School of Computing and Information Systems
COMP30026 Models of Computation
Week 7: Regular Languages, DFAs and NFAs

Homework problems

P7.1 Draw DFAs recognising the following languages. Assume that the alphabet $\Sigma = \{\texttt{0}, \texttt{1}\}$.

(i) $\{w \mid w$ is not empty and contains only 0s or only 1s$\}$

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

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

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

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

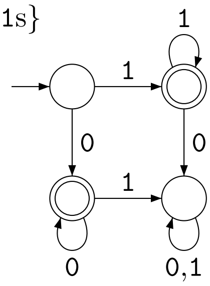
(vi) $\{w \mid w$ contains at least two 0s and at most one 1$\}$

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

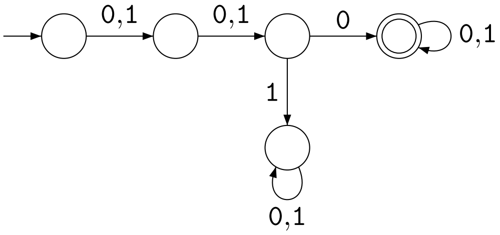
(viii) All strings except the empty string

Solution:

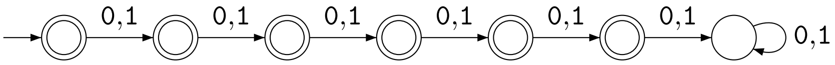
(a) $\{w \mid w$ is not empty and contains only 0s or only 1s$\}$



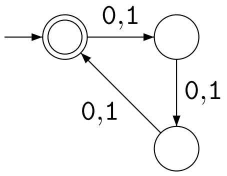
(b) $\{w \mid w$ has length at least 3 and its third symbol is 0$\}$



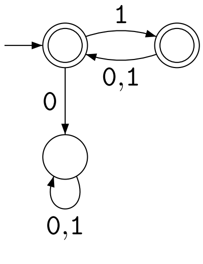
(c) $\{w \mid$ the length of w is at most 5$\}$



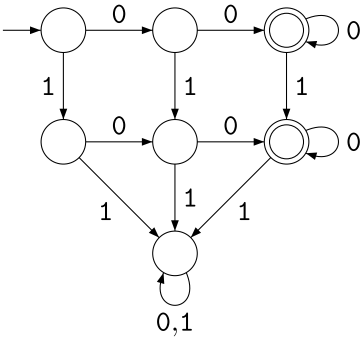
(d) $\{w \mid$ the length of w is a multiple of 3$\}$



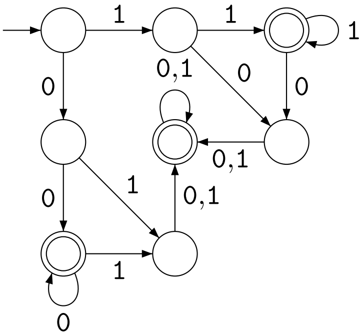
(e) $\{w \mid$ every odd position of w is a 1$\}$



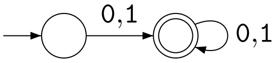
(f) $\{w \mid w$ contains at least two `0`s and at most one `1`$\}$



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



(h) All strings except the empty string



P7.2 Each of the following languages is the intersection of two simpler languages. First construct the DFAs for the simpler languages, then combine them using the construction from **??** to create a DFA for the intersection language

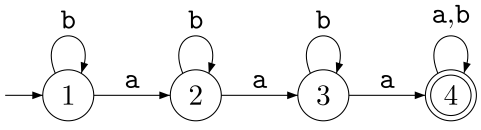
(a) $\{w \mid w$ has at least three `a`s and at least two `b`s$\}$

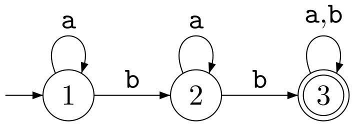
(b) $\{w \mid w$ has an odd number of `a`s and ends with `b`$\}$

