The Theory of Computation

Bernard M.E. Moret

Addison-Wesley-Longman, 1998

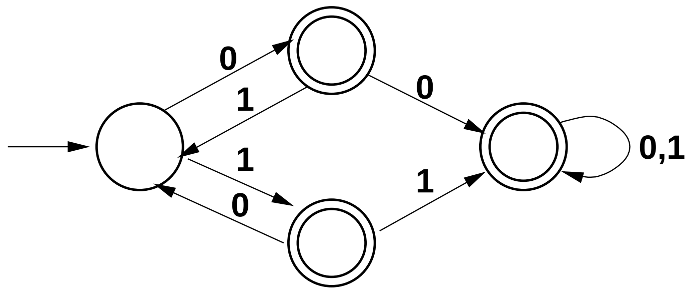
Solution Manual
Chapter 3, Problem 3.17

1.

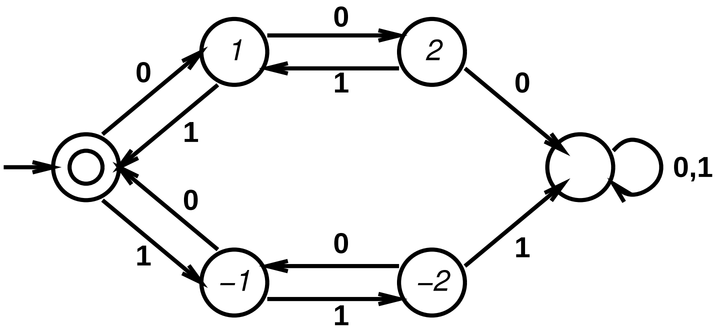
2. Not regular. Let n be the constant of the pumping lemma and pick the string $z = 0^n 2 0^n$, which is clearly in the language. Write $z = uvw$; then v is a string of 0s (at least one and at most n) and lies entirely within the left half of the string. Pumping v just once yields the new string $uvvw = 0^{n+|v|} 2 0^n$, which is not in the language.

3. Not regular. The crucial point is that w cannot be the empty string (otherwise the language would be Σ^+, since we could always choose $w = \varepsilon$ and let y account for the entire string). Let n be the constant of the pumping lemma and pick the string $z = 10^{2n-1}110^{2n-1}10$, which is clearly in the language and which, moreover, has only one acceptable decomposition in the language, namely $z = (10^{2n-1}1)(10^{2n-1}1)^R 0$. Use our extended pumping lemma to write $z = z_1 z_2 z_3$ with $z_1 = 1$, $z_2 = 0^n$, and $z_3 = 0^{n-1}110^{2n-1}10$. Now write $z_2 = uvw$; thus v is a string of 0s; pick an odd value for i, say $i = 3$; then the string $z_1 uv^3 wz_3$ should be in the language. But that string is $10^{2n+2|v|-1}110^{2n-1}10$, which is not in the language, because its first part, $10^{2n+2|v|-1}1$, is not reversible (because it has an odd number of 0s) and because its second part, $10^{2n-1}1$, no longer matches the first (because the first part now has more 0s).

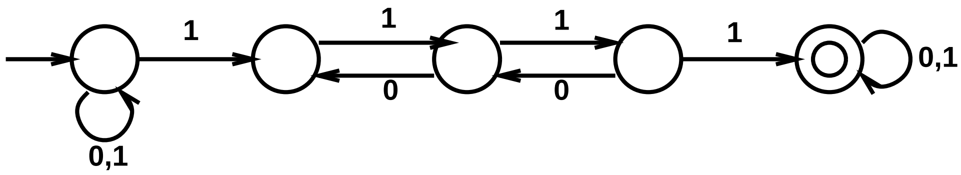
4. Regular. This language is the complement of a regular language and regular languages are closed under complementation. More directly, the following four-state DFA recognizes it:



5. Regular. All we need to do is to keep track of the number of 0s and 1s as differences that can take the values 0, 1, -1, 2, -2, and larger. The following FA recognizes it:

