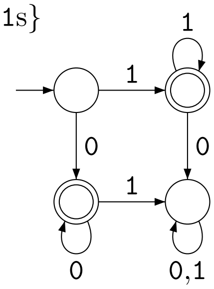
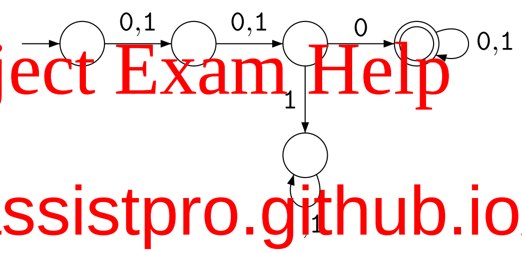
The University of Melbourne
School of Computing and Information Systems
COMP30026 Models of Computation

Sample Answers to Problem Set Exercises, Week 8

P8.1 (a) $\{w \mid w$ is not empty and contains only `0`s or only `1`s$\}$

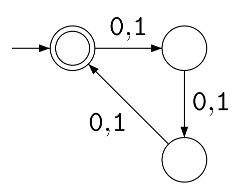


(b) $\{w \mid w$ has length at least 3 and its third symbol is `0`$\}$

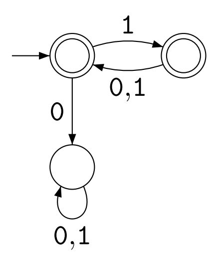


(c) $\{w \mid$ the length of w is at most 5$\}$

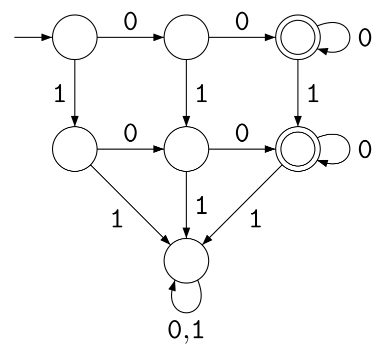
(d) $\{w \mid$ the length of w is a multiple of 3$\}$



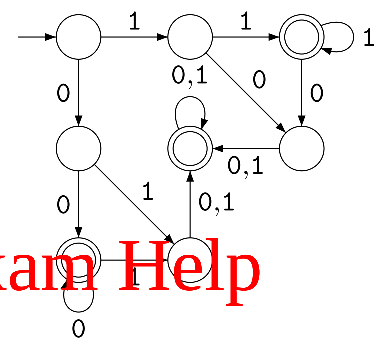
(e) $\{w \mid$ every odd position of w is a `1`$\}$



(f) $\{w \mid w$ contains at least two 0s and at most one 1$\}$

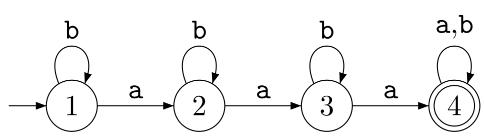


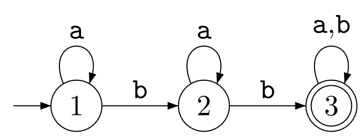
(g) $\{w \mid$ the last symbol of w occurs at least twice in $w\}$

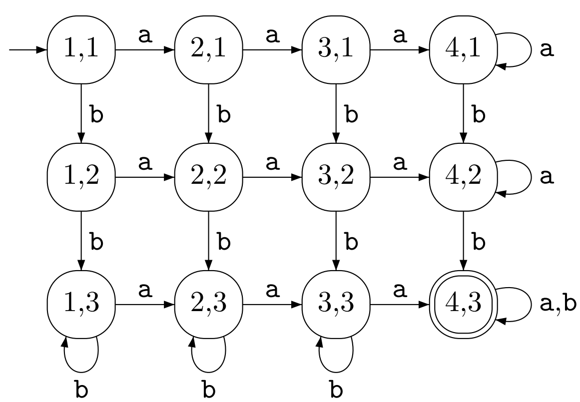


(h) All strings except t

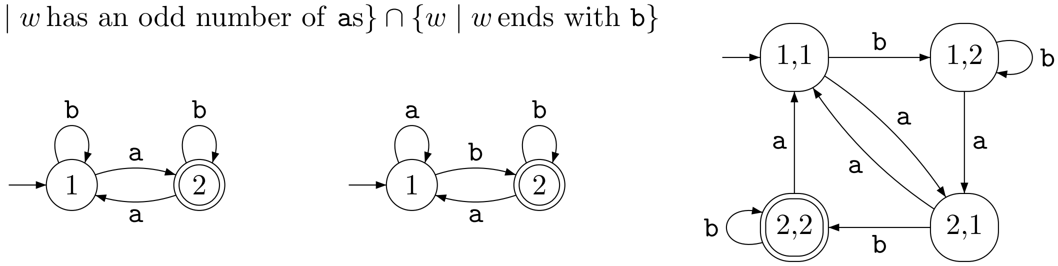
P8.2 (a) $\{w \mid w$ has at least three as$\} \cap \{w \mid w$ has at least two bs$\}$



