COL 352 Introduction to Automata and Theory of Computation

Nikhil Balaji

Bharti 420
Indian Institute of Technology, Delhi
nbalaji@cse.iitd.ac.in

January 9, 2023

Lecture 3: More on DFAs, Operations on Regular sets

Formal definition of DFA

Definition (DFA)

A deterministic finite state automaton (DFA) $A = (Q, \Sigma, q_0, F, \delta)$, where

Q is a set of states,

Σ is the input alphabet,

$q_0 \in Q$ is the initial state,

$F \subseteq Q$ is the set of final states,

δ is a set of transitions, i.e. $\delta : Q \times \Sigma \to Q$ or
$\delta \subseteq Q \times \Sigma \times Q$ such that
$\forall q \in Q, \forall a \in \Sigma,\ |\delta(q, a)| \leq 1.$

Acceptance by DFA

Definition (Run of a DFA, Acceptance by DFA)

Acceptance by DFA

Definition (Run of a DFA, Acceptance by DFA)

Let $A = (Q, \Sigma, \delta, q_0, F)$ be a DFA. A run of A on word $w = a_1 \dots a_n$ is a sequence of states $q_0, \dots, q_n$ such that $q_i = \delta(q_{i-1}, a_i)$ for all $1 \leq i \leq n$.

A word w is accepted by DFA A if there is a run of A on word w that reaches (ends in) an accepting state.

Extended Transition Function

Let $\hat{\delta} : Q \times \Sigma^* \to Q$ be defined as:

$$\begin{array}{rcl} \hat{\delta}(q, \varepsilon) & = & q \\ \hat{\delta}(q, wa) & = & \delta(\hat{\delta}(q, w), a) \end{array}$$

$$L(A) = \{w \in \Sigma^* \mid \hat{\delta}(q_0, w) \in F\}$$

L is a regular language if there exists some DFA A such that $L(A) = L$

String Theory

In physics, string theory is a theoretical framework in which the point-like particles of particle physics are replaced by one-dimensional objects called strings. String theory describes how these strings propagate through space and interact with each other. On distance scales larger than the string scale, a string looks just like an ordinary particle, with its mass, charge, and other properties determined by the vibrational state of the string. In string theory, one of the many vibrational states of the string corresponds to the graviton, a quantum mechanical particle that carries the gravitational force. Thus, string theory is a theory of quantum gravity.

String Theory of CS

String Theory of CS:Encodings objects as strings

- Numbers can be encoded as strings.
- Graphs can be encoded as strings.
- Programs (DFAs) can be encoded as strings!
- $A = (Q, \Sigma, \delta, q_0, F)$ can be encoded as a finite length string.

String Theory of CS:Encodings objects as strings

- Numbers can be encoded as strings.
- Graphs can be encoded as strings.
- Programs (DFAs) can be encoded as strings!
- $A = (Q, \Sigma, \delta, q_0, F)$ can be encoded as a finite length string.
- Assume DFA has k states and works over $\Sigma = \{0, 1\}$.

String Theory of CS:Encodings objects as strings

- Numbers can be encoded as strings.
- Graphs can be encoded as strings.
- Programs (DFAs) can be encoded as strings!
- $A = (Q, \Sigma, \delta, q_0, F)$ can be encoded as a finite length string.
- Assume DFA has k states and works over $\Sigma = \{0, 1\}$.
- Q requires

String Theory of CS:Encodings objects as strings

- Numbers can be encoded as strings.
- Graphs can be encoded as strings.
- Programs (DFAs) can be encoded as strings!
- $A = (Q, \Sigma, \delta, q_0, F)$ can be encoded as a finite length string.
- Assume DFA has k states and works over $\Sigma = \{0, 1\}$.
- Q requires $O(k \log k)$ bits, δ requires

String Theory of CS:Encodings objects as strings

- Numbers can be encoded as strings.
- Graphs can be encoded as strings.
- Programs (DFAs) can be encoded as strings!
- $A = (Q, \Sigma, \delta, q_0, F)$ can be encoded as a finite length string.
- Assume DFA has k states and works over $\Sigma = \{0, 1\}$.
- Q requires $O(k \log k)$ bits, δ requires $O(k \log k)$ bits, q_0, F together require another

String Theory of CS:Encodings objects as strings

- Numbers can be encoded as strings.
- Graphs can be encoded as strings.
- Programs (DFAs) can be encoded as strings!
- $A = (Q, \Sigma, \delta, q_0, F)$ can be encoded as a finite length string.
- Assume DFA has k states and works over $\Sigma = \{0, 1\}$.
- Q requires $O(k \log k)$ bits, δ requires $O(k \log k)$ bits, q_0, F together require another $O(k \log k)$ bits.

