CSE 135: Introduction to Theory of Computation
Closure Properties

Sungjin Im

University of California, Merced

02-24-2014

Closure Properties

- Recall that we can carry out operations on one or more languages to obtain a new language

Closure Properties

- Recall that we can carry out operations on one or more languages to obtain a new language
- Very useful in studying the properties of one language by relating it to other (better understood) languages

Closure Properties

- Recall that we can carry out operations on one or more languages to obtain a new language
- Very useful in studying the properties of one language by relating it to other (better understood) languages
- Most useful when the operations are sophisticated, yet are guaranteed to preserve interesting properties of the language.

Closure Properties

- Recall that we can carry out operations on one or more languages to obtain a new language
- Very useful in studying the properties of one language by relating it to other (better understood) languages
- Most useful when the operations are sophisticated, yet are guaranteed to preserve interesting properties of the language.
- Today: A variety of operations which preserve regularity

Closure Properties

- Recall that we can carry out operations on one or more languages to obtain a new language
- Very useful in studying the properties of one language by relating it to other (better understood) languages
- Most useful when the operations are sophisticated, yet are guaranteed to preserve interesting properties of the language.
- Today: A variety of operations which preserve regularity
  - i.e., the universe of regular languages is closed under these operations

Closure Properties

Definition

Regular Languages are closed under an operation op on languages if

$$L_1, L_2, \ldots L_n \text{ regular} \implies L = \mathrm{op}(L_1, L_2, \ldots L_n) \text{ is regular}$$

Closure Properties

Definition

Regular Languages are closed under an operation op on languages if

$$L_1, L_2, \ldots L_n \text{ regular} \implies L = \mathrm{op}(L_1, L_2, \ldots L_n) \text{ is regular}$$

Example

Regular languages are closed under

- "halving", i.e., L regular $\implies \frac{1}{2}L$ regular.

Closure Properties

Definition

Regular Languages are closed under an operation op on languages if

$$L_1, L_2, \ldots L_n \text{ regular} \implies L = \mathrm{op}(L_1, L_2, \ldots L_n) \text{ is regular}$$

Example

Regular languages are closed under

- "halving", i.e., L regular $\implies \frac{1}{2}L$ regular.
- "reversing", i.e., L regular $\implies L^{\mathsf{rev}}$ regular.

Operations from Regular Expressions

Proposition

Regular Languages are closed under $\cup$, $\circ$ *and* *.

Operations from Regular Expressions

Proposition

Regular Languages are closed under $\cup$, $\circ$ *and* *.

Proof.

(Summarizing previous arguments.)

- L_1, L_2 regular $\implies \exists$ regexes R_1, R_2 s.t. $L_1 = L(R_1)$ and $L_2 = L(R_2)$.
  - $\implies L_1 \cup L_2 = L(R_1 \cup R_2) \implies L_1 \cup L_2$ regular.

Operations from Regular Expressions

Proposition

Regular Languages are closed under $\cup$, $\circ$ *and* *.

Proof.

(Summarizing previous arguments.)

- L_1, L_2 regular $\implies \exists$ regexes R_1, R_2 s.t. $L_1 = L(R_1)$ and $L_2 = L(R_2)$.
  - $\implies L_1 \cup L_2 = L(R_1 \cup R_2) \implies L_1 \cup L_2$ regular.
  - $\implies L_1 \circ L_2 = L(R_1 \circ R_2) \implies L_1 \circ L_2$ regular.
  - $\implies L_1^* = L(R_1^*) \implies L_1^*$ regular. $\square$

Closure Under Complementation

Proposition

Regular Languages are closed under complementation, i.e., if L is regular then $\overline{L} = \Sigma^* \setminus L$ is also regular.

Closure Under Complementation

Proposition

Regular Languages are closed under complementation, i.e., if L is regular then $\overline{L} = \Sigma^* \setminus L$ is also regular.

Proof.

- If L is regular, then there is a DFA $M = (Q, \Sigma, \delta, q_0, F)$ such that $L = L(M)$.

Closure Under Complementation

Proposition

Regular Languages are closed under complementation, i.e., if L is regular then $\overline{L} = \Sigma^* \setminus L$ is also regular.

Proof.

- If L is regular, then there is a DFA $M = (Q, \Sigma, \delta, q_0, F)$ such that $L = L(M)$.
- Then, $\overline{M} = (Q, \Sigma, \delta, q_0, Q \setminus F)$ (i.e., switch accept and non-accept states) accepts $\overline{L}$. □

Closure Under Complementation

Proposition

Regular Languages are closed under complementation, i.e., if L is regular then $\overline{L} = \Sigma^* \setminus L$ is also regular.

Proof.

- If L is regular, then there is a DFA $M = (Q, \Sigma, \delta, q_0, F)$ such that $L = L(M)$.
- Then, $\overline{M} = (Q, \Sigma, \delta, q_0, Q \setminus F)$ (i.e., switch accept and non-accept states) accepts $\overline{L}$. $\square$

What happens if M (above) was an NFA?

Closure under $\cap$

Proposition

Regular Languages are closed under intersection, i.e., if L_1 and L_2 are regular then $L_1 \cap L_2$ is also regular.

Closure under $\cap$

Proposition

Regular Languages are closed under intersection, i.e., if L_1 and L_2 are regular then $L_1 \cap L_2$ is also regular.

