state q_0

Construction procedure

1. $Q = \{q_0\} \cup Q_1 \cup Q_2$: That is, the states of N are all states on N_1 and N_2 with the addition of a new state q_0
2. The **start state** of N is q_0

Construction procedure

1. $Q = \{q_0\} \cup Q_1 \cup Q_2$: That is, the states of N are all states on N_1 and N_2 with the addition of a new state q_0
2. The **start state** of N is q_0
3. The **accept states** of N are $F = F_1 \cup F_2$: That is, the accept states of N are all the accept states of N_1 and N_2

Construction procedure

1. $Q = \{q_0\} \cup Q_1 \cup Q_2$: That is, the states of N are all states on N_1 and N_2 with the addition of a new state q_0
2. The **start state** of N is q_0
3. The **accept states** of N are $F = F_1 \cup F_2$: That is, the accept states of N are all the accept states of N_1 and N_2
4. Define δ so that for any $q \in Q$ and any $a \in \Sigma_\epsilon$:

$$\delta(q, a) = \begin{cases} \delta_1(q, a), & \text{if } q \in Q_1 \\ \delta_2(q, a), & \text{if } q \in Q_2 \\ \{q_0^1, q_0^2\}, & \text{if } q = q_0 \text{ and } a = \epsilon \\ \emptyset, & \text{if } q = q_0 \text{ and } a \neq \epsilon. \end{cases}$$

Application

Consider the alphabet $\Sigma = \{0, 1\}$ and the languages:

$A = \{w \mid w \text{ begins with } 1 \text{ and ends with } 0\}$

$B = \{w \mid w \text{ contains at least three } 1\}$

$C = \{w \mid w = x0101y, x, y \in \Sigma^*\}$

$D = \{w \mid w \text{ does not contain the substring } 110\}$

Use the construction given in the proof of theorem 1.45 to give the state diagrams recognizing the languages $A \cup B$ and $C \cup D$.

Theorem 1.47

The class of regular languages is closed under concatenation operation

Proof idea: Assume two regular languages, A_1 and A_2 recognized by NFAs N_1 and N_2 , respectively. Construct N as suggested in Figure 2

