Pearson New International Edition

Introduction to Automata Theory Languages, and Computation

Hopcroft Motwani Ullman

Third Edition

ALWAYS LEARNING™

PEARSON®

Pearson Education Limited
Edinburgh Gate
Harlow
Essex CM20 2JE
England and Associated Companies throughout the world

Visit us on the World Wide Web at: www.pearsoned.co.uk

PEARSON®

ISBN 10: 1-292-03905-1
ISBN 13: 978-1-292-03905-3

British Library Cataloguing-in-Publication Data
A catalogue record for this book is available from the British Library

Printed in the United States of America

Let h be the homomorphism defined by $h(a) = 01$ and $h(b) = 10$. We claim that $h^{-1}(L)$ is the language of regular expression $(\mathbf{ba})^*$, that is, all strings of repeating ba pairs. We shall prove that $h(w)$ is in L if and only if w is of the form $baba \cdots ba$.

(If) Suppose w is n repetitions of ba for some $n \geq 0$. Note that $h(ba) = 1001$, so $h(w)$ is n repetitions of 1001. Since 1001 is composed of two 1's and a pair of 0's, we know that 1001 is in L. Therefore any repetition of 1001 is also formed from 1 and 00 segments and is in L. Thus, $h(w)$ is in L.

(Only-if) Now, we must assume that $h(w)$ is in L and show that w is of the form $baba \cdots ba$. There are four conditions under which a string is *not* of that form, and we shall show that if any of them hold then $h(w)$ is not in L. That is, we prove the contrapositive of the statement we set out to prove.

1. If w begins with a, then $h(w)$ begins with 01. It therefore has an isolated 0, and is not in L.

2. If w ends in b, then $h(w)$ ends in 10, and again there is an isolated 0 in $h(w)$.

3. If w has two consecutive a's, then $h(w)$ has a substring 0101. Here too, there is an isolated 0 in w.

4. Likewise, if w has two consecutive b's, then $h(w)$ has substring 1010 and has an isolated 0.

Thus, whenever one of the above cases hold, $h(w)$ is not in L. However, unless at least one of items (1) through (4) hold, then w is of the form $baba \cdots ba$. To see why, assume none of (1) through (4) hold. Then (1) tells us w must begin with b, and (2) tells us w ends with a. Statements (3) and (4) tell us that a's and b's must alternate in w. Thus, the logical "OR" of (1) through (4) is equivalent to the statement "w is not of the form $baba \cdots ba$." We have proved that the "OR" of (1) through (4) implies $h(w)$ is not in L. That statement is the contrapositive of the statement we wanted: "if $h(w)$ is in L, then w is of the form $baba \cdots ba$." $\square$

We shall next prove that the inverse homomorphism of a regular language is also regular, and then show how the theorem can be used.

Theorem 4.16: If h is a homomorphism from alphabet Σ to alphabet T, and L is a regular language over T, then $h^{-1}(L)$ is also a regular language.

PROOF: The proof starts with a DFA A for L. We construct from A and h a DFA for $h^{-1}(L)$ using the plan suggested by Fig. 4.6. This DFA uses the states of A but translates the input symbol according to h before deciding on the next state.

Formally, let L be $L(A)$, where DFA $A = (Q, T, \delta, q_0, F)$. Define a DFA

$$B = (Q, \Sigma, \gamma, q_0, F)$$

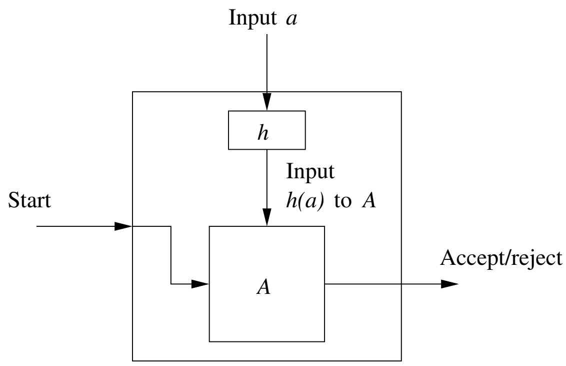

Figure 4.6: The DFA for $h^{-1}(L)$ applies h to its input, and then simulates the DFA for L

where transition function γ is constructed by the rule $\gamma(q,a) = \hat{\delta}\bigl(q, h(a)\bigr)$. That is, the transition B makes on input a is the result of the sequence of transitions that A makes on the string of symbols $h(a)$. Remember that $h(a)$ could be ϵ, it could be one symbol, or it could be many symbols, but $\hat{\delta}$ is properly defined to take care of all these cases.