Proof.

Observe that $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$.

Closure under $\cap$

Proposition

Regular Languages are closed under intersection, i.e., if L_1 and L_2 are regular then $L_1 \cap L_2$ is also regular.

Proof.

Observe that $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$. Since regular languages are closed under union and complementation, we have

- $\overline{L_1}$ and $\overline{L_2}$ are regular

Closure under $\cap$

Proposition

Regular Languages are closed under intersection, i.e., if L_1 and L_2 are regular then $L_1 \cap L_2$ is also regular.

Proof.

Observe that $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$. Since regular languages are closed under union and complementation, we have

- $\overline{L_1}$ and $\overline{L_2}$ are regular
- $\overline{L_1} \cup \overline{L_2}$ is regular

Closure under $\cap$

Proposition

Regular Languages are closed under intersection, i.e., if L_1 and L_2 are regular then $L_1 \cap L_2$ is also regular.

Proof.

Observe that $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$. Since regular languages are closed under union and complementation, we have

- $\overline{L_1}$ and $\overline{L_2}$ are regular
- $\overline{L_1} \cup \overline{L_2}$ is regular
- Hence, $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$ is regular. $\square$

Closure under $\cap$

Proposition

Regular Languages are closed under intersection, i.e., if L_1 and L_2 are regular then $L_1 \cap L_2$ is also regular.

Proof.

Observe that $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$. Since regular languages are closed under union and complementation, we have

- $\overline{L_1}$ and $\overline{L_2}$ are regular
- $\overline{L_1} \cup \overline{L_2}$ is regular
- Hence, $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$ is regular. $\square$

Is there a direct proof for intersection (yielding a smaller DFA)?

Cross-Product Construction

Let $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ and $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ be DFAs recognizing L_1 and L_2, respectively.

Idea: Run M_1 and M_2 in parallel on the same input and accept if both M_1 and M_2 accept.

Cross-Product Construction

Let $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ and $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ be DFAs recognizing L_1 and L_2, respectively.
Idea: Run M_1 and M_2 in parallel on the same input and accept if both M_1 and M_2 accept.

Consider $M = (Q, \Sigma, \delta, q_0, F)$ defined as follows

- $Q = Q_1 \times Q_2$
- $q_0 = \langle q_1, q_2 \rangle$
- $\delta(\langle p_1, p_2 \rangle, a) = \langle \delta_1(p_1, a), \delta_2(p_2, a) \rangle$
- $F = F_1 \times F_2$

Cross-Product Construction

Let $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ and $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ be DFAs recognizing L_1 and L_2, respectively.
Idea: Run M_1 and M_2 in parallel on the same input and accept if both M_1 and M_2 accept.

Consider $M = (Q, \Sigma, \delta, q_0, F)$ defined as follows

- $Q = Q_1 \times Q_2$
- $q_0 = \langle q_1, q_2 \rangle$
- $\delta(\langle p_1, p_2 \rangle, a) = \langle \delta_1(p_1, a), \delta_2(p_2, a) \rangle$
- $F = F_1 \times F_2$

M accepts $L_1 \cap L_2$ (exercise)

Cross-Product Construction

Let $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ and $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ be DFAs recognizing L_1 and L_2, respectively.
Idea: Run M_1 and M_2 in parallel on the same input and accept if both M_1 and M_2 accept.

Consider $M = (Q, \Sigma, \delta, q_0, F)$ defined as follows

- $Q = Q_1 \times Q_2$
- $q_0 = \langle q_1, q_2 \rangle$
- $\delta(\langle p_1, p_2 \rangle, a) = \langle \delta_1(p_1, a), \delta_2(p_2, a) \rangle$
- $F = F_1 \times F_2$

M accepts $L_1 \cap L_2$ (exercise)
What happens if M_1 and M_2 where NFAs?

Cross-Product Construction

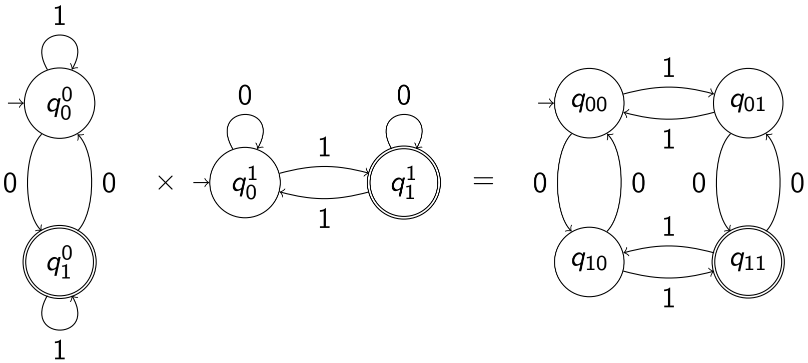
Let $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ and $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ be DFAs recognizing L_1 and L_2, respectively.
Idea: Run M_1 and M_2 in parallel on the same input and accept if both M_1 and M_2 accept.

Consider $M = (Q, \Sigma, \delta, q_0, F)$ defined as follows

- $Q = Q_1 \times Q_2$
- $q_0 = \langle q_1, q_2 \rangle$
- $\delta(\langle p_1, p_2 \rangle, a) = \langle \delta_1(p_1, a), \delta_2(p_2, a) \rangle$
- $F = F_1 \times F_2$

M accepts $L_1 \cap L_2$ (exercise)
What happens if M_1 and M_2 where NFAs? Still works! Set $\delta(\langle p_1, p_2 \rangle, a) = \delta_1(p_1, a) \times \delta_2(p_2, a)$.

