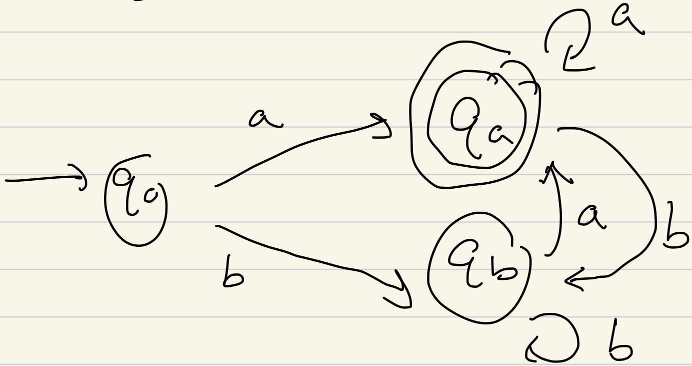
Homework 6 Solutions CPSC 421/501 2025

(c) Joel Friedman

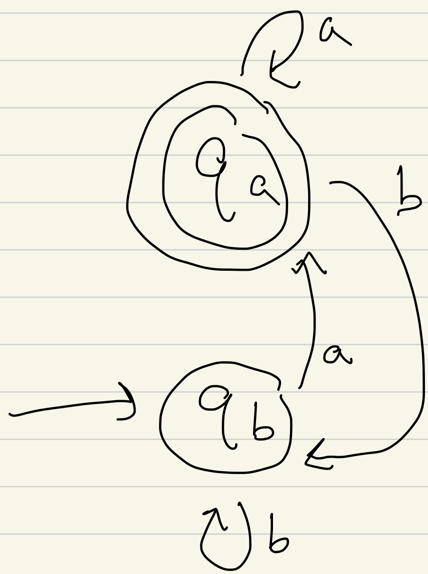
(1) (a) Perhaps the most natural DFA has 3 states:




which rejects the empty string $\varepsilon$ in $q_0$, and then transitions to two states, a state $q_a$ for when the last symbol read is an a, the other for b, $q_b$

However, we don't really need $q_0$,

since we can equivalenty start in $q_b$ (which then rejects the empty string). Hence the two-state DFA




also recognizes $L$.

(b) We similarly build a DFA that remembers the last 2 letters, with states

$$Q = \{q_{aa}, q_{ab}, q_{ba}, q_{bb}\}$$

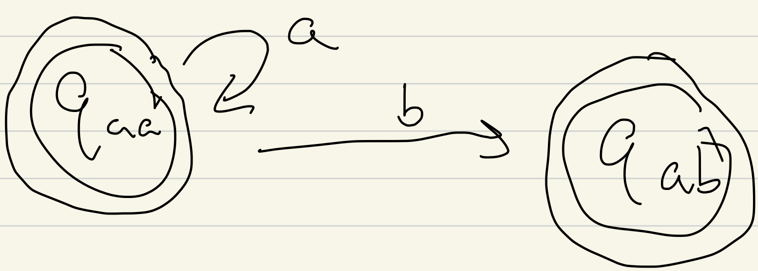
where $q_{xy}$ means that the last two symbols are $xy$.

If the 2nd to last symbol is "a" then we accept, so the accepting (or "final") states are

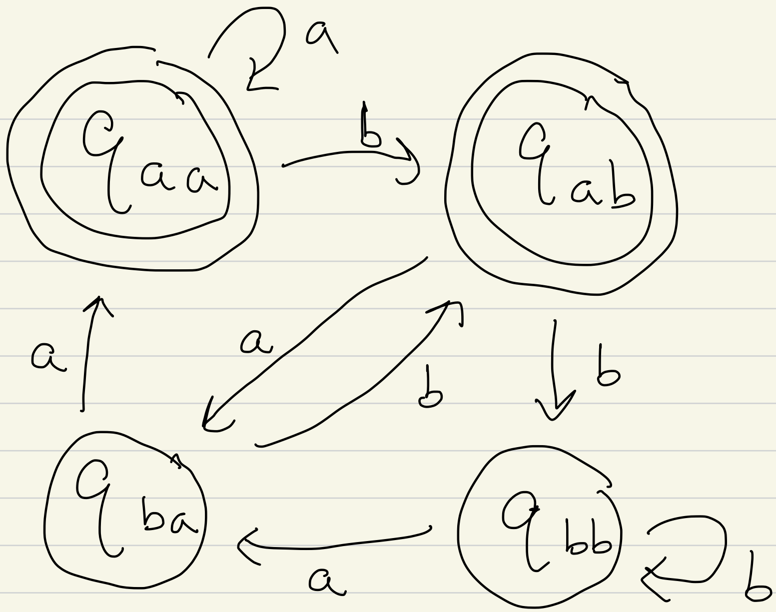
$$F = \{q_{aa}, q_{ab}\}.$$

If we are is state $q_{aa}$ and

we read an "a", then the new last two symbols are still aa, whereas if we read a "b" then the ~~new~~ new two last symbols are ab. Hence we have the transitions