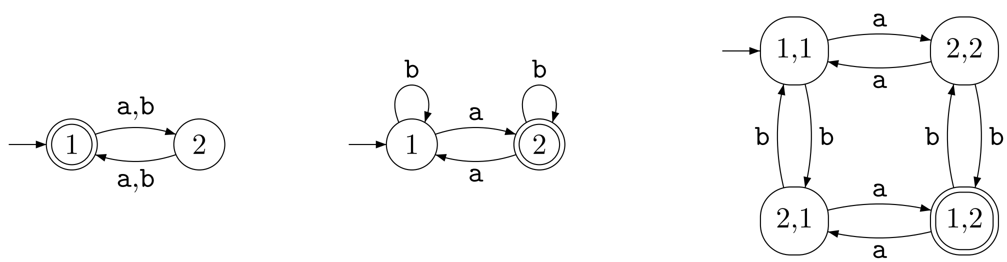




(b) $\{w \mid w$ has an odd number of as$\} \cap \{w \mid w$ ends with b$\}$

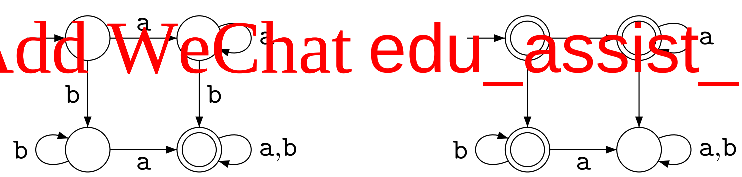


(c) $\{w \mid w$ has an even length$\} \cap \{w \mid w$ has an odd number of as$\}$

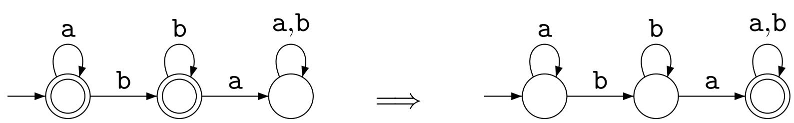


P8.3 (a) $\{w \mid w$ does not contain the substring bb$\}$

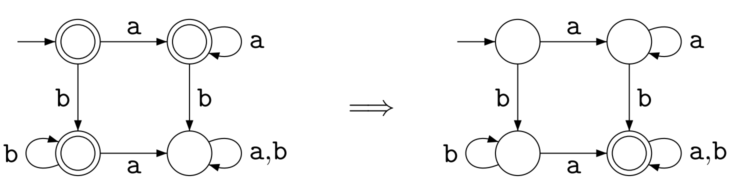
(b) $\{w \mid w$ conta



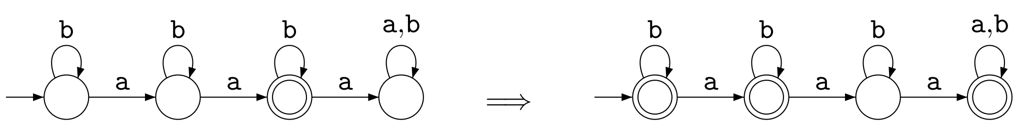
(c) $\{w \mid w$ is any string not in $\mathtt{A}^* \circ \mathtt{B}^*$, where $\mathtt{A} = \{\mathtt{a}\}, \mathtt{B} = \{\mathtt{b}\}\}$



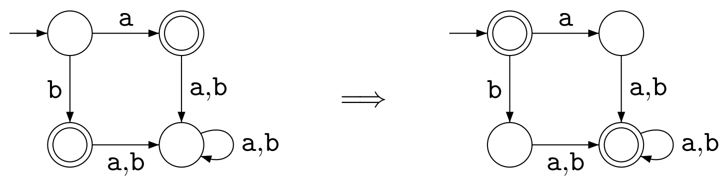
(d) $\{w \mid w$ is any string not in $\mathtt{A}^* \cup \mathtt{B}^*$, where $\mathtt{A} = \{\mathtt{a}\}, \mathtt{B} = \{\mathtt{b}\}\}$ (compare to (b)!)