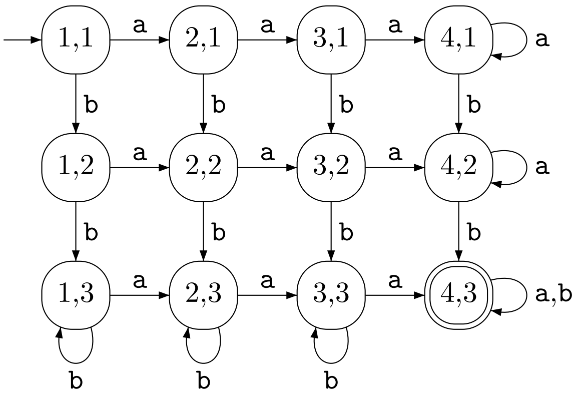
(c) $\{w \mid w$ has an odd number of `a`s and has even length$\}$

Solution:

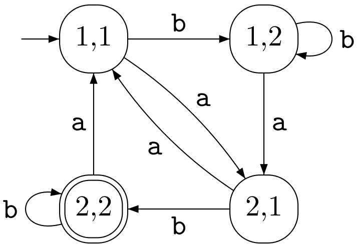
(a) $\{w \mid w\,$has at least three `as`$\} \cap \{w \mid w\,$has at least two `bs`$\}$

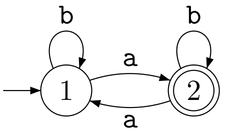


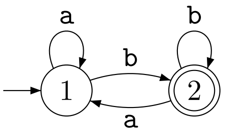




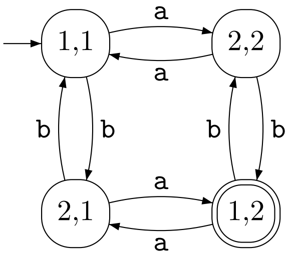
(b) $\{w \mid w\,\text{has an odd number of}\ \texttt{a}\text{s}\} \cap \{w \mid w\,\text{ends with}\ \texttt{b}\}$

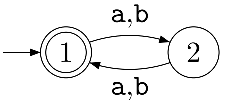


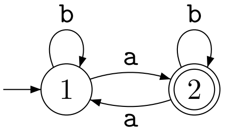




(c) $\{w \mid w\,\text{has an even length}\} \cap \{w \mid w\,\text{has an odd number of}\ \texttt{a}\text{s}\}$







P7.3 Each of the following languages is the complement of a simpler language. Again, the best way to proceed is to first construct a DFA for the simpler language, then find a DFA for the complement by transforming that DFA appropriately. Throughout this question, assume that the alphabet $\Sigma = \{\texttt{a}, \texttt{b}\}$.

(a) $\{w \mid w$ does not contain the substring $\texttt{bb}\}$

(b) $\{w \mid w$ contains neither the substring $\texttt{ab}$ nor $\texttt{ba}\}$

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

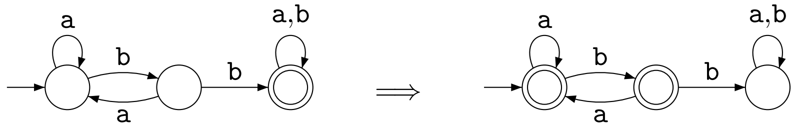
(d) $\{w \mid w$ is any string not in $\texttt{A}^* \cup \texttt{B}^*$, where $\texttt{A} = \{\texttt{a}\}, \texttt{B} = \{\texttt{b}\}\}$

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

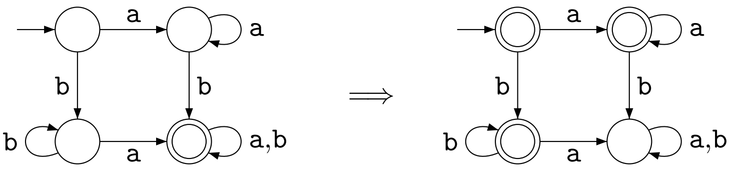
(f) $\{w \mid w$ is any string except $\texttt{a}$ and $\texttt{b}\}$

Solution:

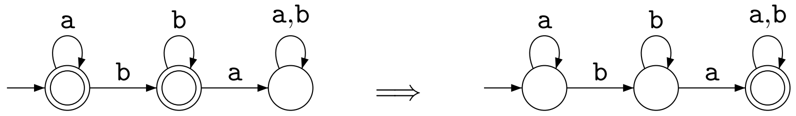
(a) $\{w \mid w\,\text{does not contain the substring}\ \texttt{bb}\}$