An Example



Homomorphism

Definition

A homomorphism is function $h : \Sigma^* \to \Delta^*$ defined as follows:

- $h(\epsilon) = \epsilon$ and for $a \in \Sigma$, $h(a)$ is any string in Δ^*
- For $a = a_1 a_2 \ldots a_n \in \Sigma^*$ $(n \geq 2)$, $h(a) = h(a_1)h(a_2)\ldots h(a_n)$.

Homomorphism

Definition

A homomorphism is function $h : \Sigma^* \to \Delta^*$ defined as follows:

- $h(\epsilon) = \epsilon$ and for $a \in \Sigma$, $h(a)$ is any string in Δ^*
- For $a = a_1 a_2 \ldots a_n \in \Sigma^*$ $(n \geq 2)$, $h(a) = h(a_1)h(a_2)\ldots h(a_n)$.

- A homomorphism h maps a string $a \in \Sigma^*$ to a string in Δ^* by mapping each character of a to a string $h(a) \in \Delta^*$

Homomorphism

Definition

A homomorphism is function $h : \Sigma^* \to \Delta^*$ defined as follows:

- $h(\epsilon) = \epsilon$ and for $a \in \Sigma$, $h(a)$ is any string in Δ^*
- For $a = a_1 a_2 \ldots a_n \in \Sigma^*$ $(n \geq 2)$, $h(a) = h(a_1)h(a_2)\ldots h(a_n)$.

- A homomorphism h maps a string $a \in \Sigma^*$ to a string in Δ^* by mapping each character of a to a string $h(a) \in \Delta^*$
- A homomorphism is a function from strings to strings that "respects" concatenation: for any $x, y \in \Sigma^*$, $h(xy) = h(x)h(y)$.

Homomorphism

Definition

A homomorphism is function $h : \Sigma^* \to \Delta^*$ defined as follows:

- $h(\epsilon) = \epsilon$ and for $a \in \Sigma$, $h(a)$ is any string in Δ^*
- For $a = a_1 a_2 \ldots a_n \in \Sigma^*$ $(n \geq 2)$, $h(a) = h(a_1)h(a_2)\ldots h(a_n)$.

- A homomorphism h maps a string $a \in \Sigma^*$ to a string in Δ^* by mapping each character of a to a string $h(a) \in \Delta^*$
- A homomorphism is a function from strings to strings that "respects" concatenation: for any $x, y \in \Sigma^*$, $h(xy) = h(x)h(y)$. (Any such function is a homomorphism.)

Example

$h : \{0, 1\} \to \{a, b\}^*$ where $h(0) = ab$ and $h(1) = ba$. Then $h(0011) =$

Homomorphism

Definition

A homomorphism is function $h : \Sigma^* \to \Delta^*$ defined as follows:

- $h(\epsilon) = \epsilon$ and for $a \in \Sigma$, $h(a)$ is any string in Δ^*
- For $a = a_1 a_2 \ldots a_n \in \Sigma^*$ $(n \geq 2)$, $h(a) = h(a_1)h(a_2)\ldots h(a_n)$.

- A homomorphism h maps a string $a \in \Sigma^*$ to a string in Δ^* by mapping each character of a to a string $h(a) \in \Delta^*$
- A homomorphism is a function from strings to strings that "respects" concatenation: for any $x, y \in \Sigma^*$, $h(xy) = h(x)h(y)$. (Any such function is a homomorphism.)

Example

$h : \{0,1\} \to \{a,b\}^*$ where $h(0) = ab$ and $h(1) = ba$. Then $h(0011) = ababbaba$

Homomorphism as an Operation on Languages

Definition

Given a homomorphism $h : \Sigma^* \to \Delta^*$ and a language $L \subseteq \Sigma^*$, define $h(L) = \{h(w) \mid w \in L\} \subseteq \Delta^*$.

Homomorphism as an Operation on Languages

Definition

Given a homomorphism $h : \Sigma^* \rightarrow \Delta^*$ and a language $L \subseteq \Sigma^*$, define $h(L) = \{h(w) \mid w \in L\} \subseteq \Delta^*$.

Example

Let $L = \{0^n 1^n \mid n \geq 0\}$ and $h(0) = ab$ and $h(1) = ba$. Then $h(L) = \{(ab)^n (ba)^n \mid n \geq 0\}$

Homomorphism as an Operation on Languages

Definition

Given a homomorphism $h : \Sigma^* \to \Delta^*$ and a language $L \subseteq \Sigma^*$, define $h(L) = \{h(w) \mid w \in L\} \subseteq \Delta^*$.

Example

Let $L = \{0^n 1^n \mid n \geq 0\}$ and $h(0) = ab$ and $h(1) = ba$. Then $h(L) = \{(ab)^n (ba)^n \mid n \geq 0\}$

Exercise: $h(L_1 \cup L_2) = h(L_1) \cup h(L_2)$.