String Theory of CS:Encodings objects as strings

- Numbers can be encoded as strings.
- Graphs can be encoded as strings.
- Programs (DFAs) can be encoded as strings!
- $A = (Q, \Sigma, \delta, q_0, F)$ can be encoded as a finite length string.
- Assume DFA has k states and works over $\Sigma = \{0, 1\}$.
- Q requires $O(k \log k)$ bits, δ requires $O(k \log k)$ bits, q_0, F together require another $O(k \log k)$ bits. So overall $O(k \log k)$ bits are sufficient to encode a k-state DFA.

Theorem

There exists a language for which there is no DFA accepting it.

- Consider an enumeration of DFAs (they are countably infinite).
- Each DFA accepts a unique language over $\{0, 1\}^*$
- Number of languages $L \subseteq \{0, 1\}^*$ is uncountable $(= 2^{\Sigma^*} \equiv 2^{\mathbb{N}} \equiv \mathbb{R})$.

Finite state automata

Example 4

Input: $w \in \{0,1\}^*$

Finite state automata

Example 4

Input: $w \in \{0,1\}^*$

Check: is the number represented by w in binary a multiple of 3?

$$L_3 = \{w \in \{0,1\}^* \mid w \text{ encodes a number in binary divisible by } 3\}$$

Finite state automata

Example 4

Input: $w \in \{0,1\}^*$

Check: is the number represented by w in binary a multiple of 3?

$$L_3 = \{w \in \{0,1\}^* \mid w \text{ encodes a number in binary divisible by } 3\}$$

Idea 1: Possible remainders are $\{0,1,2\}$. Represent them as states!

Finite state automata

Example 4

Input: $w \in \{0,1\}^*$

Check: is the number represented by w in binary a multiple of 3?

$$L_3 = \{w \in \{0,1\}^* \mid w \text{ encodes a number in binary divisible by } 3\}$$

Idea 1: Possible remainders are $\{0,1,2\}$. Represent them as states!

Idea 2: If you read a bit c at a state q then go to state $2q + c \pmod 3$.

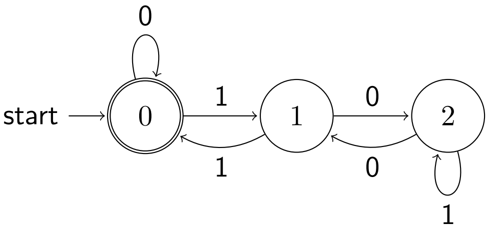
Finite state automata

Example 4

Input: $w \in \{0,1\}^*$

Check: is the number represented by w in binary a multiple of 3?

$$L_3 = \{w \in \{0,1\}^* \mid w \text{ encodes a number in binary divisible by } 3\}$$



Finite state automata

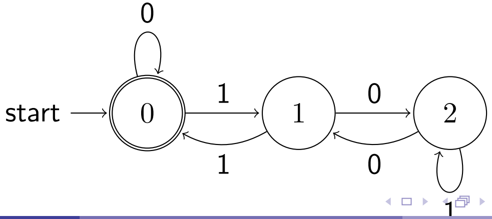
Example 4

Input: $w \in \{0,1\}^*$

Check: is the number represented by w in binary a multiple of 3?

$$L_3 = \{w \in \{0,1\}^* \mid w \text{ encodes a number in binary divisible by } 3\}$$

Prove: Automaton below exactly accepts the above language.



Proof of Correctness

Proof Idea: By induction on length of the input string.

