CIS 770: Formal Language Theory

Pavithra Prabhakar

Kansas State University

Spring 2015

Homomorphism

Definition

A homomorphism is function $h : \Sigma^* \rightarrow \Delta^*$ 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

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

Homomorphism as an Operation on Languages

Definition

Given a homomorphism $h : \Sigma^* \rightarrow \Delta^*$ 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 $h(L) = \{(ab)^n (ba)^n \mid n \geq 0\}$

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.

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 $h(L) = L(R')$.

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

- 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 ($R = R_1 R_2$ and $R = R_1^*$) similar. $\square$

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 $h(L) = a^*$.

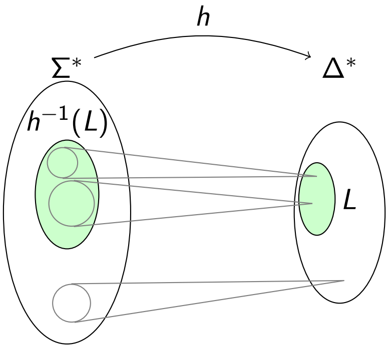
Applying a homomorphism can "simplify" a non-regular language into a regular language.

Inverse Homomorphism

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 $h(b) = 10$.

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

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 $h^{-1}(L)$

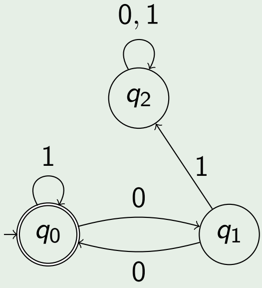
- Intuition: On input w M' will run M on $h(w)$ and accept if M does.

Closure under Inverse Homomorphism

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

Example

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



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 $h^{-1}(L)$
- 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!
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.

Proof (contd).

To show that by applying a sequence of "regularity preserving" operations to $L = \{a^n ba^n \mid n \geq 0\}$ 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 $L_2 = L_1 \cap L(a^* bc^*) = \{a^n bc^n \mid n \geq 0\}$
- 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

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA. Consider

$L = \{w | M$ accepts w and M visits every state at least once on input $w\}$

Is L regular?
Note that M does not necessarily accept all strings in L; $L \subseteq L(M)$.
By applying a series of regularity preserving operations to $L(M)$ we will construct L, thus showing that L is regular

Computations: Valid and Invalid

- Consider an alphabet Δ consisting of $[paq]$ where $p, q \in Q$, $a \in \Sigma$ and $\delta(p, a) = q$. So symbols of Δ represent transitions of M.
- Let $h : \Delta \to \Sigma^*$ be a homomorphism such that $h([paq]) = a$
- $L_1 = h^{-1}(L(M))$; L_1 contains strings of $L(M)$ where each symbol is associated with a pair of states that represent some transition
  - Some strings of L_1 represent valid computations of M. But there are also other strings in L_1 which do not correspond to valid computations of M
- We will first remove all the strings from L_1 that correspond to invalid computations, and then remove those that do not visit every state at least once.

Only Valid Computations

Strings of Δ^* that represent valid computations of M satisfy the following conditions

- The first state in the first symbol must be q_0

$$L_2 = L_1 \cap (([q_0 a_1 q_1] \cup [q_0 a_2 q_2] \cup \cdots \cup [q_0 a_k q_k])\Delta^*)$$

  ($[q_0 a_1 q_1], \ldots [q_0 a_k q_k]$ are all the transitions out of q_0 in M)

- The first state in one symbol must equal the second state in previous symbol

$$L_3 = L_2 \setminus (\Delta^*(\bigcup_{q \neq r} [paq][rbs])\Delta^*)$$

  Remove "invalid" sequences from L_2. Difference of two regular languages is regular (why?). So L_3 is regular.

- The second state of the last symbol must be in F. Holds trivially because L_3 only contains strings accepted by M

Example continued

So far, regular language L_3 = set of strings in Δ^* that represent valid computations of M.

- Let $E_q \subseteq \Delta$ be the set of symbols where q appears neither as the first nor the second state. Then E_q^* is the set of strings where q never occurs.
- We remove from L_3 those strings where some $q \in Q$ never occurs
$$L_4 = L_3 \setminus (\bigcup_{q \in Q} E_q^*)$$
- Finally we discard the state components in L_4
$$L = h(L_4)$$
- Hence, 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
- 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...)