Homomorphism as an Operation on Languages

Definition

Given a homomorphism $h : \Sigma^* \to \Delta^*$ and a language $L \subseteq \Sigma^*$, define $h(L) = \{h(w) \mid w \in L\} \subseteq \Delta^*$.

Example

Let $L = \{0^n 1^n \mid n \geq 0\}$ and $h(0) = ab$ and $h(1) = ba$. Then $h(L) = \{(ab)^n (ba)^n \mid n \geq 0\}$

Exercise: $h(L_1 \cup L_2) = h(L_1) \cup h(L_2)$. $h(L_1 \circ L_2) = h(L_1) \circ h(L_2)$, and $h(L^*) = h(L)^*$.

Closure under Homomorphism

Proposition

Regular languages are closed under homomorphism, i.e., if L is a regular language and h is a homomorphism, then $h(L)$ is also regular.

Closure under Homomorphism

Proposition

Regular languages are closed under homomorphism, i.e., if L is a regular language and h is a homomorphism, then $h(L)$ is also regular.

Proof.

We will use the representation of regular languages in terms of regular expressions to argue this.

Closure under Homomorphism

Proposition

Regular languages are closed under homomorphism, i.e., if L is a regular language and h is a homomorphism, then $h(L)$ is also regular.

Proof.

We will use the representation of regular languages in terms of regular expressions to argue this.

- Define homomorphism as an operation on regular expressions

Closure under Homomorphism

Proposition

Regular languages are closed under homomorphism, i.e., if L is a regular language and h is a homomorphism, then $h(L)$ is also regular.

Proof.

We will use the representation of regular languages in terms of regular expressions to argue this.

- Define homomorphism as an operation on regular expressions
- Show that $L(h(R)) = h(L(R))$

Closure under Homomorphism

Proposition

Regular languages are closed under homomorphism, i.e., if L is a regular language and h is a homomorphism, then $h(L)$ is also regular.

Proof.

We will use the representation of regular languages in terms of regular expressions to argue this.

- Define homomorphism as an operation on regular expressions
- Show that $L(h(R)) = h(L(R))$
- Let R be such that $L = L(R)$. Let $R' = h(R)$. Then $h(L) = L(R')$.

$\cdot\cdot\longrightarrow$

Homomorphism as an Operation on Regular Expressions

Definition

For a regular expression R, let $h(R)$ be the regular expression obtained by replacing each occurence of $a \in \Sigma$ in R by the string $h(a)$.

Homomorphism as an Operation on Regular Expressions

Definition

For a regular expression R, let $h(R)$ be the regular expression obtained by replacing each occurence of $a \in \Sigma$ in R by the string $h(a)$.

Example

If $R = (0 \cup 1)^*001(0 \cup 1)^*$ and $h(0) = ab$ and $h(1) = bc$ then $h(R) = (ab \cup bc)^*ababbc(ab \cup bc)^*$

Homomorphism as an Operation on Regular Expressions

Definition

For a regular expression R, let $h(R)$ be the regular expression obtained by replacing each occurence of $a \in \Sigma$ in R by the string $h(a)$.

Example

If $R = (0 \cup 1)^* 001 (0 \cup 1)^*$ and $h(0) = ab$ and $h(1) = bc$ then $h(R) = (ab \cup bc)^* ababbc (ab \cup bc)^*$

Formally $h(R)$ is defined inductively as follows.

$$\begin{array}{ll} h(\emptyset) = \emptyset & h(R_1 R_2) = h(R_1) h(R_2) \\ h(\epsilon) = \epsilon & h(R_1 \cup R_2) = h(R_2) \cup h(R_2) \\ h(a) = h(a) & h(R^*) = (h(R))^* \end{array}$$

Proof of Claim

Claim

For any regular expression R, $L(h(R)) = h(L(R))$.

Proof.

By induction on the number of operations in R

Proof of Claim

Claim

For any regular expression R, $L(h(R)) = h(L(R))$.

Proof.

By induction on the number of operations in R

- Base Cases: For $R = \epsilon$ or $\emptyset$, $h(R) = R$ and $h(L(R)) = L(R)$.

Proof of Claim

Claim
For any regular expression R, $L(h(R)) = h(L(R))$.

Proof.
By induction on the number of operations in R

- Base Cases: For $R = \epsilon$ or $\emptyset$, $h(R) = R$ and $h(L(R)) = L(R)$. For $R = a$, $L(R) = \{a\}$ and $h(L(R)) = \{h(a)\} = L(h(a)) = L(h(R))$. So claim holds.

Proof of Claim

Claim

For any regular expression R, $L(h(R)) = h(L(R))$.

Proof.

By induction on the number of operations in R

- Base Cases: For $R = \epsilon$ or $\emptyset$, $h(R) = R$ and $h(L(R)) = L(R)$. For $R = a$, $L(R) = \{a\}$ and $h(L(R)) = \{h(a)\} = L(h(a)) = L(h(R))$. So claim holds.
- Induction Step: For $R = R_1 \cup R_2$, observe that

Proof of Claim

Claim

For any regular expression R, $L(h(R)) = h(L(R))$.

Proof.

By induction on the number of operations in R

- **Base Cases:** For $R = \epsilon$ or $\emptyset$, $h(R) = R$ and $h(L(R)) = L(R)$. For $R = a$, $L(R) = \{a\}$ and $h(L(R)) = \{h(a)\} = L(h(a)) = L(h(R))$. So claim holds.
- **Induction Step:** For $R = R_1 \cup R_2$, observe that $h(R) = h(R_1) \cup h(R_2)$

Proof of Claim

Claim

For any regular expression R, $L(h(R)) = h(L(R))$.

Proof.

By induction on the number of operations in R

- Base Cases: For $R = \epsilon$ or $\emptyset$, $h(R) = R$ and $h(L(R)) = L(R)$.
  For $R = a$, $L(R) = \{a\}$ and
  $h(L(R)) = \{h(a)\} = L(h(a)) = L(h(R))$. So claim holds.
- Induction Step: For $R = R_1 \cup R_2$, observe that
  $h(R) = h(R_1) \cup h(R_2)$ and
  $h(L(R)) = h(L(R_1) \cup L(R_2)) = h(L(R_1)) \cup h(L(R_2))$.

Proof of Claim

Claim

For any regular expression R, $L(h(R)) = h(L(R))$.

Proof.

By induction on the number of operations in R

- Base Cases: For $R = \epsilon$ or $\emptyset$, $h(R) = R$ and $h(L(R)) = L(R)$. For $R = a$, $L(R) = \{a\}$ and $h(L(R)) = \{h(a)\} = L(h(a)) = L(h(R))$. So claim holds.
- Induction Step: For $R = R_1 \cup R_2$, observe that $h(R) = h(R_1) \cup h(R_2)$ and $h(L(R)) = h(L(R_1) \cup L(R_2)) = h(L(R_1)) \cup h(L(R_2))$. By induction hypothesis, $h(L(R_i)) = L(h(R_i))$ and so $h(L(R)) = L(h(R_1) \cup h(R_2))$

Proof of Claim

Claim

For any regular expression R, $L(h(R)) = h(L(R))$.

Proof.

By induction on the number of operations in R

- **Base Cases:** For $R = \epsilon$ or $\emptyset$, $h(R) = R$ and $h(L(R)) = L(R)$. For $R = a$, $L(R) = \{a\}$ and $h(L(R)) = \{h(a)\} = L(h(a)) = L(h(R))$. So claim holds.
- **Induction Step:** For $R = R_1 \cup R_2$, observe that $h(R) = h(R_1) \cup h(R_2)$ and $h(L(R)) = h(L(R_1) \cup L(R_2)) = h(L(R_1)) \cup h(L(R_2))$. By induction hypothesis, $h(L(R_i)) = L(h(R_i))$ and so $h(L(R)) = L(h(R_1) \cup h(R_2))$
  Other cases ($R = R_1R_2$ and $R = R_1^*$) similar. $\square$

Nonregularity and Homomorphism

If L is not regular, is $h(L)$ also not regular?

Nonregularity and Homomorphism

If L is not regular, is $h(L)$ also not regular?

- No! Consider $L = \{0^n 1^n \mid n \geq 0\}$ and $h(0) = a$ and $h(1) = \epsilon$. Then $h(L) = a^*$.

Nonregularity and Homomorphism

If L is not regular, is $h(L)$ also not regular?

- No! Consider $L = \{0^n 1^n \mid n \geq 0\}$ and $h(0) = a$ and $h(1) = \epsilon$. Then $h(L) = a^*$.

Applying a homomorphism can "simplify" a non-regular language into a regular language.

Inverse Homomorphism

Inverse Homomorphism

Definition

Given homomorphism $h : \Sigma^* \to \Delta^*$ and $L \subseteq \Delta^*$,
$h^{-1}(L) = \{w \in \Sigma^* \mid h(w) \in L\}$

Inverse Homomorphism

Definition

Given homomorphism $h : \Sigma^* \to \Delta^*$ and $L \subseteq \Delta^*$,
$h^{-1}(L) = \{w \in \Sigma^* \mid h(w) \in L\}$

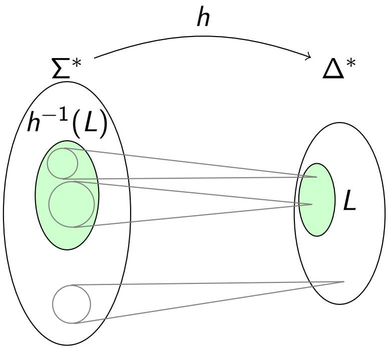
$h^{-1}(L)$ consists of strings whose homomorphic images are in L

Inverse Homomorphism

Definition

Given homomorphism $h : \Sigma^* \to \Delta^*$ and $L \subseteq \Delta^*$,
$h^{-1}(L) = \{w \in \Sigma^* \mid h(w) \in L\}$

$h^{-1}(L)$ consists of strings whose homomorphic images are in L



Inverse Homomorphism

Example

Let $\Sigma = \{a, b\}$, and $\Delta = \{0, 1\}$. Let $L = (00 \cup 1)^*$ and $h(a) = 01$ and $h(b) = 10$.

Inverse Homomorphism

Example

Let $\Sigma = \{a, b\}$, and $\Delta = \{0, 1\}$. Let $L = (00 \cup 1)^*$ and $h(a) = 01$ and $h(b) = 10$.

- $h^{-1}(1001) = \{ba\}$, $h^{-1}(010110) = \{aab\}$

Inverse Homomorphism

Example

Let $\Sigma = \{a, b\}$, and $\Delta = \{0, 1\}$. Let $L = (00 \cup 1)^*$ and $h(a) = 01$ and $h(b) = 10$.