Proof of Correctness

Proof Idea: By induction on length of the input string.
Let $\#x$:= number represented by string x in binary.

$$\begin{aligned}
\#\varepsilon &= 0 \\
\#0 &= 0 \\
\#11 &= 3 \\
\#100 &= 4
\end{aligned}$$

Proof of Correctness

Proof Idea: By induction on length of the input string.
For any string $x \in \{0,1\}^*$,

$$
\begin{aligned}
\hat{\delta}(0,x) &= 0 \iff \#x = 0 \pmod 3 \\
\hat{\delta}(0,x) &= 1 \iff \#x = 1 \pmod 3 \\
\hat{\delta}(0,x) &= 2 \iff \#x = 2 \pmod 3
\end{aligned}
$$

Proof of Correctness

Proof Idea: By induction on length of the input string.
Induction hypothesis: For any string $x \in \{0,1\}^*$, $\hat{\delta}(0,x) = x \pmod 3$
Special case: For string $x \in L_3$, $\hat{\delta}(0,x) = 0 \pmod 3$

Proof of Correctness

Proof Idea: By induction on length of the input string.
Induction hypothesis: For any string $x \in \{0,1\}^*$, $\hat{\delta}(0,x) = x \pmod 3$
Special case: For string $x \in L_3$, $\hat{\delta}(0,x) = 0 \pmod 3$

Proof of Correctness

Proof Idea: By induction on length of the input string.
Induction hypothesis: For any string $x \in \{0,1\}^*$, $\hat{\delta}(0,x) = x \pmod 3$
Special case: For string $x \in L_3$, $\hat{\delta}(0,x) = 0 \pmod 3$

$$
\begin{aligned}
\#(x0) &= 2(\#x) + 0 \\
\#(x1) &= 2(\#x) + 1
\end{aligned}
$$

Proof of Correctness

Proof Idea: By induction on length of the input string.
Induction hypothesis: For any string $x \in \{0,1\}^*$, $\hat{\delta}(0,x) = x \pmod 3$
Special case: For string $x \in L_3$, $\hat{\delta}(0,x) = 0 \pmod 3$

$$\begin{aligned}\#(x0) &= 2(\#x) + 0 \\ \#(x1) &= 2(\#x) + 1\end{aligned}$$

$$\begin{aligned}\delta(q,0) &= 2q \pmod 3 \\ \delta(q,1) &= 2q + 1 \pmod 3\end{aligned}$$

Proof of Correctness

Basis:

For $x = \varepsilon$,

Proof of Correctness

Basis:

For $x = \varepsilon$,

$$\hat{\delta}(0, \varepsilon) \quad = \quad 0 \quad \text{(by definition of } \hat{\delta}\text{)}$$

Proof of Correctness

Basis:

For $x = \varepsilon$,

$$\begin{aligned} \hat{\delta}(0, \varepsilon) &= 0 \quad \text{(by definition of } \hat{\delta}\text{)} \\ &= \#\varepsilon \quad \text{(since } \#\varepsilon = 0\text{)} \end{aligned}$$

Proof of Correctness

Basis:

For $x = \varepsilon$,

$$\begin{aligned} \hat{\delta}(0, \varepsilon) &= 0 \quad \text{(by definition of } \hat{\delta}\text{)} \\ &= \#\varepsilon \quad \text{(since } \#\varepsilon = 0\text{)} \\ &= \#\varepsilon \pmod 3 \end{aligned}$$

Proof of Correctness

Basis:

For $x = \varepsilon$,

$$\begin{aligned}\hat{\delta}(0, \varepsilon) &= 0 \quad \text{(by definition of } \hat{\delta}\text{)} \\ &= \#\varepsilon \quad \text{(since } \#\varepsilon = 0\text{)} \\ &= \#\varepsilon \pmod 3\end{aligned}$$

Induction step: Assume that $\hat{\delta}(0, x) = \#x \pmod 3$ is true for $x \in \{0, 1\}^*$

Proof of Correctness

Basis:

For $x = \varepsilon$,

$$\begin{aligned}\hat{\delta}(0,\varepsilon) &= 0 \quad \text{(by definition of } \hat{\delta}\text{)} \\ &= \#\varepsilon \quad \text{(since } \#\varepsilon = 0\text{)} \\ &= \#\varepsilon \pmod 3\end{aligned}$$

Induction step: Assume that $\hat{\delta}(0,x) = \#x \pmod 3$ is true for $x \in \{0,1\}^*$ show that it is true for xc where $c \in \{0,1\}$.