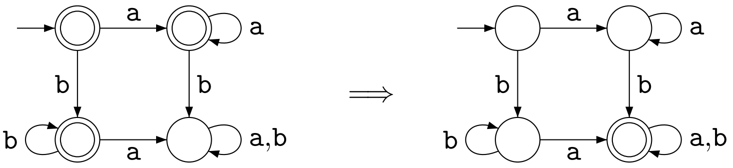
(b) $\{w \mid w \text{ contains neither the substring } \texttt{ab} \text{ nor } \texttt{ba}\}$



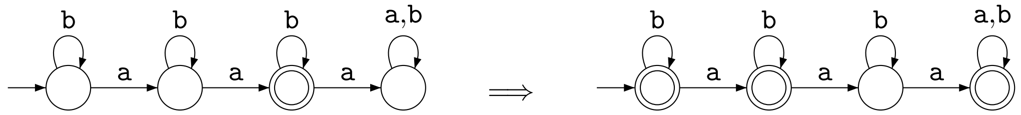
(c) $\{w \mid w \text{ is any string not in } \texttt{A}^* \circ \texttt{B}^*, \text{ where } \texttt{A} = \{\texttt{a}\}, \texttt{B} = \{\texttt{b}\}\}$



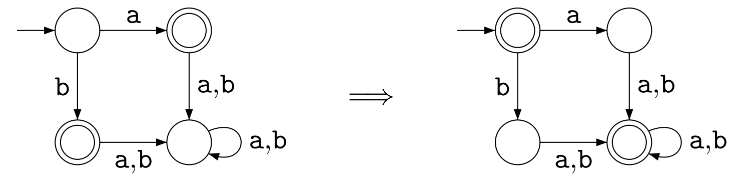
(d) $\{w \mid w \text{ is any string not in } \texttt{A}^* \cup \texttt{B}^*, \text{ where } \texttt{A} = \{\texttt{a}\}, \texttt{B} = \{\texttt{b}\}\}$ (compare to (b)!)



(e) $\{w \mid w \text{ is any string that doesn't contain exactly two } \texttt{a}\text{s}\}$



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

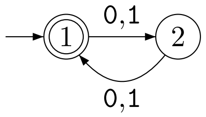


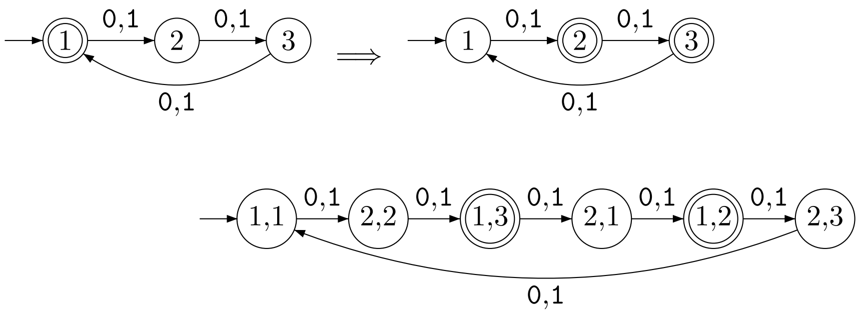
P7.4 The language

$$L = \{w \mid \text{the length of } w \text{ is a multiple of 2 and is not multiple of 3}\}$$

is the difference of two simpler languages. First construct DFAs for the simpler languages, then construct a DFA for L. Assume the alphabet $\Sigma = \{\texttt{a}, \texttt{b}\}$.

Solution: $\{w \mid \text{the length of } w \text{ is a multiple of 2 and is not multiple of 3}\}$





P7.5 A language is regular iff it is recognised by some DFA. Prove that for any languages $L, K \subseteq \Sigma^*$

(i) If L is regular then L^c is regular

(ii) If L and K are both regular and then $L \cap K$ is regular

(iii) If L and K are both regular and then $L \setminus K$ is regular

Another way to say this is "The class of regular languages is closed under intersection, complement and difference". You have demonstrated this already for some example languages. For this exercise you should specify how this works in general, using the formal definition of a DFA. *Hint:* Your proof for (iii) can be much shorter.

