1 Closure Properties

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

Definition 1. Regular Languages are closed under an operation op on languages if

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

1.1 Boolean Operators

Operations from Regular Expressions

Proposition 2. *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. $L_1 = \mathbf{L}(R_1)$ and $L_2 = \mathbf{L}(R_2)$.
  - $\implies L_1 \cup L_2 = \mathbf{L}(R_1 \cup R_2) \implies L_1 \cup L_2$ regular.
  - $\implies L_1 \circ L_2 = \mathbf{L}(R_1 \circ R_2) \implies L_1 \circ L_2$ regular.
  - $\implies L_1^* = \mathbf{L}(R_1^*) \implies L_1^*$ regular. □

Closure Under Complementation

Proposition 3. *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 $\overline{L}$. □

What happens if M (above) was an *NFA*?

Closure under $\cap$

Proposition 4. *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. □

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.

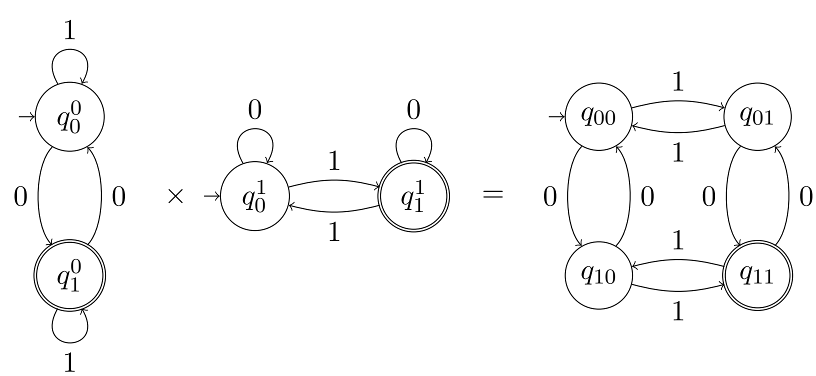
Consider $M = (Q, \Sigma, \delta, q_0, F)$ defined as follows

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

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



1.2 Homomorphisms

Homomorphism

Definition 5. 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 6. $h : \{0,1\} \to \{a,b\}^*$ where $h(0) = ab$ and $h(1) = ba$. Then $h(0011) = ababbaba$

Homomorphism as an Operation on Languages

Definition 7. 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 8. Let $L = \{0^n 1^n \mid n \geq 0\}$ and $h(0) = ab$ and $h(1) = ba$. Then $h(L) = \{(ab)^n (ba)^n \mid n \geq 0\}$

Proposition 9. *For any languages L_1 and L_2, the following hold: $h(L_1 \cup L_2) = h(L_1) \cup h(L_2)$; $h(L_1 \circ L_2) = h(L_1) \circ h(L_2)$; and $h(L_1^*) = h(L_1)^*$.*

Proof. Left as exercise. □

Closure under Homomorphism

Proposition 10. *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 $\mathbf{L}(h(R)) = h(\mathbf{L}(R))$
- Let R be such that $L = \mathbf{L}(R)$. Let $R' = h(R)$. Then $h(L) = \mathbf{L}(R')$. □

Homomorphism as an Operation on Regular Expressions

Definition 11. 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 12. 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, $\mathbf{L}(h(R)) = h(\mathbf{L}(R))$.

Proof. By induction on the number of operations in R

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

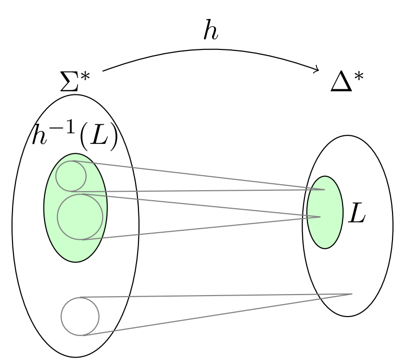
  Other cases ($R = R_1 R_2$ and $R = R_1^*$) similar. □

1.3 Inverse Homomorphism

Inverse Homomorphism

Definition 13. 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 14. 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 15. *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 16. $L = L((00 \cup 1)^*)$. $h(a) = 01$, $h(b) = 10$.

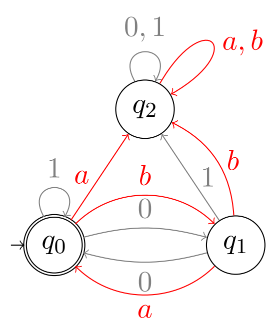


Figure 1: Transitions of automaton M accepting language L is shown in gray. The transitions of automaton accepting $h^{-1}(L)$ are shown in red.

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
  - $\delta'(q, a) = q'$ *where* $\hat{\delta}_M(q, h(a)) = \{q'\}$; M' on input a simulates M on $h(a)$
- M' accepts $h^{-1}(L)$ because $\forall w.\ \hat{\delta}_{M'}(q_0, w) = \hat{\delta}_M(q_0, h(w))$ (which you show by induction on w).

