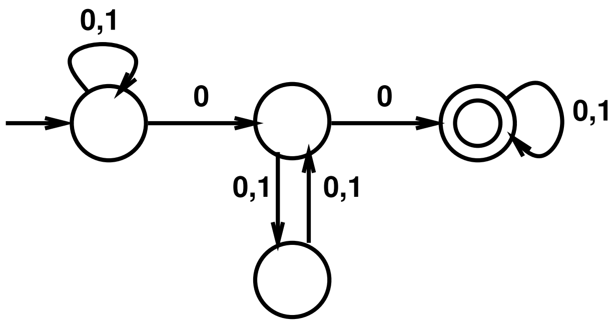
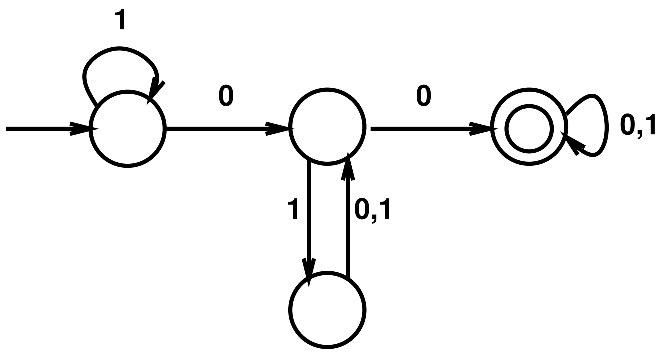
The Theory of Computation
Bernard M.E. Moret
Addison-Wesley-Longman, 1998

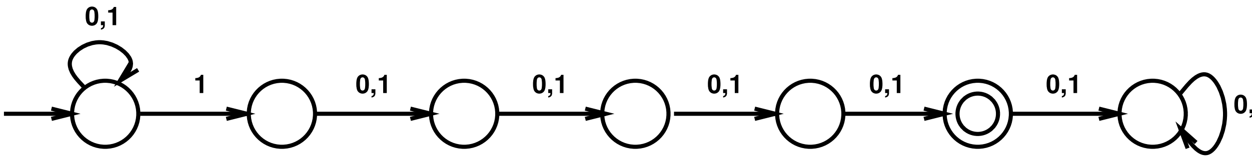
Solution Manual
Chapter 3, Problem 3.9

1. The nondeterministic automaton is quite simple: it chooses the two 0s, checking that they are separated by an even number of symbols. The



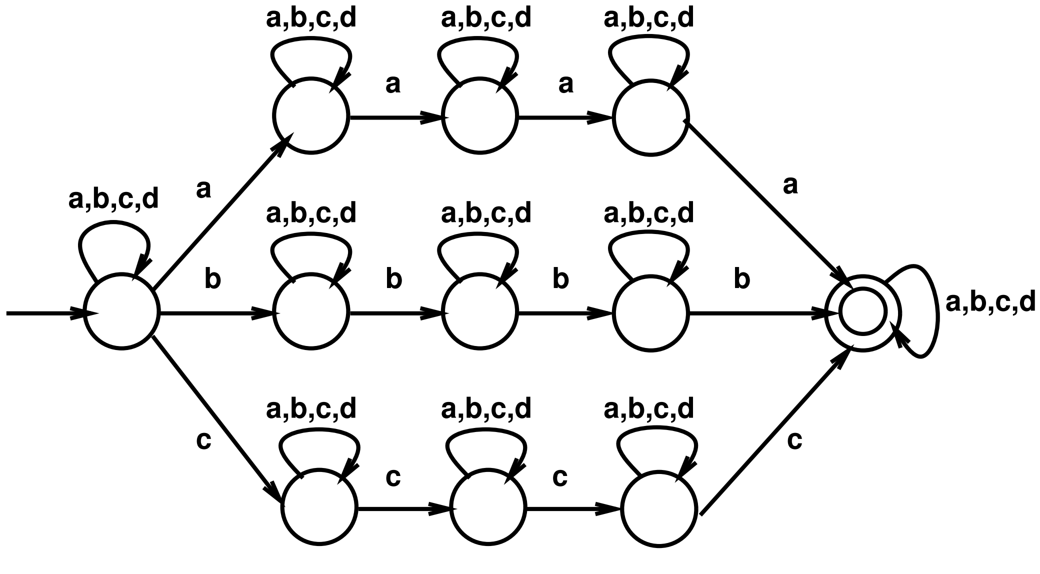
deterministic automaton is essentially the same as the nondeterministic one, but requires a bit of insight. Once we have seen the first 0, we simply wait for another 0 to turn up an even number of positions later. This works because, should a 0 turn up an odd number of positions later, we could "use" either the first 0 or the "new" one as the starting point—they have the same relative parity in the string. The result is depicted below.
2. The nondeterministic automaton simply eats up symbols until it guesses that the fifth symbol from the end has arrived, at which point it checks that this symbol is a 1 and that it is followed by exactly four more symbols; at that point it can accept the string, but any additional symbol causes rejection. The deterministic automaton is quite painful to set up. It has to remember the positions of the last few 1s it has seen—in a window of size five. If it reaches the end of the string in a state corresponding to having seen a 1 five positions back, it accepts, otherwise it rejects. Until then, it moves between the various states representing the possible windows of size five; since there are 32 such





windows, we need a lot of states—since that is not quite sufficient, as we also have a series of states for initialization (to build the first five characters in the window), all rejecting. Overall, then, we can easily build the DFA, but it will have 63 states—which explains why we do not illustrate it here . . . (Actually, it is easy to see that we need only 32 states, because we need only keep track of a window of 4 symbols—the next symbol's handling is automatic; this gives us 16 window states, 15 initializing states, and a final state—needed here, since no window state can be final.)

3. The nondeterministic automaton simply guesses which of the 3 symbols will be the one to appear at least 4 times in all and proceeds to verify that. A deterministic automaton is easy enough to build, but will need to keep track of the number of *a*s, of *b*s, and of *c*s seen so far, until one of them has been seen 4 times. So we will have states of the type "seen x *a*s, y *b*s, and z *c*s," for $0 \leq x, y, z \leq 3$, or at least $4^3 = 64$ states. The machine must have one more state for accepting and thus will have 65 states in all. Whenever it sees a d, it simply loops in place; whenever it sees one of the three symbols of interest, it moves to the state denoting the updated count, until it reaches the accepting state, which is a trap.



a,b,c,d
a,b,c,d
a,b,c,d
a,b,c,d
a,b,c,d
a,b,c,d
a,b,c,d
a,b,c,d
a,b,c,d
a,b,c,d
a
a
a
a
b
b
b
b
c
c
c
c