Construction of NFA N

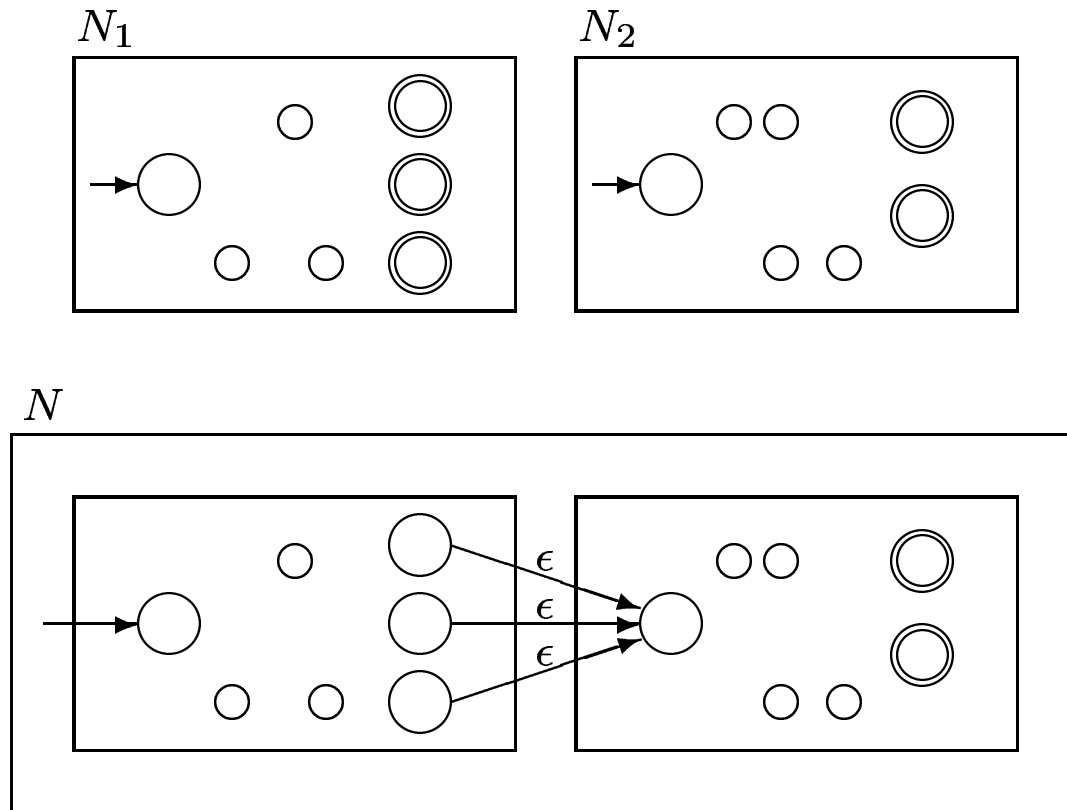


Figure 2: Construction of N to recognize $A_1 \circ A_2$

Construction procedure

- Combine N_1 and N_2 into a new automaton N that starts in the start state of N_1

Construction procedure

- Combine N_1 and N_2 into a new automaton N that starts in the start state of N_1
- Add ϵ transitions from the accept states of N_1 to the start state of N_2

Construction procedure

- Combine N_1 and N_2 into a new automaton N that starts in the start state of N_1
- Add ϵ transitions from the accept states of N_1 to the start state of N_2
- Set accept states of N to be the accept states on N_2

Proof

Let $N_1 = (Q_1, \Sigma, \delta_1, q_0^1, F_1)$ recognize A_1 and $N_2 = (Q_2, \Sigma, \delta_2, q_0^2, F_2)$ recognize A_2 .

Construct $N = (Q, \Sigma, \delta, q_0^1, F_2)$ by the following procedure:

Construction procedure

1. $Q = Q_1 \cup Q_2$. The states of N are all states of N_1 and N_2

Construction procedure

1. $Q = Q_1 \cup Q_2$. The states of N are all states of N_1 and N_2
2. The **start state** is the state q_0^1 of N_1

Construction procedure

1. $Q = Q_1 \cup Q_2$. The states of N are all states of N_1 and N_2
2. The **start state** is the state q_0^1 of N_1
3. The **accept states** is the set F_2 of the accept states of N_2

Construction procedure

1. $Q = Q_1 \cup Q_2$. The states of N are all states of N_1 and N_2
2. The **start state** is the state q_0^1 of N_1
3. The **accept states** is the set F_2 of the accept states of N_2
4. Define δ so that for any $q \in Q$ and any $a \in \Sigma_\epsilon$:

$$\delta(q, a) = \begin{cases} \delta_1(q, a), & \text{if } q \in Q_1 \text{ and } q \notin F_1 \\ \delta_1(q, a), & \text{if } q \in F_1 \text{ and } a \neq \epsilon \\ \delta_1(q, a) \cup \{q_0^2\}, & \text{if } q \in F_1 \text{ and } a = \epsilon \\ \delta_2(q, a), & \text{if } q \in Q_2. \end{cases}$$

Application

Consider the alphabet $\Sigma = \{0, 1\}$ and the languages:

$$A = \{w \mid |w| \leq 5\}$$

$$B = \{w \mid \text{every odd position of } w \text{ is } 1\}$$

$$C = \{w \mid w \text{ contains at least three } 1\}$$

$$D = \{\epsilon\}$$

Use the construction given in the proof of theorem 1.47 to give the state diagrams recognizing the languages $A \circ B$ and $C \circ D$ where \circ is concatenation operator.

Theorem 1.49

The class of regular languages is closed under star operation

Proof idea: we have a regular language A_1 , recognized by the NFA N_1 and want to prove that A_1^* is also a regular language.

The procedure to prove this theorem is by construction of the NFA N that recognizes A_1^* as shown in Figure 3

Procedure for the construction of N

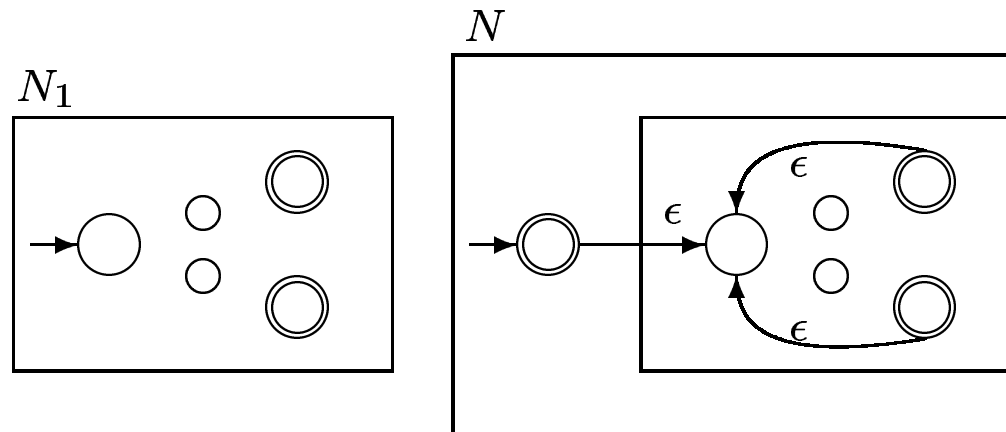


Figure 3: Construction of N to recognize A_1^*

More on the proof idea

- N is like N_1 with a new start state and an ϵ transition from the new start state to q_1

More on the proof idea

- N is like N_1 with a new start state and an ϵ transition from the new start state to q_1
- Since $\epsilon \in A_1^*$ the new start state is an accepts state

More on the proof idea

- N is like N_1 with a new start state and an ϵ transition from the new start state to q_1
- Since $\epsilon \in A_1^*$ the new start state is an accepts state
- We add ϵ transitions from the previous accept states of N_1 to the start state of N_1 allowing the machine to read and recognize strings of the form $w_1 \circ \dots \circ w_k$ where $w_1, \dots, w_k \in A_1$

Proof

Let $N_1 = (Q_1, \Sigma, \delta_1, q_0^1, F_1)$ recognize A_1 .

Construct $N = (Q, \Sigma, \delta, q_0, F)$ by the procedure:

Construction procedure

1. $Q = \{q_0\} \cup Q_1$; i.e. states of N_1 plus a new state q_0

Construction procedure

1. $Q = \{q_0\} \cup Q_1$; i.e. states of N_1 plus a new state q_0
2. Start state if N is q_0

Construction procedure

1. $Q = \{q_0\} \cup Q_1$; i.e. states of N_1 plus a new state q_0
2. **Start state** if N is q_0
3. $F = \{q_0\} \cup F_1$; that is, the **accept states** of N are the accept states of N_1 plus the new start state

Construction procedure

1. $Q = \{q_0\} \cup Q_1$; i.e. states of N_1 plus a new state q_0
2. **Start state** if N is q_0
3. $F = \{q_0\} \cup F_1$; that is, the **accept states** of N are the accept states of N_1 plus the new start state
4. Define δ so that for any $q \in Q$ and $a \in \Sigma_\epsilon$:

$$\delta(q, a) = \begin{cases} \delta_1(q, a), & \text{if } q \in Q_1 \text{ and } q \notin F_1 \\ \delta_1(q, a), & \text{if } q \in F_1 \text{ and } a \neq \epsilon \\ \delta_1(q, a) \cup \{q_0^1\}, & \text{if } q \in F_1 \text{ and } a = \epsilon \\ \{q_0^1\}, & \text{if } q = q_0 \text{ and } a = \epsilon \\ \emptyset, & \text{if } q = q_0 \text{ and } a \neq \epsilon. \end{cases}$$

Application

Consider the alphabet $\Sigma = \{0, 1\}$ and the languages:

$A = \{w \mid w \text{ contains at least three } 1s\}$

$B = \{w \mid w \text{ contains at least two } 0s \text{ and at most one } 1\}$

$C = \{\epsilon\}$

Use the construction given in the proof of theorem 1.49 to give the state diagrams recognizing the languages A^* , B^* and C^* .

Closure under complementation

- The class of regular languages is closed under complementation.

Closure under complementation

- The class of regular languages is closed under complementation.
- For that we will first show that if M is a DFA that recognizes a language B , swapping the accept and non-accept states in M yields a new DFA that recognizes the complement of B .

Proof

Let M' be the DFA M with accept and non-accept states swapped. We will show that M' recognizes the complement of B

Proof

Let M' be the DFA M with accept and non-accept states swapped. We will show that M' recognizes the complement of B

1. suppose M' accept x , i.e., if we run M' on x we end in an accept state of M'

Proof

Let M' be the DFA M with accept and non-accept states swapped. We will show that M' recognizes the complement of B

1. Suppose M' accept x , i.e., if we run M' on x we end in an accept state of M'
2. Because M and M' have swapped accept/non-accept states, if we run M on x we would end in a non-accept state. Hence, $x \notin B$

Proof

Let M' be the DFA M with accept and non-accept states swapped. We will show that M' recognizes the complement of B

1. Suppose M' accept x , i.e., if we run M' on x we end in an accept state of M'
2. Because M and M' have swapped accept/non-accept states, if we run M on x we would end in a non-accept state. Hence, $x \notin B$
3. Similarly, if x is not accepted by M' , it would be accepted by M

Proof

Let M' be the DFA M with accept and non-accept states swapped. We will show that M' recognizes the complement of B

1. Suppose M' accept x , i.e., if we run M' on x we end in an accept state of M'
2. Because M and M' have swapped accept/non-accept states, if we run M on x we would end in a non-accept state. Hence, $x \notin B$
3. Similarly, if x is not accepted by M' , it would be accepted by M

Consequently, M' accepts those strings that are not accepted by M and therefore M' recognizes the complement of B .

Conclusion

- B has been an arbitrary regular language. Therefore, our construction shows how to build an automaton to recognize its complement

Conclusion

- B has been an arbitrary regular language. Therefore, our construction shows how to build an automaton to recognize its complement
- Hence, the complement of any regular language is also regular

Conclusion

- B has been an arbitrary regular language. Therefore, our construction shows how to build an automaton to recognize its complement
- Hence, the complement of any regular language is also regular
- Consequently the class of regular languages is closed under complementation

Interesting property

If M is an NFA that recognizes language C , swapping its accept and non-accept states doesn't necessarily yield a new NFA that recognizes the complement of C .

Proof

We prove the interesting property by constructing a counter-example.

- Consider the construction in Figure 4, where both NFA-s, M and M' , accept aa .

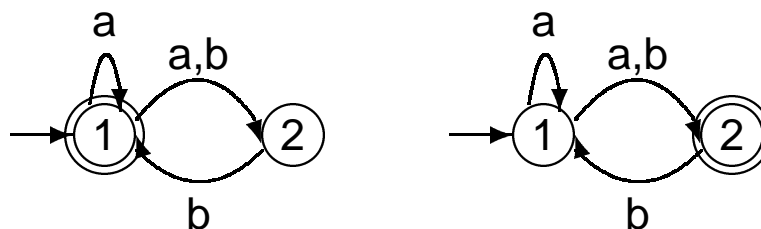


Figure 4: NFAs M and M'

Question

Is the class of languages recognized by NFAs closed under complementation?

Closure under complementation

- The class of languages recognized by NFA is still closed under complementation

Closure under complementation

- The class of languages recognized by NFA is still closed under complementation
- This follows from the fact that the class of languages recognized by NFAs is precisely the class of languages recognized by DFA

Closure under complementation

- The class of languages recognized by NFA is still closed under complementation
- This follows from the fact that the class of languages recognized by NFAs is precisely the class of languages recognized by DFA
- The counter-example in Figure 4 shows the difference between the process of computations performed by DFAs and NFAs