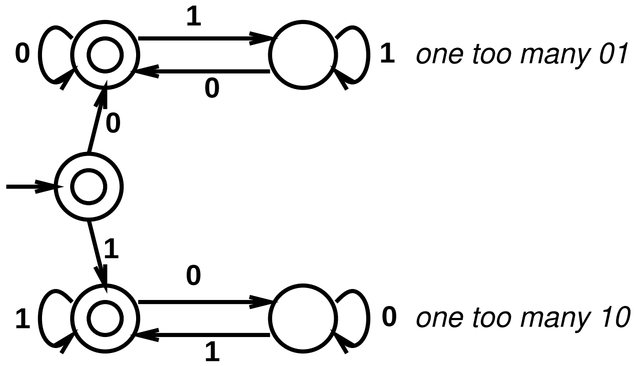


6. Not regular. Let n be the constant of the pumping lemma and choose the string $z = 0^n 1^{2n} 0^n$. z is in the language, because the third piece is as long as the first. Now write $z = uvw$, with $|uv| \leq n$; it follows that uv is a string of 0s and thus also v. But then the pumped string $z' = uvvw = 0^{n+|v|} 1^{2n} 0^n$ is such that the first two pieces are each longer than the last and thus $z' \notin L$.
7. This language is something of a trick: it is simply 0^*, since, for every $n \in \Re$, there exists some $y \in \Re$ such that $y = n^2$. Thus the language is regular.
8. Regular. Being able to use just prefixes means that we need not worry about initial deficits in the number of 1s: we can choose a substring where the number of 1s always dominates the number of 0s and need only keep track of dominance by 1, 2, 3 or 4, which an FA can do. Thus the language is in fact regular: an NFA need only guess where the substring of interest starts (always at a 11 block), then move right on another 1, left on a 0, until it has moved three states to the right; if it ever tries to move back to the start state, the guess was bad. The resulting NFA is shown below.



A DFA with exactly the same states, but no loop on input of 1 on the start state and a transition back to the start state from the second state on input of 0 does the same job.

9. Regular. The key insight here is that we need only worry about transitions from 0 to 1 and from 1 to 0 and that these transitions must match. (Any string of 0s or any string of 1s by itself is entirely neutral, since neither one introduces a 01 or a 10.) We need not even count the transitions, since a transition from 1 to 0 can only be followed by one from 0 to 1 and vice versa. Thus we have a very simple machine or regular expression, which simply swallows consecutive 0s and consecutive 1s and only checks for transitions; a possible regular expression is $0^* + 1^* + (0^+1^+)^*0^+ + (1^+0^+)^*1^+$. Another way to view the problem is to realize that matching the number of transitions will cause the string to end with the same character with which it started, so we could also write $\varepsilon + 0(0+1)^*0 + 1(0+1)^*1$. The following DFA recognizes the language:



10. Not regular. Let n be the constant of the pumping lemma. Consider the string $z = 0^{n!+1}1^{n!}$. $n!$ and $n!+1$ are relatively prime (the only prime divisors of $n!$ are the primes between 2 and n and all leave a remainder of 1 when dividing $n!+1$), so z is in the language. By construction, the substring uv of the lemma is composed entirely of 0s, so the string $z(i) = uv^iw$ will have $n!+1+(i-1)|v|$ 0s. If $|v| < n$, we pump up once ($i = 2$) to obtain $z' = uvvw = 0^{n!+1+|v|}1^{n!}$; but $n!+1+|v| = n!+(|v|+1)$ is divisible by $|v|+1$, which is at least

2 and at most n and so also divides $n!$, so that the gcd is at least 2 and z' is not in the language. If $|v| = n$, we pump down once ($i = 0$) to obtain $z'' = uw = 0^{n!+1-|v|}1^{n!}$; but $n! + 1 - |v| = n! - (|v| - 1)$ is divisible by $|v| - 1 = n - 1$, so that the gcd is at least $n - 1$ and z'' is not in the language.