- $h^{-1}(1001) = \{ba\}$, $h^{-1}(010110) = \{aab\}$
- $h^{-1}(L) =$

Inverse Homomorphism

Example

Let $\Sigma = \{a, b\}$, and $\Delta = \{0, 1\}$. Let $L = (00 \cup 1)^*$ and $h(a) = 01$ and $h(b) = 10$.

- $h^{-1}(1001) = \{ba\}$, $h^{-1}(010110) = \{aab\}$
- $h^{-1}(L) = \ (ba)^*$

Inverse Homomorphism

Example

Let $\Sigma = \{a, b\}$, and $\Delta = \{0, 1\}$. Let $L = (00 \cup 1)^*$ and $h(a) = 01$ and $h(b) = 10$.

- $h^{-1}(1001) = \{ba\}$, $h^{-1}(010110) = \{aab\}$
- $h^{-1}(L) = \ (ba)^*$
- What is $h(h^{-1}(L))$?

Inverse Homomorphism

Example

Let $\Sigma = \{a, b\}$, and $\Delta = \{0, 1\}$. Let $L = (00 \cup 1)^*$ and $h(a) = 01$ and $h(b) = 10$.

- $h^{-1}(1001) = \{ba\}$, $h^{-1}(010110) = \{aab\}$
- $h^{-1}(L) = \ (ba)^*$
- What is $h(h^{-1}(L))$? $(1001)^* \subsetneq L$

Note: In general $h(h^{-1}(L)) \subseteq L \subseteq h^{-1}(h(L))$, but neither containment is necessarily an equality.

Closure under Inverse Homomorphism

Proposition

Regular languages are closed under inverse homomorphism, i.e., if L is regular and h is a homomorphism then $h^{-1}(L)$ is regular.

Closure under Inverse Homomorphism

Proposition

Regular languages are closed under inverse homomorphism, i.e., if L is regular and h is a homomorphism then $h^{-1}(L)$ is regular.

Proof.

We will use the representation of regular languages in terms of DFA to argue this.

Closure under Inverse Homomorphism

Proposition

Regular languages are closed under inverse homomorphism, i.e., if L is regular and h is a homomorphism then $h^{-1}(L)$ is regular.

Proof.

We will use the representation of regular languages in terms of DFA to argue this.
Given a DFA M recognizing L, construct an DFA M' that accepts $h^{-1}(L)$

- Intuition: On input w M' will run M on $h(w)$ and accept if M does.

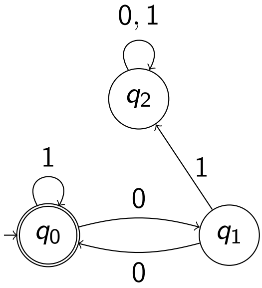
$\cdot\cdot\longrightarrow$

Closure under Inverse Homomorphism

- Intuition: On input w M' will run M on $h(w)$ and accept if M does.

Example

$L = L\left((00 \cup 1)^*\right)$. $h(a) = 01$, $h(b) = 10$.

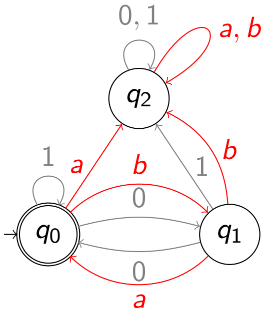


Closure under Inverse Homomorphism

- Intuition: On input w M' will run M on $h(w)$ and accept if M does.

Example

$L = L\left((00 \cup 1)^*\right)$. $h(a) = 01$, $h(b) = 10$.



Closure under Inverse Homomorphism

Formal Construction

- Let $M = (Q, \Delta, \delta, q_0, F)$ accept $L \subseteq \Delta^*$ and let $h : \Sigma^* \to \Delta^*$ be a homomorphism
- Define $M' = (Q', \Sigma, \delta', q_0', F')$, where
  - $Q' = Q$
  - $q_0' = q_0$
  - $F' = F$, and
  - $\delta'(q, a) = \hat{\delta}_M(q, h(a))$; M' on input a simulates M on $h(a)$
- M' accepts $h^{-1}(L)$

Closure under Inverse Homomorphism

Formal Construction

- Let $M = (Q, \Delta, \delta, q_0, F)$ accept $L \subseteq \Delta^*$ and let $h : \Sigma^* \to \Delta^*$ be a homomorphism
- Define $M' = (Q', \Sigma, \delta', q'_0, F')$, where
  - $Q' = Q$
  - $q'_0 = q_0$
  - $F' = F$, and
  - $\delta'(q, a) = \hat{\delta}_M(q, h(a))$; M' on input a simulates M on $h(a)$
- M' accepts $h^{-1}(L)$
- Because $\forall w.\ \hat{\delta}_{M'}(q_0, w) = \hat{\delta}_M(q_0, h(w))$

Proving Non-Regularity

Problem

Show that $L = \{a^n b a^n \mid n \geq 0\}$ *is not regular*

Proof.

Use pumping lemma!

Proving Non-Regularity

Problem

Show that $L = \{a^n b a^n \mid n \geq 0\}$ *is not regular*

Proof.

Use pumping lemma!

Alternate Proof: If we had an automaton M accepting L then we can construct an automaton accepting $K = \{0^n 1^n \mid n \geq 0\}$ ("reduction")

Proving Non-Regularity

Problem

Show that $L = \{a^n b a^n \mid n \geq 0\}$ *is not regular*

Proof.

Use pumping lemma!

Alternate Proof: If we had an automaton M accepting L then we can construct an automaton accepting $K = \{0^n 1^n \mid n \geq 0\}$ ("reduction")

More formally, we will show that by applying a sequence of "regularity preserving" operations to L we can get K.

Proving Non-Regularity

Problem

Show that $L = \{a^n b a^n \mid n \geq 0\}$ *is not regular*

Proof.

Use pumping lemma!

Alternate Proof: If we had an automaton M accepting L then we can construct an automaton accepting $K = \{0^n 1^n \mid n \geq 0\}$ ("reduction")

More formally, we will show that by applying a sequence of "regularity preserving" operations to L we can get K. Then, since K is not regular, L cannot be regular. $\cdot\cdot\!\longrightarrow$

Proving Non-Regularity

Using Closure Properties

Proof (contd).

To show that by applying a sequence of "regularity preserving" operations to $L = \{a^n b a^n \mid n \geq 0\}$ we can get $K = \{0^n 1^n \mid n \geq 0\}$.

Proof (contd).

To show that by applying a sequence of "regularity preserving" operations to $L = \{a^n b a^n \mid n \geq 0\}$ we can get $K = \{0^n 1^n \mid n \geq 0\}$.

- Consider homomorphism $h_1 : \{a, b, c\}^* \to \{a, b, c\}^*$ defined as $h_1(a) = a$, $h_1(b) = b$, $h_1(c) = a$.
  - $L_1 = h_1^{-1}(L) = \{(a \cup c)^n b (a \cup c)^n \mid n \geq 0\}$

Proof (contd).

To show that by applying a sequence of "regularity preserving" operations to $L = \{a^n b a^n \mid n \geq 0\}$ we can get $K = \{0^n 1^n \mid n \geq 0\}$.

- Consider homomorphism $h_1 : \{a, b, c\}^* \rightarrow \{a, b, c\}^*$ defined as $h_1(a) = a$, $h_1(b) = b$, $h_1(c) = a$.
  - $L_1 = h_1^{-1}(L) = \{(a \cup c)^n b (a \cup c)^n \mid n \geq 0\}$
- Let $L_2 = L_1 \cap L(a^* b c^*) = \{a^n b c^n \mid n \geq 0\}$

Proof (contd).

To show that by applying a sequence of "regularity preserving" operations to $L = \{a^n b a^n \mid n \geq 0\}$ we can get $K = \{0^n 1^n \mid n \geq 0\}$.

- Consider homomorphism $h_1 : \{a, b, c\}^* \to \{a, b, c\}^*$ defined as $h_1(a) = a$, $h_1(b) = b$, $h_1(c) = a$.
  - $L_1 = h_1^{-1}(L) = \{(a \cup c)^n b (a \cup c)^n \mid n \geq 0\}$
- Let $L_2 = L_1 \cap L(a^* b c^*) = \{a^n b c^n \mid n \geq 0\}$
- Homomorphism $h_2 : \{a, b, c\}^* \to \{0, 1\}^*$ is defined as $h_2(a) = 0$, $h_2(b) = \epsilon$, and $h_2(c) = 1$.'
  - $L_3 = h_2(L_2) = \{0^n 1^n \mid n \geq 0\} = K$

Proof (contd).

To show that by applying a sequence of "regularity preserving" operations to $L = \{a^n b a^n \mid n \geq 0\}$ we can get $K = \{0^n 1^n \mid n \geq 0\}$.

- Consider homomorphism $h_1 : \{a, b, c\}^* \to \{a, b, c\}^*$ defined as $h_1(a) = a$, $h_1(b) = b$, $h_1(c) = a$.
  - $L_1 = h_1^{-1}(L) = \{(a \cup c)^n b (a \cup c)^n \mid n \geq 0\}$
- Let $L_2 = L_1 \cap L(a^* b c^*) = \{a^n b c^n \mid n \geq 0\}$
- Homomorphism $h_2 : \{a, b, c\}^* \to \{0, 1\}^*$ is defined as $h_2(a) = 0$, $h_2(b) = \epsilon$, and $h_2(c) = 1$.'
  - $L_3 = h_2(L_2) = \{0^n 1^n \mid n \geq 0\} = K$
- Now if L is regular then so are L_1, L_2, L_3, and K. But K is not regular, and so L is not regular. $\square$

Proving Regularity

For a language L, define $head(L)$ to be the set of all prefixes of strings in L. Prove that if L is regular, so is $head(L)$.

Proving Regularity

For a language L, define $head(L)$ to be the set of all prefixes of strings in L. Prove that if L is regular, so is $head(L)$. We can prove this by

Proving Regularity

For a language L, define $head(L)$ to be the set of all prefixes of strings in L. Prove that if L is regular, so is $head(L)$. We can prove this by

- constructing a DFA/NFA that accepts $head(L)$; or

Proving Regularity

For a language L, define $head(L)$ to be the set of all prefixes of strings in L. Prove that if L is regular, so is $head(L)$. We can prove this by

- constructing a DFA/NFA that accepts $head(L)$; or
- giving a regular expression for $head(L)$; or

Proving Regularity

For a language L, define $head(L)$ to be the set of all prefixes of strings in L. Prove that if L is regular, so is $head(L)$. We can prove this by

- constructing a DFA/NFA that accepts $head(L)$; or
- giving a regular expression for $head(L)$; or
- by applying a sequence of regularity-preserving operations.

Proving Regularity via Regularity-preserving Operations

Proving Regularity via Regularity-preserving Operations

- For simplicity, assume $\sum = \{0, 1\}$; the proof easliy extends to a general alphabet set.

Proving Regularity via Regularity-preserving Operations

- For simplicity, assume $\sum = \{0, 1\}$; the proof easliy extends to a general alphabet set.
- Define a homomorphism h where $h(0) = 0, h(1) = 1, h(a) = 0, h(b) = 1$. Then $h^{-1}(L)$ is regular.

Proving Regularity via Regularity-preserving Operations

- For simplicity, assume $\sum = \{0, 1\}$; the proof easliy extends to a general alphabet set.
- Define a homomorphism h where $h(0) = 0, h(1) = 1, h(a) = 0, h(b) = 1$. Then $h^{-1}(L)$ is regular.
- $(0 \cup 1)^*(a \cup b)^*)$ is regular, so is $(0 \cup 1)^*(a \cup b)^* \cap h^{-1}(L)$.

Proving Regularity via Regularity-preserving Operations

- For simplicity, assume $\sum = \{0, 1\}$; the proof easliy extends to a general alphabet set.
- Define a homomorphism h where $h(0) = 0, h(1) = 1, h(a) = 0, h(b) = 1$. Then $h^{-1}(L)$ is regular.
- $(0 \cup 1)^*(a \cup b)^*)$ is regular, so is $(0 \cup 1)^*(a \cup b)^* \cap h^{-1}(L)$.
- Define a homomorphism g where $g(0) = 0, g(1) = 1, g(a) = \epsilon, g(b) = \epsilon$. Then $g\Big((0 \cup 1)^*(a \cup b)^* \cap h^{-1}(L)\Big)$ is regular.

Proving Regularity via Regularity-preserving Operations

- For simplicity, assume $\sum = \{0, 1\}$; the proof easliy extends to a general alphabet set.
- Define a homomorphism h where $h(0) = 0, h(1) = 1, h(a) = 0, h(b) = 1$. Then $h^{-1}(L)$ is regular.
- $(0 \cup 1)^*(a \cup b)^*)$ is regular, so is $(0 \cup 1)^*(a \cup b)^* \cap h^{-1}(L)$.
- Define a homomorphism g where $g(0) = 0, g(1) = 1, g(a) = \epsilon, g(b) = \epsilon$. Then $g\Big((0 \cup 1)^*(a \cup b)^* \cap h^{-1}(L)\Big)$ is regular.
- Do you see $head(L) = g\Big((0 \cup 1)^*(a \cup b)^* \cap h^{-1}(L)\Big)$?

Proving Regularity and Non-Regularity

Showing that L is not regular

Proving Regularity and Non-Regularity

Showing that L is not regular

- Use the pumping lemma

Proving Regularity and Non-Regularity

Showing that L is not regular

- Use the pumping lemma
- Or, show that from L you can obtain a known non-regular language through regularity preserving operations.

Proving Regularity and Non-Regularity

Showing that L is not regular

- Use the pumping lemma
- Or, show that from L you can obtain a known non-regular language through regularity preserving operations.
- Note: **Non-regular languages are not closed under the operations discussed.**

Proving Regularity and Non-Regularity

Showing that L is not regular

- Use the pumping lemma
- Or, show that from L you can obtain a known non-regular language through regularity preserving operations.
- Note: **Non-regular languages are not closed under the operations discussed.**

Showing that L is regular

Proving Regularity and Non-Regularity

Showing that L is not regular

- Use the pumping lemma
- Or, show that from L you can obtain a known non-regular language through regularity preserving operations.
- Note: **Non-regular languages are not closed under the operations discussed.**

Showing that L is regular

- Construct a DFA or NFA or regular expression recognizing L

Proving Regularity and Non-Regularity

Showing that L is not regular

- Use the pumping lemma
- Or, show that from L you can obtain a known non-regular language through regularity preserving operations.
- Note: **Non-regular languages are not closed under the operations discussed.**

Showing that L is regular

- Construct a DFA or NFA or regular expression recognizing L
- Or, show that L can be obtained from known regular languages $L_1, L_2, \ldots L_k$ through regularity preserving operations

Proving Regularity and Non-Regularity

Showing that L is not regular

- Use the pumping lemma
- Or, show that from L you can obtain a known non-regular language through regularity preserving operations.
- Note: **Non-regular languages are not closed under the operations discussed.**

Showing that L is regular

- Construct a DFA or NFA or regular expression recognizing L
- Or, show that L can be obtained from known regular languages $L_1, L_2, \ldots L_k$ through regularity preserving operations
- Note: **Do not use pumping lemma to prove regularity!!**

A list of Regularity-Preserving Operations

Regular languages are closed under the following operations.

- Regular Expression operations
- Boolean operations: union, intersection, complement
- Homomorphism
- Inverse Homomorphism

(And several other operations...)