Proof of Correctness

Basis:

For $x = \varepsilon$,

$$\begin{aligned}\hat{\delta}(0, \varepsilon) &= 0 \quad \text{(by definition of } \hat{\delta}\text{)} \\ &= \#\varepsilon \quad \text{(since } \#\varepsilon = 0\text{)} \\ &= \#\varepsilon \pmod 3\end{aligned}$$

Induction step: Assume that $\hat{\delta}(0, x) = \#x \pmod 3$ is true for $x \in \{0, 1\}^*$ show that it is true for xc where $c \in \{0, 1\}$.

$$\hat{\delta}(0, xc) = \delta(\hat{\delta}(0, x), c)$$

Proof of Correctness

Basis:

For $x = \varepsilon$,

$$\begin{aligned}\hat{\delta}(0,\varepsilon) &= 0 \quad \text{(by definition of } \hat{\delta}\text{)}\\ &= \#\varepsilon \quad \text{(since } \#\varepsilon = 0\text{)}\\ &= \#\varepsilon \pmod 3\end{aligned}$$

Induction step: Assume that $\hat{\delta}(0,x) = \#x \pmod 3$ is true for $x \in \{0,1\}^*$ show that it is true for xc where $c \in \{0,1\}$.

$$\begin{aligned}\hat{\delta}(0,xc) &= \delta(\hat{\delta}(0,x),c)\\ &= \delta(\#x \mod 3, c)\end{aligned}$$

Proof of Correctness

Basis:

For $x = \varepsilon$,

$$\begin{aligned}\hat{\delta}(0,\varepsilon) &= 0 \quad \text{(by definition of } \hat{\delta}\text{)}\\ &= \#\varepsilon \quad \text{(since } \#\varepsilon = 0\text{)}\\ &= \#\varepsilon \pmod 3\end{aligned}$$

Induction step: Assume that $\hat{\delta}(0,x) = \#x \pmod 3$ is true for $x \in \{0,1\}^*$ show that it is true for xc where $c \in \{0,1\}$.

$$\begin{aligned}\hat{\delta}(0,xc) &= \delta(\hat{\delta}(0,x),c)\\ &= \delta(\#x \mod 3, c)\\ &= (2(\#x)+c) \pmod 3\end{aligned}$$

Proof of Correctness

Basis:

For $x = \varepsilon$,

$$\begin{aligned}\hat{\delta}(0,\varepsilon) &= 0 \quad \text{(by definition of } \hat{\delta}\text{)}\\ &= \#\varepsilon \quad \text{(since } \#\varepsilon = 0\text{)}\\ &= \#\varepsilon \pmod 3\end{aligned}$$

Induction step: Assume that $\hat{\delta}(0,x) = \#x \pmod 3$ is true for $x \in \{0,1\}^*$ show that it is true for xc where $c \in \{0,1\}$.

$$\begin{aligned}\hat{\delta}(0,xc) &= \delta(\hat{\delta}(0,x),c)\\ &= \delta(\#x \mod 3, c)\\ &= (2(\#x)+c) \pmod 3\\ &= \#xc \pmod 3\end{aligned}$$

Building complicated DFAs from simple ones

Example

Let $\Sigma = \{a\}$ for this example.

Building complicated DFAs from simple ones

Example

Let $\Sigma = \{a\}$ for this example.

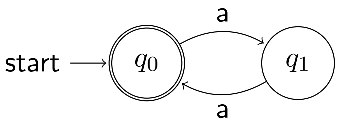
Let $L_1 = \{w \mid |w| \equiv 0 \pmod 2\}$

Building complicated DFAs from simple ones

Example

Let $\Sigma = \{a\}$ for this example.

Let $L_1 = \{w \mid |w| \equiv 0 \pmod 2\}$

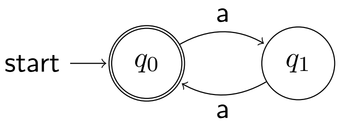


Building complicated DFAs from simple ones

Example

Let $\Sigma = \{a\}$ for this example.

Let $L_1 = \{w \mid |w| \equiv 0 \pmod 2\}$



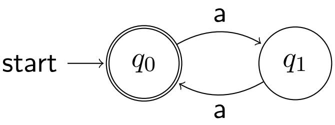
Let $L_2 = \{w \mid |w| \equiv 0 \pmod 3\}$

Building complicated DFAs from simple ones

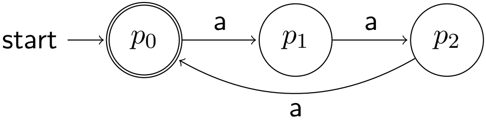
Example

Let $\Sigma = \{a\}$ for this example.