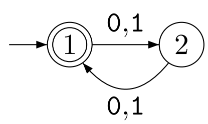
(e) $\{w \mid w$ is any string that doesn't contain exactly two as$\}$

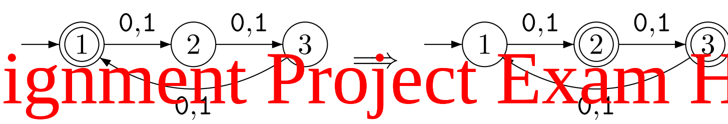


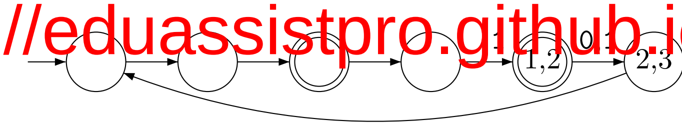
(f) $\{w \mid w$ is any string except a and b$\}$



P8.4 $\{w \mid$ the length of w is a multiple of 2 and is not multiple of 3$\}$







P8.5 (i) Suppose L is regular. Then there is some DFA $D = (Q, \Sigma, \delta, q_0, F)$ which recognises L. Another way to say this is that the language recognised by D is exactly L. We define the language recognised by D as the set

$$L(D) = \{w \in \Sigma^* \mid D \text{ accepts } w\}$$

We claim that L^c is regular, so we must show that there is a DFA D' such that $L(D') = L^c$. Let $D' = (Q, \Sigma, \delta, q_0, Q \setminus F)$, i.e. it has the exact same set of states, transition function and start state as D, but all the non-accept states are now reject states (and vice versa). Then we claim that $L(D') = L^c$, since

$$\begin{aligned} L(D') &= \{w \in \Sigma^* \mid D' \text{ accepts } w\} \\ &= \{w \in \Sigma^* \mid D \text{ rejects } w\} \\ &= \{w \in \Sigma^* \mid w \notin L(D)\} \\ &= \{w \in \Sigma^* \mid w \notin L\} \\ &= L^c \end{aligned}$$

Hence L^c is regular, since the DFA D' recognises it. The core of this proof is the step "D' accepts w iff D rejects w", which can be shown by unwrapping the definition of *acceptance* for DFAs. Another way to explain it, is that if D' rejects w, then after running D' on input w, it should finish in a reject state, $q \notin Q \setminus F$, since $Q \setminus F$ is the set of accept states of D'. But $q' \notin Q \setminus F$ iff $q \in F$. So if we run D on w, it will take the exact same transitions and move through the same states as in D', ending with q, and $q \in F$, so D must accept w. We can reason similarly to show that if D accepts w then D' rejects w.

(ii) Before we dive into the proof, note that we assume L and K are languages over the same alphabet. If we wanted to intersect languages over distinct alphabets, we could think of them as languages over the union of their alphabets. Suppose L and K are regular languages. Then there are DFAs

$$D_L = (Q_L, \Sigma, \delta_L, q_{L0}, F_L)$$
$$D_K = (Q_K, \Sigma, \delta_K, q_{K0}, F_K)$$

such that $L(D_L) = L$ and $L(D_K) = K$. Define

$$D' = (Q_L \times Q_K, \Sigma, \delta', (q_{L0}, q_{K0}), F_L \times F_K)$$

where $\delta' : (Q_L \times Q_K) \times \Sigma \to Q_L \times Q_K$ is defined

$$\delta'((q_1, q_2), x) = (\delta_L(q_1, x), \delta_K(q_2, x))$$

Check that this definition makes sense, by inspecting the function signature of δ_L : $Q_L \times \Sigma \to$ ... $'$

We claim that ... irstly we could show by i

if D_L is in state $q_1 \in Q_L$ after consumi ... K ... 2 ... K ... suming input w, then D' is in state (q_1, q_2) after consuming input ... e empty string, since D' starts in state (q_{L0}, q_{K0}). Now ... gth n.

We want to show that it's also true for any string of length ... t wx be such a string, where w is a string of length n and $x \in \Sigma$ is a symbol. Then after consuming input w on each of D_L, D_K, D', if D_L is in state q_1 and D_K is in state q_2, we know by the inductive hypothesis that D' is in state (q_1, q_2). After then consuming x on all three machines, we know that D_L will be in state $\delta_L(q_1, x)$, and D_K will be in state $\delta_K(q_2, x)$. So we just need to show that D' will be in state $(\delta_L(q_1, x), \delta_K(q_2, x))$ after consuming x. But this is true by definition, since,

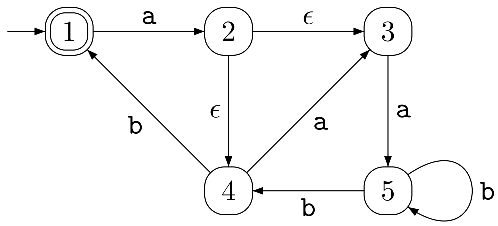
$$\delta'((q1, q2), x) = (\delta_L(q_1, x), \delta_L(q_2, x))$$

Then we can show $L(D') = L \cap K$ directly, since

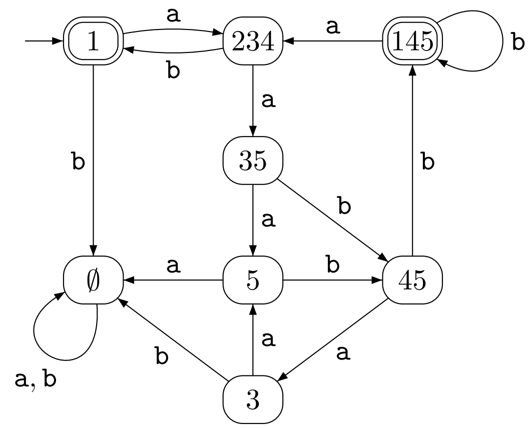
$$\begin{aligned}
L(D') &= \{w \in \Sigma^* \mid D' \text{ accepts } w\} \\
&= \{w \in \Sigma^* \mid D' \text{ is in state } (q_1, q_2) \text{ after consuming } w \text{ and } (q_1, q_2) \in F_L \times F_K\} \\
&= \left\{ w \in \Sigma^* \;\middle|\; \begin{array}{l} D_L \text{ is in state } q_1 \text{ after consuming } w \text{ and } q_1 \in F_L, \\ D_K \text{ is in state } q_2 \text{ after consuming } w \text{ and } q_2 \in F_K \end{array} \right\} \\
&= \left\{ w \in \Sigma^* \;\middle|\; \begin{array}{l} D_L \text{ accepts } w, \\ D_K \text{ accepts } w \end{array} \right\} \\
&= \{w \in \Sigma^* \mid D_L \text{ accepts } w\} \cap \{w \in \Sigma^* \mid D_K \text{ accepts } w\} \\
&= L \cap K
\end{aligned}$$

(iii) Suppose L and K are both regular. Then K^c is regular using (i), and therefore $L \cap K^c$ is regular using (ii). But $L \setminus K = L \cap K^c$, so we are done.

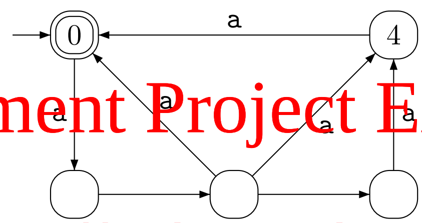
P8.6 From this NFA:



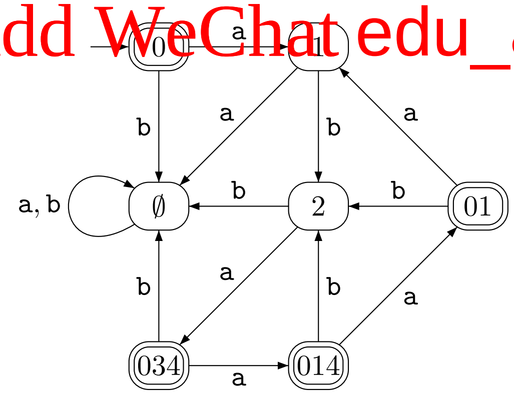
we get:



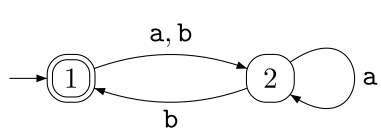
P8.7 From this NFA:



we end up with the following DFA:



P8.8 This is the minimal DFA:



P8.9 This is the minimal DFA: