2.4 LANGUAGES THAT ARE AND ARE NOT REGULAR

The results of the last two sections establish that the regular languages are closed under a variety of operations and that regular languages may be specified either by regular expressions or by deterministic or nondeterministic finite automata. These facts, used singly or in combinations, provide a variety of techniques for showing languages to be regular.

Example 2.4.1: Let $\Sigma = \{0, 1, \ldots, 9\}$ and let $L \subseteq \Sigma^*$ be the set of decimal representations for nonnegative integers (without redundant leading 0's) divisible by 2 or 3. For example, $0, 3, 6, 244 \in L$, but $1, 03, 00 \notin L$. Then L is regular. We break the proof into four parts.

Let L_1 be the set of decimal representations of nonnegative integers. Then it is easy to see that

$$L_1 = 0 \cup \{1, 2, \ldots, 9\}\Sigma^*,$$

which is regular since it is denoted by a regular expression.

Let L_2 be the set of decimal representations of nonnegative integers divisible by 2. Then L_2 is just the set of members of L, ending in 0, 2, 4, 6, or 8; that is,

$$L_2 = L_1 \cap \Sigma^*\{0, 2, 4, 6, 8\},$$

which is regular by Theorem 2.3.1(e).

Let L_3 be the set of decimal representations of nonnegative integers divisible by 3. Recall that a number is divisible by 3 if and only if the sum of its digits is divisible by 3. We construct a finite automaton that keeps track in its finite control of the sum modulo 3 of a string of digits. L_3 will then be the intersection

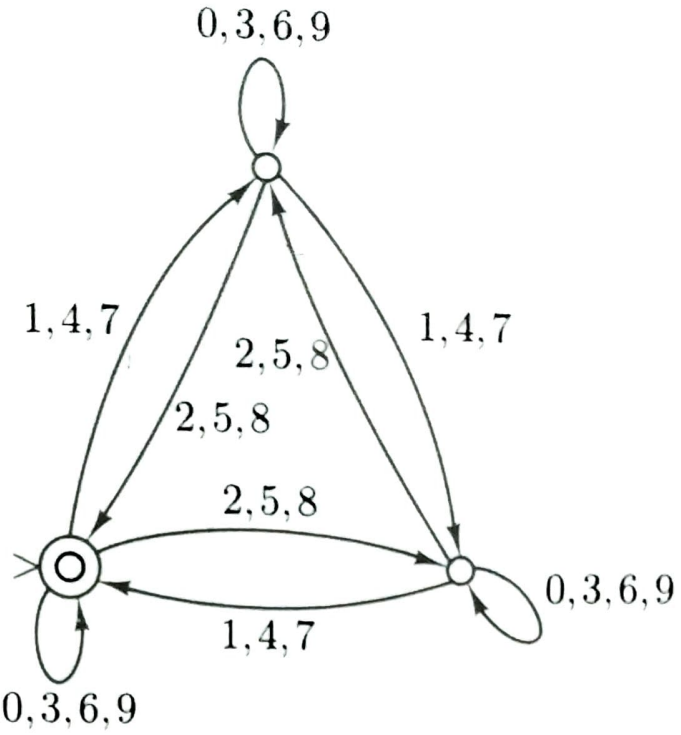

Figure 2-18

with L_1 of the language accepted by this finite automaton. The automaton is pictured in Figure 2-18.

Finally, $L = L_2 \cup L_3$, surely a regular language.$\Diamond$

Although we now have a variety of powerful techniques for showing that languages are regular, as yet we have none for showing that languages are *not* regular. We know on fundamental principles that nonregular languages do exist, since the number of regular expressions (or the number of finite automata) is countable, whereas the number of languages is uncountable. But to demonstrate that any particular language is not regular requires special tools.

Two properties shared by all regular languages, but not by certain nonregular languages, may be phrased intuitively as follows:

(1) As a string is scanned left to right, the amount of memory that is required in order to determine at the end whether or not the string is in the language must be bounded, fixed in advance and dependent on the language, not the particular input string. For example, we would expect that $\{a^n b^n : n \geq 0\}$ is not regular, since it is difficult to imagine how a finite-state device could be constructed that would correctly remember, upon reaching the border between the a's and the b's, how many a's it had seen, so that the number could be compared against the number of b's.

(2) Regular languages with an infinite number of strings are represented by automata with cycles and regular expressions involving the Kleene star. Such languages must have infinite subsets with a certain simple repetitive structure that arises from the Kleene star in a corresponding regular expression or a cycle in the state diagram of a finite automaton. This would lead us to

expect, for example, that $\{a^n : n \geq 1 \text{ is a prime}\}$ is not regular, since there is no simple periodicity in the set of prime numbers.

These intuitive ideas are accurate but not sufficiently precise to be used in formal proofs. We shall now prove a theorem that captures some of this intuition and yields the nonregularity of certain languages as an easy consequence.

Theorem 2.4.1: *Let L be a regular language. There is an integer $n \geq 1$ such that any string $w \in L$ with $|w| \geq n$ can be rewritten as $w = xyz$ such that $y \neq e$, $|xy| \leq n$, and $xy^iz \in L$ for each $i \geq 0$.*

Proof: Since L is regular, L is accepted by a deterministic finite automaton M. Suppose that n is the number of states of M, and let w be a string of length n or greater. Consider now the first n steps of the computation of M on w:

$$(q_0, w_1 w_2 \ldots w_n) \vdash_M (q_1, w_2 \ldots w_n) \vdash_M \ldots \vdash_M (q_n, e),$$

where q_0 is the initial state of M, and $w_1 \ldots w_n$ are the n first symbols of w. Since M has only n states, and there are $n + 1$ configurations $(q_i, w_{i+1} \ldots, w_n)$ appearing in the computation above, *by the pigeonhole principle* there exist i and j, $0 \leq i < j \leq n$, such that $q_i = q_j$. That is, the string $y = w_i w_{i+1} \ldots w_j$ drives M from state q_i back to state q_i, and this string is nonempty since $i < j$. But then this string could be removed from w, or repeated any number of times in w just after the jth symbol of w, and M would still accept this string. That is, M accepts $xy^iz \in L$ for each $i \geq 0$, where $x = w_1 \ldots w_i$, and $z = w_{j+1} \ldots w_m$. Notice finally that the length of xy, the number we called j above, is by definition at most n, as required. ■

This theorem is one of a general class called *pumping theorems*, because they assert the existence of certain points in certain strings where a substring can be repeatedly inserted without affecting the acceptability of the string. In terms of its structure as a mathematical statement, the pumping theorem above is by far the most sophisticated theorem that we have seen in this book, because its assertion, however simple to prove and apply, involves *five alternating quantifiers*. Consider again what it says:

for each regular language L;
there exists an $n \geq 1$, such that
for each string w in L longer than n,
there exist strings x, y, z with $w = xyz$, $y \neq e$, and $|xy| \leq n$, such that
for each $i \geq 0$ $xy^iz \in L$.

Applying the theorem correctly can be subtle. It is often useful to think of the application of this result as a *game* between yourself, the prover, who is striving to establish that the given language L is not regular, and an adversary who is insisting that L is regular. The theorem states that, once L has been fixed, the

adversary must start by providing a number n; then you come up with a string w in the language that is longer than n; the adversary must now supply an appropriate decomposition of w into xyz; and, finally, you triumphantly point out i for which xy^iz is not in the language. If you have a strategy that always wins, no matter how brilliantly the adversary plays, then you have established that L is not regular.

It follows from this theorem that each of the two languages mentioned earlier in this section is not regular.

Example 2.4.2: The language $L = \{a^ib^i : i \geq 0\}$ is not regular, for if it were regular, Theorem 2.4.1 would apply for some integer n. Consider then the string $w = a^nb^n \in L$. By the theorem, it can be rewritten as $w = xyz$ such that $|xy| \leq n$ and $y \neq e$ —that is, $y = a^i$ for some $i > 0$. But then $xz = a^{n-i}b^n \notin L$, contradicting the theorem.$\Diamond$

Example 2.4.3: The language $L = \{a^n : n \text{ is prime}\}$ is not regular. For suppose it were, and let x, y, and z be as specified in Theorem 2.4.1. Then $x = a^p$, $y = a^q$, and and $z = a^r$, where $p, r \geq 0$ and $q > 0$. By the theorem, $xy^nz \in L$ for each $n \geq 0$; that is, $p + nq + r$ is prime for each $n \geq 0$. But this is impossible; for let $n = p + 2q + r + 2$; then $p + nq + r = (q + 1)\ \ (p + 2q + r)$, which is a product of two natural numbers, each greater than 1.$\Diamond$

Example 2.4.4: Sometimes it pays to use closure properties to show that a language is not regular. Take for example

$$L = \{w \in \{a, b\}^* : w \text{ has an equal number of } a\text{'s and } b\text{'s}\}.$$

L is not regular, because if L were indeed regular, then so would be $L \cap a^*b^*$ —by closure under intersection; recall Theorem 2.3.1(e). However, this latter language is precisely $\{a^nb^n : n \geq 0\}$, which we just showed is not regular.$\Diamond$

In fact, Theorem 2.4.1 can be strengthened substantially in several ways (see Problem 2.4.11 for one of them).