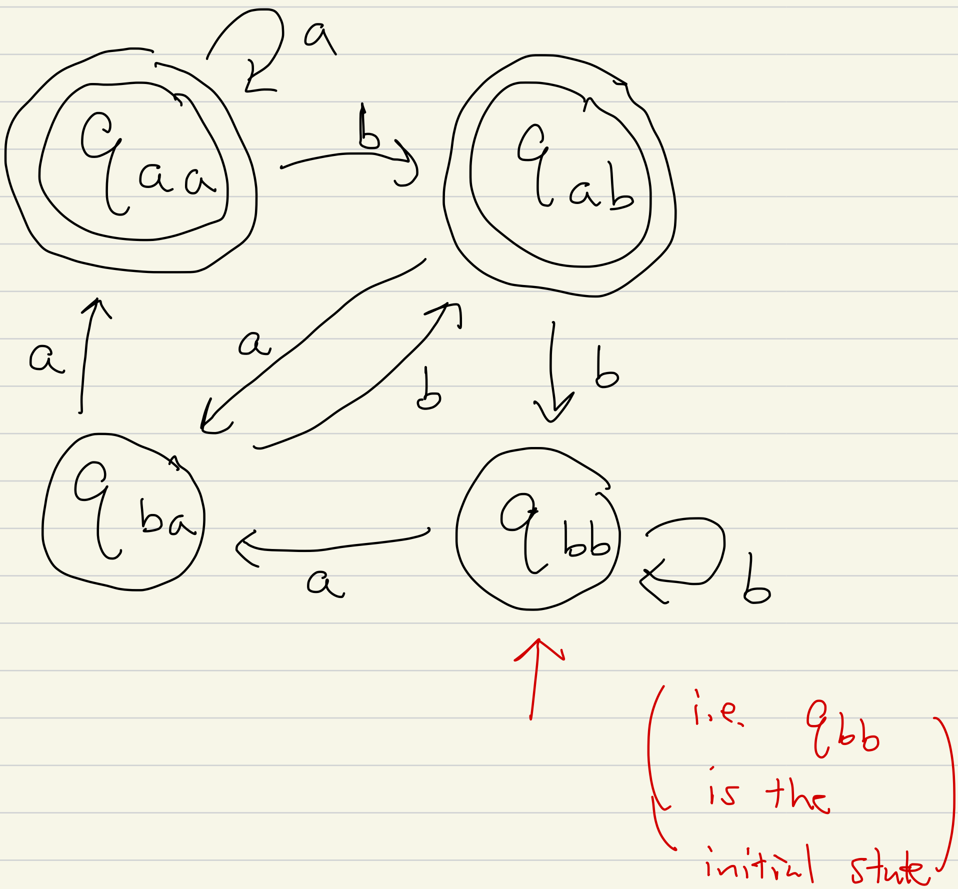



doing similarly for the other 3 states we get a partial DFA

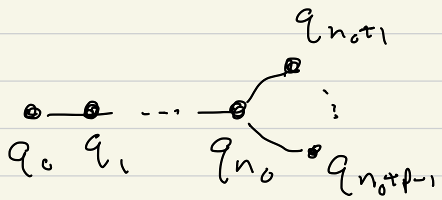



Now we see that we can take $q_{bb}$ as the initial state, because doing so will reject $\varepsilon$, a, b and after two steps through the DFA we get to the state that appropiately represents the last two symbols we've seen.

Hence a 4-state DFA accepting L is



(2) Since $L$ is regular, it has an eventual period $p \in \mathbb{N}$, and is recognized by a DFA



So if $n_i \geq n_0$ and $a^{n_i} \in L$ we have that

$$a^{n_i} \in L \iff a^{n_i+p} \in L$$

and hence $a^{n_i+p} \in L$ and so $n_{i+1} \leq n_i + p$.

Since $n_1 < n_2 < n_3 < \cdots$, there are only finitely many $i$ such that $n_i < n_0$.

Letting $i'$ be the largest integer

such that $n_{i'} < n_0$, it follows

that

$$i > i' \implies n_{i+1} \leq n_i + p.$$

Setting

$$p' = \max(n_2 - n_1, n_3 - n_2, \ldots, n_{i'+1} - n_{i'}),$$

it follows that ($p'$ is finite, i.e. $p' \in \mathbb{N}$)

and for any $i$,

$$n_{i+1} \leq n_i + \max(p, p').$$

(3) Quick solution (although we hadn't covered regular expressions at the time...):

Regular expressions, as in [Sip], can be written as strings over:

$$\Sigma_+ \stackrel{def}{=} \Sigma \cup \Sigma_1 ,$$

where

$$\Sigma_1 = \{ (, ), \cup, \circ, {}^* \}$$

(for simplicity we assume that $\Sigma \cap \Sigma_1 = \emptyset$, otherwise there are many workarounds, e.g. let $\Sigma_+ \stackrel{def}{=} \Sigma \amalg \Sigma_1$, where $\Sigma \amalg \Sigma_1$ is the disjoint union (you can use ChatGPT / Gemini / etc. to look this up if you don't know what the disjoint union, $\amalg$, means when $\Sigma$ and $\Sigma$ intersect).

Another route, using only [Sip], section 1.1:

Let $L$ be regular, hence recognized by a DFA $= (Q, \Sigma, \delta, q_0, F)$, and order the states of $Q$ as $(q_0, q_1, \ldots, q_s)$ (hence $|Q| = s+1$).

Using the bijection

$$\begin{aligned} q_0 &\mapsto 1 \\ q_1 &\mapsto 2 \\ &\vdots \\ q_s &\mapsto s+1 \end{aligned}$$

we may replace $Q = \{q_0, \ldots, q_s\}$

to have $Q = [s+1] = \{1, 2, \ldots, s+1\}$.

Hence $L$ is a recognized by a DFA

$$\tilde{M} = ([s+1], \Sigma, \delta, 1, \tilde{F})$$

where

$$\delta : [s+1] \times \Sigma \to [s+1].$$

Hence there ~~are~~ are only finitely many such machines $\tilde{M}$.

Hence

$$\mathcal{L}_s = \left\{ L \subset \Sigma^* \;\middle|\; \begin{array}{c} L \text{ is recognized by} \\ \text{a DFA with } s+1 \\ \text{states} \end{array} \right\}$$

is a finite set.

Hence the set of regular languages is

$$\mathcal{L}_1 \cup \mathcal{L}_2 \cup \dots$$

which is countable

(in more detail, we can create a map

$$f : \mathcal{L}_1 \cup \mathcal{L}_2 \cup \dots \cup \longrightarrow \mathbb{N}$$

such that each regular language is in the image of $f$).

(4)(a) [There are a number of possible solutions here.] Say that

$$n_1 = 3,\ n_3 = 6,\ n_5 = 9,\ n_7 = 12, \dots$$

Then $3 < n_2 < 6$ so $n_2 =$ either 4 or 5

and $6 < n_4 < 9$ so $n_4 =$ either 7 or 8

$\vdots$

and, for any $j \in \mathbb{N}$

$$n_{2j} = \text{either } 3j+1 \text{ or } 3j+2.$$

To each $S \subset \mathbb{N}$, we can therefore satisfy the above, taking

$$n_{2j} = \begin{cases} 3j+1 & \text{if } j \in S \\ 3j+2 & \text{if } j \notin S \end{cases}$$

This gives a map

$$f : \text{Power}(\mathbb{N}) \longrightarrow \{\text{sequences } n_1 < n_2 < \dots\}$$

which is injective (since if $S, S' \in \text{Power}(\mathbb{N})$ and $S \neq S'$, then for some $j \in \mathbb{N}$ either $j \in S$ and $j \notin S'$ or vice versa, and so $n_{2j}$ meaning $n_{2j}(S)$ and $n_{2j}(S')$ are different). Hence $f$ is injective. Since $\text{Power}(\mathbb{N})$ is uncountable, so is the image of $f$.

Since $n_{2j} = 3j+1, 3j+2$ and $n_{2j-1} = 3j$ and $n_{2j+1} = 3j+3$, we have

$$\forall i \in \mathbb{N} \quad n_{i+1} - n_i = 1 \text{ or } 2.$$

[There are many other injections

$$f : \text{Power}(\mathbb{N}) \longrightarrow \{\text{sequences } n_1 < n_2 < \ldots\}$$

that one can give, such as

$$f(S) = \{2i \mid i \in S\} \cup \{1,3,5,7,\ldots\}.]$$

(b) By problem (3), there are only countably many regular languages, but uncountably many languages described in (a). Hence there is at least one language of type (a) that is not regular.