1. The original automaton is shown in Figure 1. The determinized version of it is shown in Figure 2.

- Number of states of determinized automaton is \((O)(2^n)\), where \(n\) is the number of places before the end where '1' must occur.

- No. We always need to remember last \(n\) letters. Assume we were able to build a deterministic automaton accepting the same language as the original (nondeterministic) one with \(k < 2^n\) states. Then there are two inputs \(x = x_1 \ldots x_n\) and \(y = y_1 \ldots y_n\), \(x \neq y\) that lead to the same state. There is index \(i\) such that \(i = \max (j : x_j \neq y_j)\). Without loss of generality assume \(x_i = 1, y_i = 0\). Now add both to \(x\) and \(y\) ones - \(x' = x_1 \ldots 1, y' = y_1 \ldots 1\). We know that after reading first \(n\) letters of \(x'\) and \(y'\) the automaton will be in the same state. Since after that it only reads same input, after reading whole \(x'\) and \(y'\) it will be in the same state. However, \(x'\) should be accepted (\(n\)th letter from the end is 1) while \(y'\) should be rejected. Contradiction.

2. Assume \(w = a_1a_2 \ldots a_m\). We are using the notion of traces as defined in lectures: \(t_N = q_0^N a_1 a_2^N \ldots a_m q_m^N\), \(t_D = q_0^D a_1 q_1^D a_1 \ldots a_m q_m^D\). (Note that for the same input there are multiple \(t_N\), traces of a nondeterministic automaton possible). We claim \(\forall p \in q_i^D : \exists t_N : p = q_i^N\) and we prove it by induction on the length of trace. In order to prove the base of induction we consider the definition of initial state of the deterministic automaton, \(q_0^D = \{q_0^N\}\) and note that the claim holds. Now assume the claim for the trace of length smaller than \(i + 1\). Let \(p \in q_{i+1}^D\). According to the definition of \(q_{i+1}^D\) we have \(p \in \{q_i^N : \exists q_i^N \in q_i^D, q_i^N \in \delta_N(q_{i-1}^N, a_{i+1})\}\). From the definition we see that \(p\) was a state to which we transferred upon reading \(a_{i+1}\) in (some) state \(q_{i-1}^N\). But according to our induction assumption that was also a \(q_i^N\) in some nondeterministic trace. Therefore, \(p\) is \(q_{i+1}^N\) in the extension of that trace. Having this claim, assume \(w \in L(D)\). This gives \(q_m^D \in F_D \Rightarrow q_m^D \cap F_N \neq \emptyset \Rightarrow \exists r \in q_m^D \cap F_N\). From what we’ve just proven, there is a trace \(t_N\) such that \(q_m^N = r \Rightarrow q_m^N \in F_N \Rightarrow w \in L(N)\)
Figure 2: Deterministic automaton
3. The problem with directly applying the definition of a product automaton is that thread 1 would be able to unlock the lock made by thread 0. Therefore, the alphabet needs to change a bit so that lock$^i$ is followed by a corresponding unlock, unlock$^i$. The final product of lock spec and control flow automaton is shown in 3.