Let $L_1 = \{w \mid |w| \equiv 0 \pmod 2\}$



Let $L_2 = \{w \mid |w| \equiv 0 \pmod 3\}$

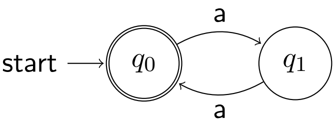


Building complicated DFAs from simple ones

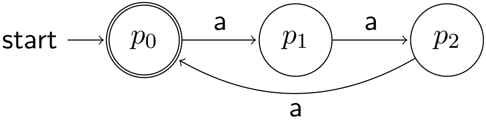
Example

Let $\Sigma = \{a\}$ for this example.

Let $L_1 = \{w \mid |w| \equiv 0 \pmod 2\}$



Let $L_2 = \{w \mid |w| \equiv 0 \pmod 3\}$



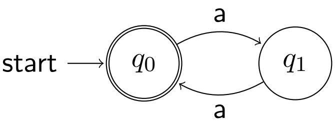
What is $L_1 \cap L_2$?

Building complicated DFAs from simple ones

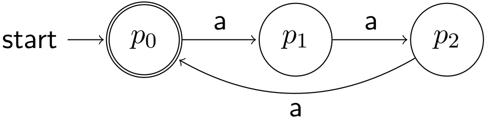
Example

Let $\Sigma = \{a\}$ for this example.

Let $L_1 = \{w \mid |w| \equiv 0 \pmod 2\}$



Let $L_2 = \{w \mid |w| \equiv 0 \pmod 3\}$



What is $L_1 \cap L_2$?

$L_1 \cap L_2 = \{w \mid |w| \equiv 0 \pmod 6\}$

Building complicated DFAs from simple ones

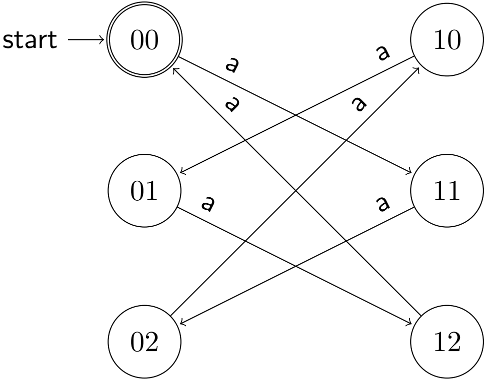
Example

$$L_1 \cap L_2 = \{w \mid |w| \equiv 0 \pmod 2\}$$

Building complicated DFAs from simple ones

Example

$$L_1 \cap L_2 = \{w \mid |w| \equiv 0 \pmod 2\}$$



Intersection of two regular languages

$$L' = \{w \in \{a,b\}^* \mid w \text{ contains aa}\}$$
$$L_{odd} = \{w \in \{a,b\}^* \mid w \text{ contains odd number of a}\}$$

What about $L_{odd} \cap L'$

Intersection of two regular languages

$$L' = \{w \in \{a,b\}^* \mid w \text{ contains aa}\}$$
$$L_{odd} = \{w \in \{a,b\}^* \mid w \text{ contains odd number of a}\}$$

What about $L_{odd} \cap L'$

Building complicated DFAs from simple ones

Lemma

Let $L_1, L_2 \subseteq \Sigma^*$ be two regular languages, then $L_1 \cap L_2$ is also a regular language.

Proof.

Product construction

Let $A_1 = (Q_1, \Sigma, q_0^1, F_1, \delta_1)$ and $A_2 = (Q_2, \Sigma, q_0^2, F_2, \delta_2)$ be the automata accepting L_1, L_2, respectively.

Building complicated DFAs from simple ones

Lemma

Let $L_1, L_2 \subseteq \Sigma^*$ be two regular languages, then $L_1 \cap L_2$ is also a regular language.

Proof.

Product construction

Let $A_1 = (Q_1, \Sigma, q_0^1, F_1, \delta_1)$ and $A_2 = (Q_2, \Sigma, q_0^2, F_2, \delta_2)$ be the automata accepting L_1, L_2, respectively.

Let A be a finite state automaton $(Q, \Sigma, q_0, F, \delta)$ s.t.

Building complicated DFAs from simple ones

Lemma

Let $L_1, L_2 \subseteq \Sigma^*$ be two regular languages, then $L_1 \cap L_2$ is also a regular language.

Proof.

Product construction

Let $A_1 = (Q_1, \Sigma, q_0^1, F_1, \delta_1)$ and $A_2 = (Q_2, \Sigma, q_0^2, F_2, \delta_2)$ be the automata accepting L_1, L_2, respectively.

Let A be a finite state automaton $(Q, \Sigma, q_0, F, \delta)$ s.t.

$$Q = \{(q, q') \mid q \in Q_1, q' \in Q_2\}$$

Building complicated DFAs from simple ones

Lemma

Let $L_1, L_2 \subseteq \Sigma^*$ be two regular languages, then $L_1 \cap L_2$ is also a regular language.

Proof.

Product construction

Let $A_1 = (Q_1, \Sigma, q_0^1, F_1, \delta_1)$ and $A_2 = (Q_2, \Sigma, q_0^2, F_2, \delta_2)$ be the automata accepting L_1, L_2, respectively.

Let A be a finite state automaton $(Q, \Sigma, q_0, F, \delta)$ s.t.

$$\begin{array}{lcl} Q & = & \{(q, q') \mid q \in Q_1, q' \in Q_2\} \\ q_0 & = & (q_0^1, q_0^2) \end{array}$$

Building complicated DFAs from simple ones

Lemma

Let $L_1, L_2 \subseteq \Sigma^*$ be two regular languages, then $L_1 \cap L_2$ is also a regular language.

Proof.

Product construction

Let $A_1 = (Q_1, \Sigma, q_0^1, F_1, \delta_1)$ and $A_2 = (Q_2, \Sigma, q_0^2, F_2, \delta_2)$ be the automata accepting L_1, L_2, respectively.

Let A be a finite state automaton $(Q, \Sigma, q_0, F, \delta)$ s.t.

$$\begin{array}{lcl} Q & = & \{(q, q') \mid q \in Q_1, q' \in Q_2\} \\ q_0 & = & (q_0^1, q_0^2) \\ F & = & \{(q, q') \mid q \in F_1, q' \in F_2\} \end{array}$$

Building complicated DFAs from simple ones

Lemma

Let $L_1, L_2 \subseteq \Sigma^*$ be two regular languages, then $L_1 \cap L_2$ is also a regular language.

Proof.

Product construction

Let $A_1 = (Q_1, \Sigma, q_0^1, F_1, \delta_1)$ and $A_2 = (Q_2, \Sigma, q_0^2, F_2, \delta_2)$ be the automata accepting L_1, L_2, respectively.

Let A be a finite state automaton $(Q, \Sigma, q_0, F, \delta)$ s.t.

$$\begin{array}{ccl} Q & = & \{(q, q') \mid q \in Q_1, q' \in Q_2\} \\ q_0 & = & (q_0^1, q_0^2) \\ F & = & \{(q, q') \mid q \in F_1, q' \in F_2\} \\ \delta((q, q'), a) & = & (\delta_1(q, a), \delta_2(q', a)) \end{array}$$

Building complicated DFAs from simple ones

Lemma

Let $L_1, L_2 \subseteq \Sigma^*$ be two regular languages, then $L_1 \cap L_2$ is also a regular language.

Proof.

Product construction

Let $A_1 = (Q_1, \Sigma, q_0^1, F_1, \delta_1)$ and $A_2 = (Q_2, \Sigma, q_0^2, F_2, \delta_2)$ be the automata accepting L_1, L_2, respectively.

Let A be a finite state automaton $(Q, \Sigma, q_0, F, \delta)$ s.t.

$$
\begin{array}{rcl}
Q & = & \{(q, q') \mid q \in Q_1, q' \in Q_2\} \\
q_0 & = & (q_0^1, q_0^2) \\
F & = & \{(q, q') \mid q \in F_1, q' \in F_2\} \\
\delta((q, q'), a) & = & (\delta_1(q, a), \delta_2(q', a))
\end{array}
$$

Correctness

$\forall w \in \Sigma^*$, w is accepted by A iff w is accepted by both A_1 and A_2.

Building complicated DFAs from simple ones

Lemma

Let $L_1, L_2 \subseteq \Sigma^*$ be two regular languages, then $L_1 \cup L_2$ is also a regular language.

Proof.

Product construction

Let $A_1 = (Q_1, \Sigma, q_0^1, F_1, \delta_1)$ and $A_2 = (Q_2, \Sigma, q_0^2, F_2, \delta_2)$ be the automata accepting L_1, L_2, respectively.

Building complicated DFAs from simple ones

Lemma

Let $L_1, L_2 \subseteq \Sigma^*$ be two regular languages, then $L_1 \cup L_2$ is also a regular language.

Proof.

Product construction

Let $A_1 = (Q_1, \Sigma, q_0^1, F_1, \delta_1)$ and $A_2 = (Q_2, \Sigma, q_0^2, F_2, \delta_2)$ be the automata accepting L_1, L_2, respectively.

Let A be a finite state automaton $(Q, \Sigma, q_0, F, \delta)$ s.t.

Building complicated DFAs from simple ones

Lemma

Let $L_1, L_2 \subseteq \Sigma^*$ be two regular languages, then $L_1 \cup L_2$ is also a regular language.

Proof.

Product construction

Let $A_1 = (Q_1, \Sigma, q_0^1, F_1, \delta_1)$ and $A_2 = (Q_2, \Sigma, q_0^2, F_2, \delta_2)$ be the automata accepting L_1, L_2, respectively.

Let A be a finite state automaton $(Q, \Sigma, q_0, F, \delta)$ s.t.

$$Q = \{(q, q') \mid q \in Q_1, q' \in Q_2\}$$

Building complicated DFAs from simple ones

Lemma

Let $L_1, L_2 \subseteq \Sigma^*$ be two regular languages, then $L_1 \cup L_2$ is also a regular language.

Proof.

Product construction

Let $A_1 = (Q_1, \Sigma, q_0^1, F_1, \delta_1)$ and $A_2 = (Q_2, \Sigma, q_0^2, F_2, \delta_2)$ be the automata accepting L_1, L_2, respectively.

Let A be a finite state automaton $(Q, \Sigma, q_0, F, \delta)$ s.t.

$$\begin{array}{lcl} Q & = & \{(q, q') \mid q \in Q_1, q' \in Q_2\} \\ q_0 & = & (q_0^1, q_0^2) \end{array}$$

Building complicated DFAs from simple ones

Lemma

Let $L_1, L_2 \subseteq \Sigma^*$ be two regular languages, then $L_1 \cup L_2$ is also a regular language.

Proof.

Product construction

Let $A_1 = (Q_1, \Sigma, q_0^1, F_1, \delta_1)$ and $A_2 = (Q_2, \Sigma, q_0^2, F_2, \delta_2)$ be the automata accepting L_1, L_2, respectively.

Let A be a finite state automaton $(Q, \Sigma, q_0, F, \delta)$ s.t.

$$\begin{array}{lcl} Q & = & \{(q, q') \mid q \in Q_1, q' \in Q_2\} \\ q_0 & = & (q_0^1, q_0^2) \\ F & = & \{(q, q') \mid q \in F_1 \text{ or } q' \in F_2\} \end{array}$$

Building complicated DFAs from simple ones

Lemma

Let $L_1, L_2 \subseteq \Sigma^*$ be two regular languages, then $L_1 \cup L_2$ is also a regular language.

Proof.

Product construction

Let $A_1 = (Q_1, \Sigma, q_0^1, F_1, \delta_1)$ and $A_2 = (Q_2, \Sigma, q_0^2, F_2, \delta_2)$ be the automata accepting L_1, L_2, respectively.

Let A be a finite state automaton $(Q, \Sigma, q_0, F, \delta)$ s.t.

$$
\begin{array}{rcl}
Q & = & \{(q, q') \mid q \in Q_1, q' \in Q_2\} \\
q_0 & = & (q_0^1, q_0^2) \\
F & = & \{(q, q') \mid q \in F_1 \text{ or } q' \in F_2\} \\
\delta((q, q'), a) & = & (\delta_1(q, a), \delta_2(q', a))
\end{array}
$$

Building complicated DFAs from simple ones

Lemma

Let $L_1, L_2 \subseteq \Sigma^*$ be two regular languages, then $L_1 \cup L_2$ is also a regular language.

Proof.

Product construction

Let $A_1 = (Q_1, \Sigma, q_0^1, F_1, \delta_1)$ and $A_2 = (Q_2, \Sigma, q_0^2, F_2, \delta_2)$ be the automata accepting L_1, L_2, respectively.

Let A be a finite state automaton $(Q, \Sigma, q_0, F, \delta)$ s.t.

$$\begin{array}{rcl} Q & = & \{(q, q') \mid q \in Q_1, q' \in Q_2\} \\ q_0 & = & (q_0^1, q_0^2) \\ F & = & \{(q, q') \mid q \in F_1 \text{ or } q' \in F_2\} \\ \delta((q, q'), a) & = & (\delta_1(q, a), \delta_2(q', a)) \end{array}$$

Correctness

$\forall w \in \Sigma^*$, w is accepted by A iff w is accepted by either A_1 or A_2.

Closure under complement

Lemma

Let $L \subseteq \Sigma^*$ be a regular language, then $\overline{L} = \{w \mid w \notin L\}$ is also a regular language.

Proof.

Let $A = (Q, \Sigma, q_0, F, \delta)$ be the automata accepting L.

Closure under complement

Lemma

Let $L \subseteq \Sigma^*$ be a regular language, then $\overline{L} = \{w \mid w \notin L\}$ is also a regular language.

Proof.

Let $A = (Q, \Sigma, q_0, F, \delta)$ be the automata accepting L.

Let A' be a finite state automaton $(Q', \Sigma', q_0', F', \delta')$ s.t.

Closure under complement

Lemma

Let $L \subseteq \Sigma^*$ be a regular language, then $\overline{L} = \{w \mid w \notin L\}$ is also a regular language.

Proof.

Let $A = (Q, \Sigma, q_0, F, \delta)$ be the automata accepting L.

Let A' be a finite state automaton $(Q', \Sigma', q_0', F', \delta')$ s.t.

$$Q' = Q$$

Closure under complement

Lemma

Let $L \subseteq \Sigma^*$ be a regular language, then $\overline{L} = \{w \mid w \notin L\}$ is also a regular language.

Proof.

Let $A = (Q, \Sigma, q_0, F, \delta)$ be the automata accepting L.

Let A' be a finite state automaton $(Q', \Sigma', q'_0, F', \delta')$ s.t.

$$\begin{aligned} Q' &= Q \\ q'_0 &= q_0 \end{aligned}$$

Closure under complement

Lemma

Let $L \subseteq \Sigma^*$ be a regular language, then $\overline{L} = \{w \mid w \notin L\}$ is also a regular language.

Proof.

Let $A = (Q, \Sigma, q_0, F, \delta)$ be the automata accepting L.

Let A' be a finite state automaton $(Q', \Sigma', q_0', F', \delta')$ s.t.

$$
\begin{aligned}
Q' &= Q \\
q_0' &= q_0 \\
F' &= \{q \in Q \mid q \notin F\}
\end{aligned}
$$

Closure under complement

Lemma

Let $L \subseteq \Sigma^*$ be a regular language, then $\overline{L} = \{w \mid w \notin L\}$ is also a regular language.

Proof.

Let $A = (Q, \Sigma, q_0, F, \delta)$ be the automata accepting L.

Let A' be a finite state automaton $(Q', \Sigma', q_0', F', \delta')$ s.t.

$$\begin{aligned} Q' &= Q \\ q_0' &= q_0 \\ F' &= \{q \in Q \mid q \notin F\} \\ \delta' &= \delta \end{aligned}$$

Closure under complement

Lemma

Let $L \subseteq \Sigma^*$ be a regular language, then $\overline{L} = \{w \mid w \notin L\}$ is also a regular language.

Proof.

Let $A = (Q, \Sigma, q_0, F, \delta)$ be the automata accepting L.

Let A' be a finite state automaton $(Q', \Sigma', q_0', F', \delta')$ s.t.

$$\begin{aligned} Q' &= Q \\ q_0' &= q_0 \\ F' &= \{q \in Q \mid q \notin F\} \\ \delta' &= \delta \end{aligned}$$

Correctness

$\forall w \in \Sigma^*$, w is accepted by A' iff w is not accepted by A.

□

Concatenation and Kleene star

Let $L_1, L_2, L \subseteq \Sigma^*$

Concatenation and Kleene star

Let $L_1, L_2, L \subseteq \Sigma^*$

$$L_1 \circ L_2 := \{xy \mid x \in L_1, y \in L_2\}$$

Concatenation and Kleene star

Let $L_1, L_2, L \subseteq \Sigma^*$

$$L_1 \circ L_2 := \{xy \mid x \in L_1, y \in L_2\}$$

$$\begin{aligned} L^k &:= \{x_1 x_2 \dots x_k \mid x_i \in L\} \\ L^* &:= \bigcup_{k \geq 0} L^k \end{aligned}$$