It is an easy induction on $|w|$ to show that $\hat{\gamma}(q_0, w) = \hat{\delta}\bigl(q_0, h(w)\bigr)$. Since the accepting states of A and B are the same, B accepts w if and only if A accepts $h(w)$. Put another way, B accepts exactly those strings w that are in $h^{-1}(L)$. $\square$

Example 4.17: In this example we shall use inverse homomorphism and several other closure properties of regular sets to prove an odd fact about finite automata. Suppose we required that a DFA visit every state at least once when accepting its input. More precisely, suppose $A = (Q, \Sigma, \delta, q_0, F)$ is a DFA, and we are interested in the language L of all strings w in Σ^* such that $\hat{\delta}(q_0, w)$ is in F, and also for every state q in Q there is some prefix x_q of w such that $\hat{\delta}(q_0, x_q) = q$. Is L regular? We can show it is, but the construction is complex.

First, start with the language M that is $L(A)$, i.e., the set of strings that A accepts in the usual way, without regard to what states it visits during the processing of its input. Note that $L \subseteq M$, since the definition of L puts an additional condition on the strings of $L(A)$. Our proof that L is regular begins by using an inverse homomorphism to, in effect, place the states of A into the input symbols. More precisely, let us define a new alphabet T consisting of symbols that we may think of as triples $[paq]$, where:

1. p and q are states in Q,
2. a is a symbol in Σ, and
3. $\delta(p, a) = q$.

That is, we may think of the symbols in T as representing transitions of the automaton A. It is important to see that the notation $[paq]$ is our way of expressing a single symbol, not the concatenation of three symbols. We could have given it a single letter as a name, but then its relationship to p, q, and a would be hard to describe.

Now, define the homomorphism $h([paq]) = a$ for all p, a, and q. That is, h removes the state components from each of the symbols of T and leaves only the symbol from Σ. Our first step in showing L is regular is to construct the language $L_1 = h^{-1}(M)$. Since M is regular, so is L_1 by Theorem 4.16. The strings of L_1 are just the strings of M with a pair of states, representing a transition, attached to each symbol.

As a very simple illustration, consider the two-state automaton of Fig. 4.4(a). The alphabet Σ is $\{0, 1\}$, and the alphabet T consists of the four symbols $[p0q]$, $[q0q]$, $[p1p]$, and $[q1q]$. For instance, there is a transition from state p to q on input 0, so $[p0q]$ is one of the symbols of T. Since 101 is a string accepted by the automaton, h^{-1} applied to this string will give us $2^3 = 8$ strings, of which $[p1p][p0q][q1q]$ and $[q1q][q0q][p1p]$ are two examples.

We shall now construct L from L_1 by using a series of further operations that preserve regular languages. Our first goal is to eliminate all those strings of L_1 that deal incorrectly with states. That is, we can think of a symbol like $[paq]$ as saying the automaton was in state p, read input a, and thus entered state q. The sequence of symbols must satisfy three conditions if it is to be deemed an accepting computation of A:

1. The first state in the first symbol must be q_0, the start state of A.

2. Each transition must pick up where the previous one left off. That is, the first state in one symbol must equal the second state of the previous symbol.

3. The second state of the last symbol must be in F. This condition in fact will be guaranteed once we enforce (1) and (2), since we know that every string in L_1 came from a string accepted by A.

The plan of the construction of L is shown in Fig. 4.7.

We enforce (1) by intersecting L_1 with the set of strings that begin with a symbol of the form $[q_0aq]$ for some symbol a and state q. That is, let E_1 be the expression $[q_0a_1q_1] + [q_0a_2q_2] + \cdots$, where the pairs a_iq_i range over all pairs in $\Sigma \times Q$ such that $\delta(q_0, a_i) = q_i$. Then let $L_2 = L_1 \cap L(E_1T^*)$. Since E_1T^* is a regular expression denoting all strings in T^* that begin with the start state (treat T in the regular expression as the sum of its symbols), L_2 is all strings that are formed by applying h^{-1} to language M and that have the start state as the first component of its first symbol; i.e., it meets condition (1).

To enforce condition (2), it is easier to subtract from L_2 (using the set-difference operation) all those strings that violate it. Let E_2 be the regular expression consisting of the sum (union) of the concatenation of all pairs of

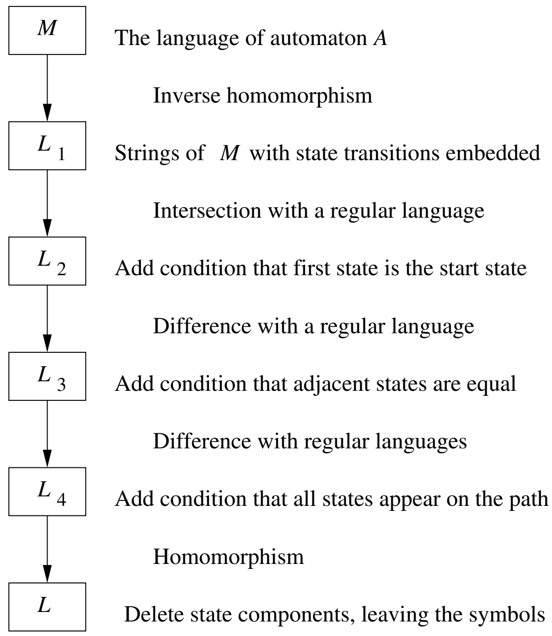

Figure 4.7: Constructing language L from language M by applying operations that preserve regularity of languages

symbols that fail to match; that is, pairs of the form $[paq][rbs]$ where $q \neq r$. Then $T^*E_2T^*$ is a regular expression denoting all strings that fail to meet condition (2).

We may now define $L_3 = L_2 - L(T^*E_2T^*)$. The strings of L_3 satisfy condition (1) because strings in L_2 must begin with the start symbol. They satisfy condition (2) because the subtraction of $L(T^*E_2T^*)$ removes any string that violates that condition. Finally, they satisfy condition (3), that the last state is accepting, because we started with only strings in M, all of which lead to acceptance by A. The effect is that L_3 consists of the strings in M with the states of the accepting computation of that string embedded as part of each symbol. Note that L_3 is regular because it is the result of starting with the regular language M, and applying operations — inverse homomorphism, intersection, and set difference — that yield regular sets when applied to regular sets.

Recall that our goal was to accept only those strings in M that visited every state in their accepting computation. We may enforce this condition by additional applications of the set-difference operator. That is, for each state q, let E_q be the regular expression that is the sum of all the symbols in T such that q appears in neither its first or last position. If we subtract $L(E_q^*)$ from L_3 we have those strings that are an accepting computation of A and that visit state q at least once. If we subtract from L_3 all the languages $L(E_q^*)$ for q in Q, then we have the accepting computations of A that visit all the states. Call this language L_4. By Theorem 4.10 we know L_4 is also regular.

Our final step is to construct L from L_4 by getting rid of the state components. That is, $L = h(L_4)$. Now, L is the set of strings in Σ^* that are accepted by A and that visit each state of A at least once during their acceptance. Since regular languages are closed under homomorphisms, we conclude that L is regular. $\square$

4.2.5 Exercises for Section 4.2

Exercise 4.2.1: Suppose h is the homomorphism from the alphabet $\{0,1,2\}$ to the alphabet $\{a,b\}$ defined by: $h(0) = a$; $h(1) = ab$, and $h(2) = ba$.

* a) What is $h(0120)$?

b) What is $h(21120)$?

* c) If L is the language $L(\mathbf{01^*2})$, what is $h(L)$?

d) If L is the language $L(\mathbf{0+12})$, what is $h(L)$?

* e) Suppose L is the language $\{ababa\}$, that is, the language consisting of only the one string $ababa$. What is $h^{-1}(L)$?

! f) If L is the language $L\big(\mathbf{a}(\mathbf{ba})^*\big)$, what is $h^{-1}(L)$?

*! **Exercise 4.2.2:** If L is a language, and a is a symbol, then L/a, the *quotient* of L and a, is the set of strings w such that wa is in L. For example, if $L = \{a, aab, baa\}$, then $L/a = \{\epsilon, ba\}$. Prove that if L is regular, so is L/a. *Hint*: Start with a DFA for L and consider the set of accepting states.

! **Exercise 4.2.3:** If L is a language, and a is a symbol, then $a\backslash L$ is the set of strings w such that aw is in L. For example, if $L = \{a, aab, baa\}$, then $a\backslash L = \{\epsilon, ab\}$. Prove that if L is regular, so is $a\backslash L$. *Hint*: Remember that the regular languages are closed under reversal and under the quotient operation of Exercise 4.2.2.

! **Exercise 4.2.4:** Which of the following identities are true?

a) $(L/a)a = L$ (the left side represents the concatenation of the languages L/a and $\{a\}$).

b) $a(a\backslash L) = L$ (again, concatenation with $\{a\}$, this time on the left, is intended).

c) $(La)/a = L$.

d) $a\backslash(aL) = L$.