---

2 Applications of Closure Properties

Example I

Definition 17. For a language $L \subseteq \Sigma^*$, define $\mathrm{suffix}(L) = \{v \in \Sigma^* \mid \exists u \in \Sigma^*.\ uv \in L\}$.

Proposition 18. *Regular languages are closed under the* suffix$(\cdot)$ *operator. That is, if L is regular then* suffix(L) *is also regular.*

Proof. We present two possible proofs of this result.
Direct Construction: Since L is regular, there is a DFA $M = (Q, \Sigma, \delta, q_0, F)$ that recognizes L. We will construct an NFA N such that $\mathbf{L}(N) = \mathrm{suffix}(\mathbf{L}(M)) = \mathrm{suffix}(L)$. Let us first spell out what N needs to do in order to recognize suffix(L) — on input v, it needs to check if there is some u such that $uv \in L$ or uv is accepted by M. N will do this by simulating M on the input v, but instead of starting from the initial state q_0, it will first guess a state that M reaches on some string u (such that $uv \in L$), and then simulate M on the input v. Formally, $N = (Q', \Sigma, \delta', q_0', F')$ where

- $Q' = Q \cup \{q_0'\}$, where $q_0' \notin Q$
- $F' = F$
- And δ' is given by

$$\delta'(q, a) = \begin{cases} \delta(q, a) & \text{if } q \in Q \\ \{q \in Q \mid \exists u.\ q_0 \xrightarrow{u}_M q\} & \text{if } q = q_0' \text{ and } a = \epsilon \\ \emptyset & \end{cases}$$

To complete the proof we need to argue that v is accepted by N iff $v \in \text{suffix}(\mathbf{L}(M))$. Suppose v is accepted by N. Since the only transitions out of the initial state q_0' are ϵ-transitions, the accepting computation of N on v looks like

$$q_0' \xrightarrow{\epsilon}_N q \xrightarrow{v}_N q'$$

with $q' \in F' = F$, and q being such that there is a u such that $q_0 \xrightarrow{u}_M q$. In other words, we have

$$q_0 \xrightarrow{u}_M q \xrightarrow{v}_M q'$$

and so $uv \in \mathbf{L}(M) = L$. Thus, $v \in \text{suffix}(L)$. Conversely, suppose $v \in \text{suffix}(L)$. Then there is u such that $uv \in L$. Since M recognizes L, M accepts uv using a computation of the form

$$q_0 \xrightarrow{u}_M q \xrightarrow{v}_M q'$$

where q is some state in Q and $q' \in F$. Then from the definition of N, we have a computation

$$q_0' \xrightarrow{\epsilon}_N q \xrightarrow{v}_N q'$$

and since $F' = F$, $v \in \mathbf{L}(N)$. This completes the correctness proof of N.
Closure Properties: Another proof of the same result uses closure properties.

- For an alphabet Σ, let $\bar{\Sigma} = \{\bar{a} \mid a \in \Sigma\}$.
- Define the homomorphisms unbar : $(\Sigma \cup \bar{\Sigma})^* \to \Sigma^*$ and rembar : $(\Sigma \cup \bar{\Sigma})^* \to \Sigma^*$ as

$$\begin{array}{ll} \text{unbar}(\bar{a}) = a \text{ for } \bar{a} \in \bar{\Sigma} & \text{unbar}(a) = a \text{ for } a \in \Sigma \\ \text{rembar}(\bar{a}) = \epsilon \text{ for } \bar{a} \in \bar{\Sigma} & \text{rembar}(a) = a \text{ for } a \in \Sigma \end{array}$$

- Let $L_1 = \text{unbar}^{-1}(L)$; since L is regular and regular languages are closed under inverse homomorphisms, L_1 is regular. L_1 contains strings belonging to L which have some (or none) of the letters annotated with a bar.
- Let $L_2 = L_1 \cap \bar{\Sigma}^*\Sigma^*$; L_2 is regular because regular languages are closed under intersection. L_2 is the set of strings from L where some of the first few letters have been annotated with a bar.
- Observe that $\text{suffix}(L) = \text{rembar}(L_2)$. Thus $\text{suffix}(L)$ is regular.

□

Example II

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

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

Is L regular?

Note that M does not necessarily accept all strings in L; $L \subseteq \mathbf{L}(M)$.

By applying a series of regularity preserving operations to $\mathbf{L}(M)$ 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 $h([paq]) = a$
- $L_1 = h^{-1}(\mathbf{L}(M))$; L_1 contains strings of $\mathbf{L}(M)$ 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

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

  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.

2.1 In a nutshell ...

Proving Regularity using Closure Properties
How can one show that L is a regular language?

- 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

---

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...)