Solution:

(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 accept 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

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

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 $\delta_L : Q_L \times \Sigma \to Q_L$ and $\delta_K : Q_K \times \Sigma \to Q_K$, and δ'.
We claim that $L(D') = L \cap K$. There are a few ways of reasoning about this. Firstly we could show by induction on the length of the input string that if D_L is in state $q_1 \in Q_L$ after consuming input w, and D_K is in state $q_2 \in Q_K$ after consuming input w, then D' is in state (q_1, q_2) after consuming input w. This is true by definition for the empty string, since D' starts in state (q_{L0}, q_{K0}). Now suppose this true for strings of length n. We want to show that it's also true for any string of length $n + 1$. Let 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,

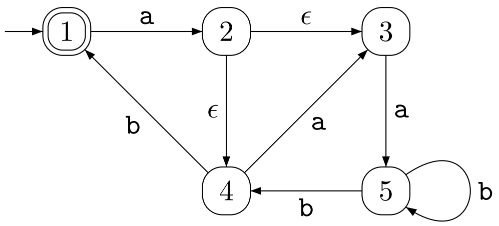
$$\delta'((q1, q2), x) = (\delta_L(q_1, x), \delta_L(q_2, x))$$

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

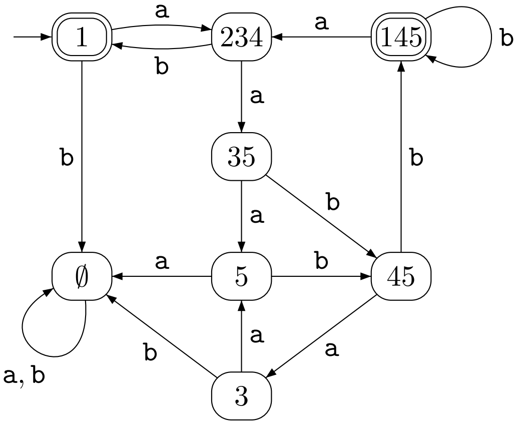
$$\begin{aligned} L(D') &= \{w \in \Sigma^* \mid D' \text{ accepts } w\} \\ &= \{w \in \Sigma^* \mid D' \text{ is in state } (q_1, q_2) \text{ after consuming } w \text{ and } (q_1, q_2) \in F_L \times F_K\} \\ &= \left\{ w \in \Sigma^* \;\middle|\; \begin{array}{l} D_L \text{ is in state } q_1 \text{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

(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.

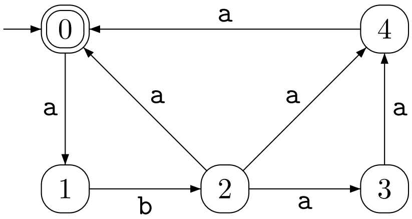
P7.6 Use the subset construction method to turn this NFA into an equivalent DFA:



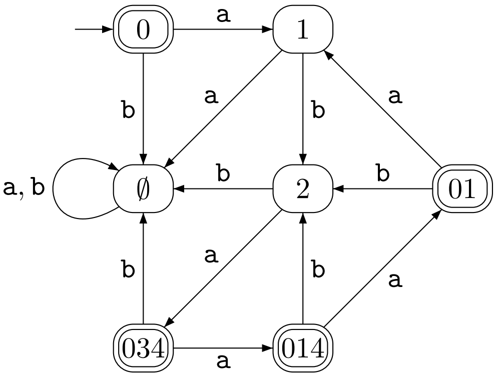
Solution:



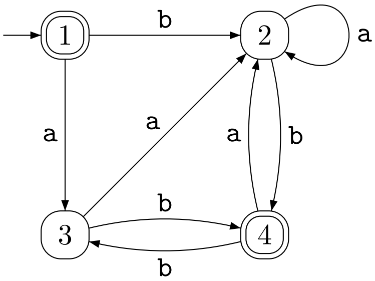
P7.7 Use the subset construction method to turn this NFA into an equivalent DFA:



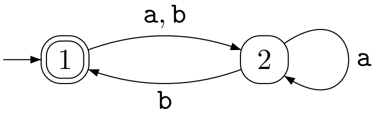
Solution:



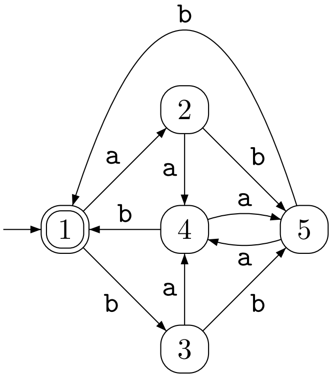
P7.8 Construct a minimal DFA equivalent to this one:



Solution:



P7.9 Construct a minimal DFA equivalent to this one:



Solution: