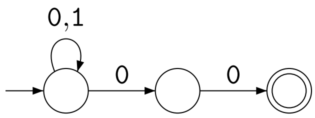
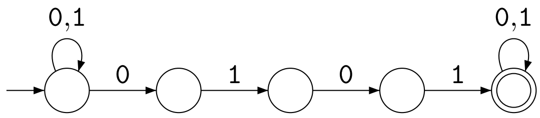
The University of Melbourne
School of Computing and Information Systems
COMP30026 Models of Computation

Selected Tutorial Solutions, Week 9

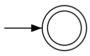
64. (a) $\{w \mid w$ ends with 00$\}$ using three states



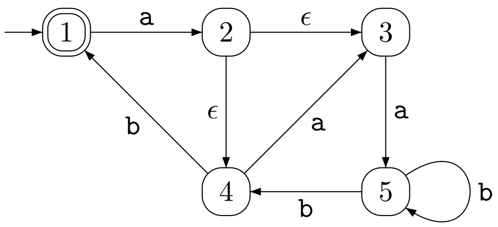
(b) $\{w \mid w$ contains the substring 0101$\}$ using five states



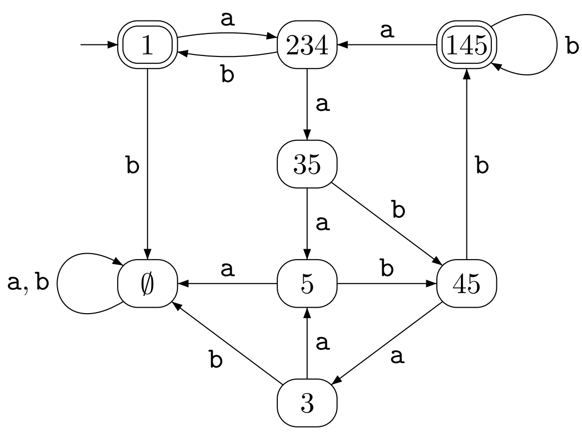
(c) The language $\{\epsilon\}$ using one state



65. From this NFA:



we end up with this DFA:



66. (a) $\{w \mid w$ begins with a `1` and ends with a `0`$\}$: $1(0 \cup 1)^*0$
(b) $\{w \mid w$ contains the substring `0101`$\}$: $(0 \cup 1)^*0101(0 \cup 1)^*$
(c) $\{w \mid w$ has length at least 3 and its third symbol is `0`$\}$: $(0 \cup 1)(0 \cup 1)0(0 \cup 1)^*$
(d) $\{w \mid$ the length of w is at most 5$\}$: $(\epsilon \cup 0 \cup 1)(\epsilon \cup 0 \cup 1)(\epsilon \cup 0 \cup 1)(\epsilon \cup 0 \cup 1)(\epsilon \cup 0 \cup 1)$
(e) $\{w \mid w$ is any string except `11` and `111`$\}$: $\epsilon \cup 1 \cup 11111^* \cup (0 \cup 1)^*0(0 \cup 1)^*$
(f) $\{w \mid$ every odd position of w is a `1`$\}$: $(1(0 \cup 1))^*(\epsilon \cup 1)$
(g) $\{w \mid w$ contains at least two `0`s and at most one `1`$\}$: $0^*(00 \cup 001 \cup 010 \cup 100)0^*$
(h) $\{\epsilon, \texttt{0}\}$: $\epsilon \cup 0$
(i) The empty set: $\emptyset$
(j) All strings except the empty string: $(0 \cup 1)(0 \cup 1)^*$

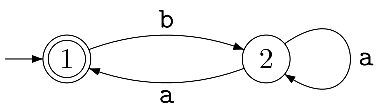
67. If A is regular then *suffix*(A) is regular. Namely, let $D = (Q, \Sigma, \delta, q_0, F)$ be a DFA for A. Assume every state in Q is *reachable* from q_0. Then we can turn D into an NFA N for *suffix*(A) by adding a new state q_{-1} which becomes the NFA's start state. For each state $q \in Q$, we add an epsilon transition from q_{-1} to q.

That is, we define N to be $(Q \cup \{q_{-1}\}, \Sigma, \delta', q_{-1}, F)$, with transition function

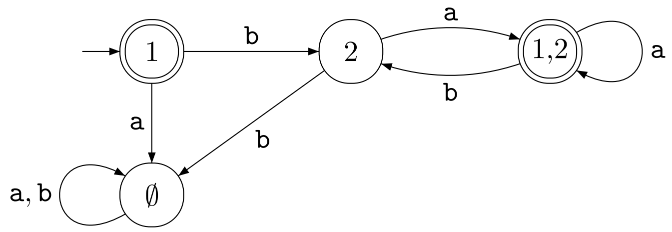
$$\delta'(q, x) = \begin{cases} \{\delta(q, x)\} & \text{for } q \in Q \text{ and } x \in \Sigma \\ Q & \text{for } q = q_{-1} \text{ and } x = \epsilon \\ \emptyset & \text{for } q \neq q_{-1} \text{ and } x = \epsilon \end{cases}$$

The restriction we assumed, that all of D's states are reachable, is not a severe one. It is easy to identify unreachable states and eliminate them (which of course does not change the language of the DFA). To see why we need to eliminate unreachable states before generating N in the suggested way, consider what happens to this DFA for $\{\epsilon\}$: $(\{q_0, q_1, q_2\}, \{\texttt{a}\}, \delta, q_0, \{q_0\})$, where $\delta(q_0, \texttt{a}) = \delta(q_1, \texttt{a}) = q_1$ and $\delta(q_2, \texttt{a}) = q_0$.

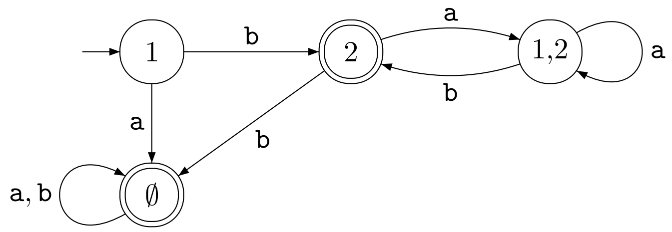
68. (a) Here is an NFA for $(\texttt{ba}^*\texttt{a})^*$:



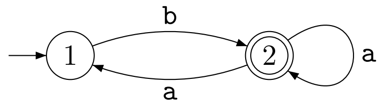
(b) Here is an equivalent DFA, obtained using the subset construction:



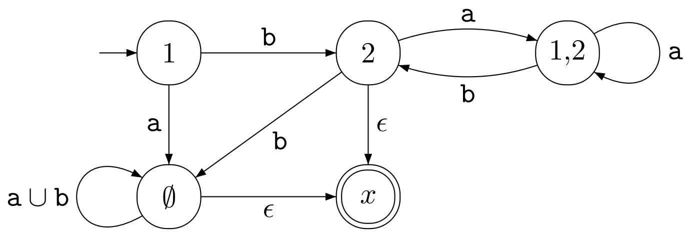
(c) It is easy to get a DFA for the complement:



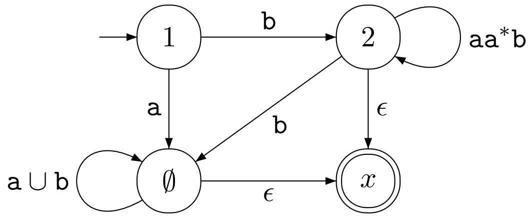
(d) It would be problematic to do the “complement trick” on the NFA, as it is only guaranteed to work on DFAs. We would get the following, which accepts, for example, `baa`:



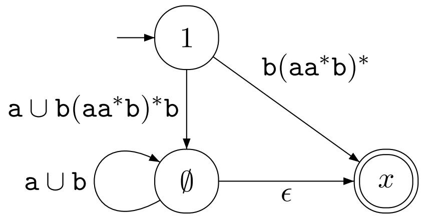
(e) Starting from the DFA that we found in (c), we make sure that we have just one accept state:



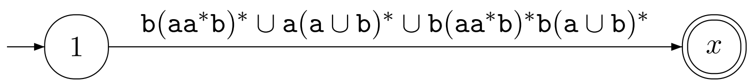
Let us first remove the state labeled 1,2:



Now we can remove state 2:

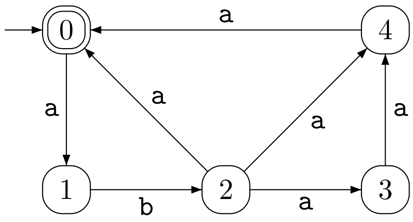


Finally, eliminating the state labelled $\emptyset$, we are left with:

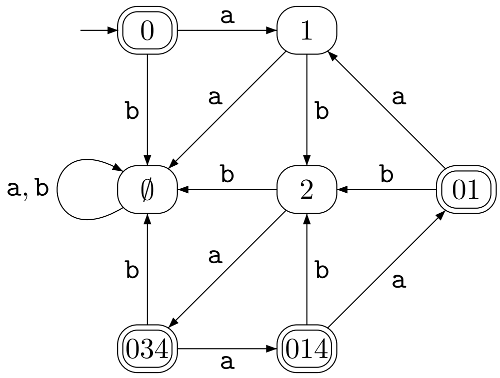


The resulting regular expression can be read from that diagram.

69. From this NFA:

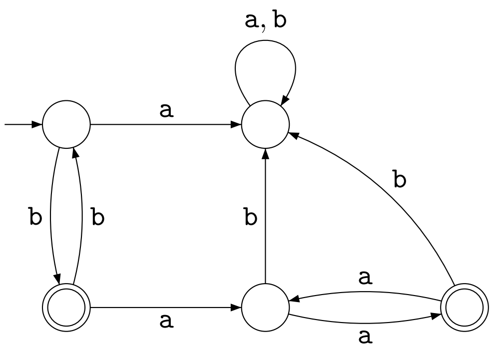


we end up with the following DFA:



70. Note that a string in A must contain at least one `b`. Moreover, all `a`s (if there are any) must come after all `b`s, because otherwise we would find a substring `ab` somewhere. So a regular expression for A is $\texttt{b}(\texttt{bb})^*(\texttt{aa})^*$.

71. Here is a five-state DFA which recognises $\texttt{b}(\texttt{bb})^*(\texttt{aa})^*$:



72. This is the minimal DFA: